CONVOLUTIONAL NEURAL NETWORK FOR FACIAL EMOTION RECOGNITION WITH GEOMETRICAL FEATURES OF FACE

Capstone Project

İlker Arslan

İSTANBUL, 2021

MEF UNIVERSITY

CONVOLUTIONAL NEURAL NETWORK FOR FACIAL EMOTION RECOGNITION WITH GEOMETRICAL FEATURES OF FACE

Capstone Project

İlker Arslan

Advisor: Asst. Prof. Dr. Tuna Çakar

İSTANBUL, 2021

MEF UNIVERSITY

Name of the project: Convolutional Neural Network for Facial Emotion Recognition Predictions with Geometrical Features of Face
Name/Last Name of the Student: İlker Arslan
Date of Project Report Submission: 30/06/2021

I hereby state that the graduation project prepared by İlker Arslan has been completed under my supervision. I accept this work as a "Graduation Project".

30/06/2021
Asst. Prof. Tuna Çakar

I hereby state that I have examined this graduation project by İlker Arslan which is accepted by his supervisor. This work is acceptable as a graduation project and the student is eligible to take the graduation project examination.

30/06/2021

Director
of
Information Technologies
Program

We hereby state that we have held the graduation examination of İlker Arslan and agree that the student has satisfied all requirements.

THE EXAMINATION COMMITTEE

Committee Member	Signature /Date
1. Asst. Prof. Tuna Çakar	……..
2. …….	……..

Academic Honesty Pledge

I promise not to collaborate with anyone, not to seek or accept any outside help, and not to give any help to others.

I understand that all resources in print or on the web must be explicitly cited.

In keeping with MEF University's ideals, I pledge that this work is my own and that I have neither given nor received inappropriate assistance in preparing it.

İlker Arslan 30.06.20201 Signature

EXECUTIVE SUMMARY

CONVOLUTIONAL NEURAL NETWORK FOR FACIAL EMOTION RECOGNITION WITH GEOMETRICAL FEATURES OF FACE

İlker Arslan

Advisor: Asst. Prof. Tuna Çakar

JUNE 2021, 51 pages

One of the recent challenging machine learning problems is to make predictions on image datasets. The aim of the project is to construct a convolutional neural network to guess emotions for a face of a human given in an image file considering the face. After the geometrical features are extracted using pretrained models, we construct five models which are convolutional networks fed with handcrafted geometrical features extracted. The last model uses the outputs of other four models to predict more accurately.

Key Words: Facial emotion recognition, convolutional neural network, AffectNet.

ÖZET

YÜZÜN GEOMETRİK ÖZELLİKLERİYLE YÜZSEL DUYGU TANIMASI İÇİN EVİRİMŞİMSEL SİNİR AĞLARI

İlker Arslan

Proje Danışmanı: Dr. Öğr. Üyesi Tuna Çakar

HAZİRAN, 2021, 51 sayfa

Son zamanlardaki en zorlu makine öğrenmesi problemlerinden biri resim dosyaları üzerinde tahminlerde bulunmaktır. Projenin amacı evrişimli sinirsel ağları kullanarak resim dosyasındaki resmi verilmiş insan yüzünü değerlendirerek resimdeki insanın duygu durumunu tahmin etmeye çalışmaktır. Daha önceden eğitilmiş modelleri kullanarak yüzün geometric özellikleri çıkartıldıktan sonra bu özelliklerle beslenene beş tane evirişimli sinirsel ağ yapılandırılmıştır. Son model daha iyi bir tahminde bulunmak için diğer dört modelin sonuçlarını kullanmıştır.

Anahtar Kelimeler: Yüzsel duygu durumu, evirimşel sinirsel ağ, AffectNet.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Making predictions on the image files is one of the most popular and important challenging problems among all artificial intelligence problems. The main issues can vary from detecting specific pictures of objects inside the image files such as cars, human faces, parts of nature, etc to extract stories of what is happening in those pictures such as car crashes, natural disasters, etc. One of the famous models to solve these is convolutional neural networks (CNN), which are especially specific to be successful at learning images. In this project, we try to estimate the probabilities of different emotions of a human-being face given in an image file which is a pretty challenging problem. There are eight emotions which are 'happiness, sadness, surprise, fear, disgust, anger, contempt, neutral' and another category of image 'no-face', where 'no-face' means that there is no person shown up in the image. Although there are more than these emotion types and even more complex ones, we worked on these eight because it gets harder to detect when you add more types especially if they seem to be close to other types. For example, 'happiness' and 'surprise' or 'disgust' and 'contempt' sometimes seem to be confusing to distinguish even to a person looking at the image on the screen.

1.1. On Applications of Facial Emotion Recognition (FER)

Detecting the mood/state of a person is important for automated human-interacted systems to mimic the interactions between two people and is used for commercial aims. For instance, the application 'iMotions' uses its face emotion recognition algorithm(s) to extract information and has products (iMotions). Another application is that there are serious discussions on the study of FER about how FER may indicate early warnings of neurological diseases to be diagnosed and treated (Pietschnig J. et al, 2016). An interesting example is that there are works on micro-expressions, which are repressed, of human faces to detect lies as well as detecting emotions of drivers for dealing with fatigue states (N. Rodrigez-Diaz et al, 2021), (Z. Kowalczk et al, 2019).

1.2. About Works Through Comparison of Extracting Hand-crafted Features and Deep Learnt Features

In this section, we go through the survey (S. Li et al, 2019) for a brief history and methods of research on FER. Afterwards, we will present some of the details of the deep learning methods applied in (Dunau P. et al, 2019), (G. Viswanatha Reddy et al, 2020), (M. A. Jalal et al, 2019).

In (S. Li at al, 2019), the authors mention that the learning methods for FER were usually based on handcrafted features until the competitions such as FER2013 and Emotion Recognition in the Wild (EmotiW) leading to adequate data for deep learning methods to be applied for FER. Recently known databases presented in (S. Li at al, 2019) are CK+, MMI, Oulu-CASIA, JAFFEE, FER2013, AFEW, SFEW, Multi-PIE, BU-3DFE, BU-4DFE, EmotioNet, RAF-DB, AffectNet, ExpW, 4DFAB. These datasets can differ in many aspects, such as, of the number of images, the way they are annotated (manually of automated), the number of different subjects used, the number of emotional expressions to label images, the sources from which they created (internet, lab, movie, …). As an instance, the dataset AffectNet has two groups of image files which are imported from the web sources using search algorithms. One group is annotated manually having 420,229 images and the other group is annotated automatically and has 550,000 images. The number of labels is eleven and they are 'neutral, happiness, sadness, surprise, fear, disgust, anger, contempt, none, uncertain, no-face'. Each image is also provided with their valence and arousal features. Valence property is about how positive or negative the image is and arousal property is about how intense the emotion is. These two are represented by values between -1 and 1 and by -2 for uncertain and no-face categories. The authors of (G. Viswanatha Reddy et al, 2020) mention that this dataset is labelled by 12 expert human annotators at the University of Denver and In the documentation of AffectNet, it is said ResNext Neural Network is used for automatic annotation trained on the manually annotated training set samples with average accuracy of 65%. The 68 facial landmarks are also given corresponding to each image together with the position of boxes in which faces are located.

The authors in (S. Li at al, 2019) emphasize three important data preprocessing steps which are face alignment, data augmentation, and face normalization. Face alignment is said to be crucial in terms of its effect on performance and efficiency. Data augmentation is said to be for obtaining sufficient data and generalizing, which may be made through cropping, flipping horizontally, random perturbations (such as shifting up/down, rotations, skew, scaling, noise, …). Lastly, in order to prohibit variations in illuminations and head poses, facial normalization is applied according to (S. Li at al, 2019).

An example of a method based on handcrafted features is treated in the conference paper (Dunau P. et al, 2019) by considering angle and size information extracted from the image. They carried the process of learning from an image sequentially getting the features and classifying emotions only based on the features extracted. The features extracted are angles and lengths/sizes which are formulated geometrically in terms of the location of landmarks obtained

from ‘dlib’, the facial landmark detector (King, Davis E., 2009). They used Generalized Procrustes Analysis (GPA) to get the appropriate set of landmark features to be used comparatively with another set of features, what they call as ASF (Angle and Size Features), obtained by geometrical results from locations of eyes and mouth. They apply PCA for each of the set of features before they are classified by the multilayer perceptron (MLP). The emotions to be classified are anger, disgust, fear, happiness, sadness, and surprise which are, in total, six. The below figure captured from (Dunau P. et al, 2019) is their accuracies for both of the set of the features.

Features	Class	Precision	Recall	F_1-Score	Accuracy
Landmarks	Anger (0)	$\mathbf{0.77 \pm 0.11}$	0.72 ± 0.18	$\mathbf{0.74 \pm 0.14}$	
	Disgust (1)	0.81 ± 0.02	0.86 ± 0.14	0.83 ± 0.06	
	Fear (2)	0.79 ± 0.12	0.70 ± 0.13	0.73 ± 0.06	
	Happiness (3)	0.85 ± 0.04	$\mathbf{0.93 \pm 0.07}$	0.89 ± 0.03	
	Sadness (4)	$\mathbf{0.83 \pm 0.11}$	0.81 ± 0.13	$\mathbf{0.82 \pm 0.11}$	
	Surprise (5)	0.96 ± 0.04	0.93 ± 0.06	0.94 ± 0.02	
	Average	0.83 ± 0.06	0.83 ± 0.09	0.82 ± 0.08	0.84 ± 0.02
ASF	Anger (0)	0.75 ± 0.04	$\mathbf{0.72 \pm 0.16}$	0.72 ± 0.08	
	Disgust (1)	$\mathbf{0.85 \pm 0.12}$	$\mathbf{0.92 \pm 0.07}$	$\mathbf{0.88 \pm 0.06}$	
	Fear (2)	$\mathbf{0.85 \pm 0.09}$	$\mathbf{0.76 \pm 0.11}$	$\mathbf{0.79 \pm 0.04}$	
	Happiness (3)	$\mathbf{0.90 \pm 0.03}$	0.92 ± 0.06	$\mathbf{0.91 \pm 0.02}$	
	Sadness (4)	0.82 ± 0.08	$\mathbf{0.81 \pm 0.08}$	0.81 ± 0.03	
	Surprise (5)	$\mathbf{0.98 \pm 0.03}$	$\mathbf{0.94 \pm 0.04}$	$\mathbf{0.96 \pm 0.02}$	
	Average	$\mathbf{0.86 \pm 0.07}$	$\mathbf{0.85 \pm 0.09}$	$\mathbf{0.84 \pm 0.08}$	$\mathbf{0.86 \pm 0.02}$

Figure 1: Accuracies presented in (Dunau P. et al, 2019).

Contrary to (Dunau P. et al, 2019), the authors of the paper (M. A. Jalal et al, 2019) propose convolutional neural networks integrated with a self-attention mechanism without any handcrafted feature extracting. What they claim as a reason why they use a self-attention network is to take the relationships between the regions in the feature maps from previous layers into account. That would give clues about the positions of sub-regions regarding the other subregions within one image. Below is the figure for the architecture they constructed.

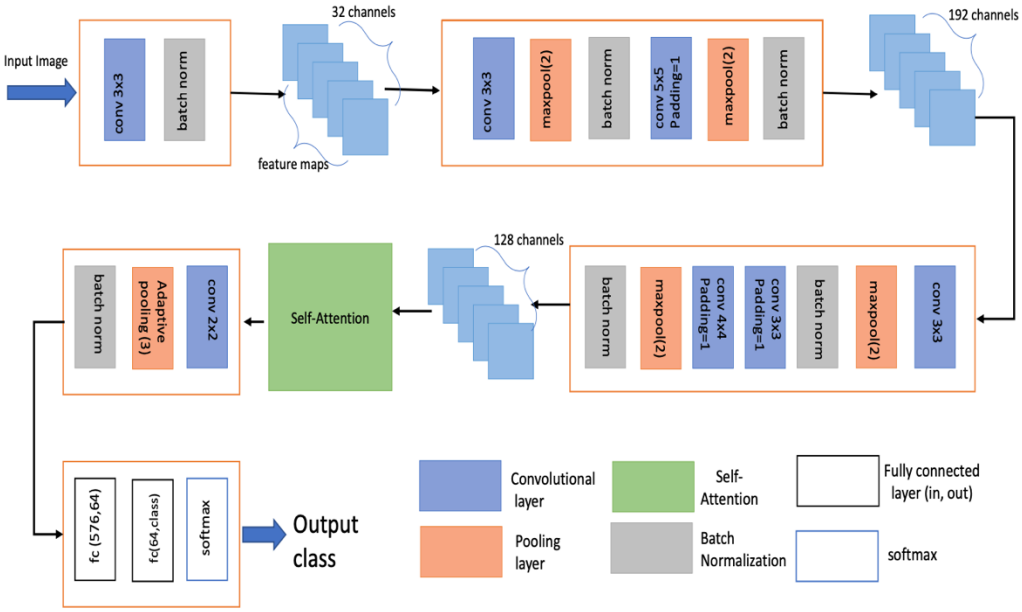

Figure 2: The architecture of the model presented in (M. A. Jalal et al, 2019).

Inside the self-attention network in the above figure, they map features to three spaces **j**, **k** and **l** as

$$j(y) = W_j y, \qquad k(y) = W_k y, \qquad l(y) = W_l y$$

where **y** is the feature from the previous layers and W_j, W_k, W_l are weights to be trained through back-propagation, then a matrix **e** is evaluated composed with the softmax function as each entry is calculated as follows

$$e_{ij} = softmax\left(j(y_i)^T k(y_j)\right)$$

and the outputs are

$$attention\_output = \gamma\big(e.l(y)\big) + y$$

where γ is set to be randomly initialized and the 'dot' represents the matrix multiplication.

The authors of (M. A. Jalal et al, 2019) considered 8 categories of emotions (neutral, happy, sad, surprise, fear, disgust, anger). They chose approximately 13000 images for each category from the dataset AffectNet for training and the test data has 500 images for each category. Furthermore, they augmented the data by applying horizontal flipping and random cropping to increase the diversity and normalized and resized them. The number of epochs they run is about 700 and their accuracy on the validation set is claimed to be 93.8%.

Another paper (G. Viswanatha Reddy et al, 2020) proposes a combined method in which both convolutional neural network and handcrafted feature extracting are used. However, instead of using the given facial landmark information and face region information in AffectNet

they used the implementation presented in (V. Kazemi and J. Sullivan, 2014) for landmark detection and Faster RCNN (S. Ren, 2017) for extracting the face region. Afterwards, they calculated all possible distances between these 68 landmarks. The number of distances is $\binom{68}{2}$ which is 2278. The deep learnt features are obtained by a model, where they made use of the last convolutional layer features of XceptionNet to get 2048 features. Then, they concatenate these two feature vectors (having 4326 components in total) and to reduce the dimension by Principal Component Analysis (PCA). They end the model up with a SVM (Support Vector Machine) classifier. Furthermore, they presented the results with three distinct classifiers as SVM, SVM with Radial Basis Function (RBF), and Neural Net. Their accuracies captured from (G. Viswanatha Reddy et al, 2020) are shown below.

Accuracies of the proposed method with three different classifiers.

Classifiers	Imbalanced set	Down-sampled set	Up-sampled set
SVM (RBF)	0.54	0.58	0.59
SVM (Linear)	0.47	0.52	0.51
Neural Net	0.48	0.53	0.50

Figure 3: The accuracies from (G. Viswanatha Reddy et al, 2020).

Below is the figure of the whole procedure they carried and presented in (G. Viswanatha Reddy et al, 2020).

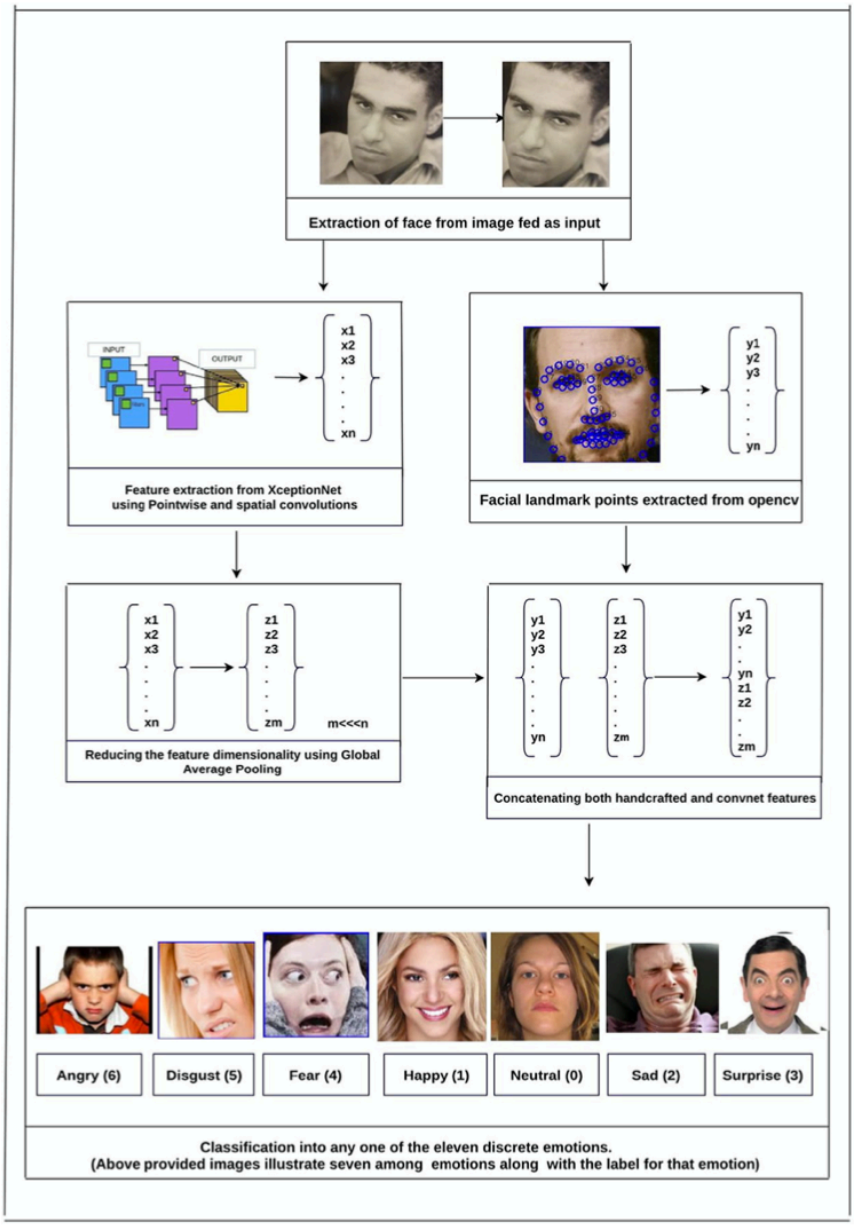

Figure 4: The architecture from (G. Viswanatha Reddy et al, 2020).

The authors (G. Viswanatha Reddy et al, 2020) have done their experiments on the dataset AffectNet which has highly unbalanced number of (manually annotated) images per categories (neutral: 75,374; happy: 134,915; sad: 25,959; surprise: 14,590; fear: 6,878; disgust: 4,303; anger: 25,382; contempt: 4,250; none: 33,588; uncertain: 12,145; non-face: 82,915). In order to avoid this bias over 11 categories they sampled 10,000 images from each class using the data augmentation techniques, where the classes having less than 10,000 images remained the same.

1.3. Usage of Transfer Learning In FER

As well known, models of neural networks are actually pretty complex compositions of linear and/or often nonlinear multivariable functions with lots of parameters (weights/variables) defined by them. Convolutional neural networks may also be pretty much complicated in terms of its (activation) functions which are possibly non-linear, non-convex (such as sigmoid) even non-smooth (such as Relu). More or less, a layer of a convolutional neural network may be considered as one of the functions which are composed to build that model. The more nodes (neurons) lead to higher number of weights (variables of the functions which are composed). The backpropagation process is designed to update these weights to minimize the loss function, that is the distance, defined by various metrics, between model evaluations (outputs of models) and exact target values. The point is that the backpropagation slows down by consuming much time when the number of weights is high and most of the recent problems need a large number of weights. In addition to the high number of weights, training is done with numerous epochs because the loss functions in complex models are probably not easy to optimize due to loss functions dependent on non-linear/non-convex activation functions, in particular, for convolutional neural networks. One may think that if a successful convolutional neural network model is trained by so much effort, it may be used for other tasks as well. This usage is, what is called, “transfer learning”. Transfer learning is one of the popular techniques used in FER (Akhand, M. A. H., et al. 2021), (Ng, H., et al. ,2015). Transfer learning is a very efficient way at creating models which are (partially or thoroughly) derived from pretrained models and explained in more abstract mathematical terms (Zhuang F. et al., 2019).

Basically, transfer learning is to make use of the parameters of a pretrained model possibly extracting from some of the weights (in convolutional neural networks they may be from some critical layers) or the whole architecture of it or a part of its architecture in solving another problem based on different tasks or target domains. To give a specific discussion on an instance, the authors of (Akhand, M. A. H., et al., 2021) say that the first layers record simple features (such as edges and corners) of an image while the next layers keep complex features (such as textures and shapes) of it. What they claim is, in other words, the deeper layer of a convolutional neural network the more complicated features it carries. They think that the basic features are similar to all images (for FER problems). So, they used the first convolutional layers of a model called VGG-16 and trained it with another fine-tuned dense layer (Akhand, M. A. H., et al., 2021) for their proposed model. The figure below picture captured is their architecture.

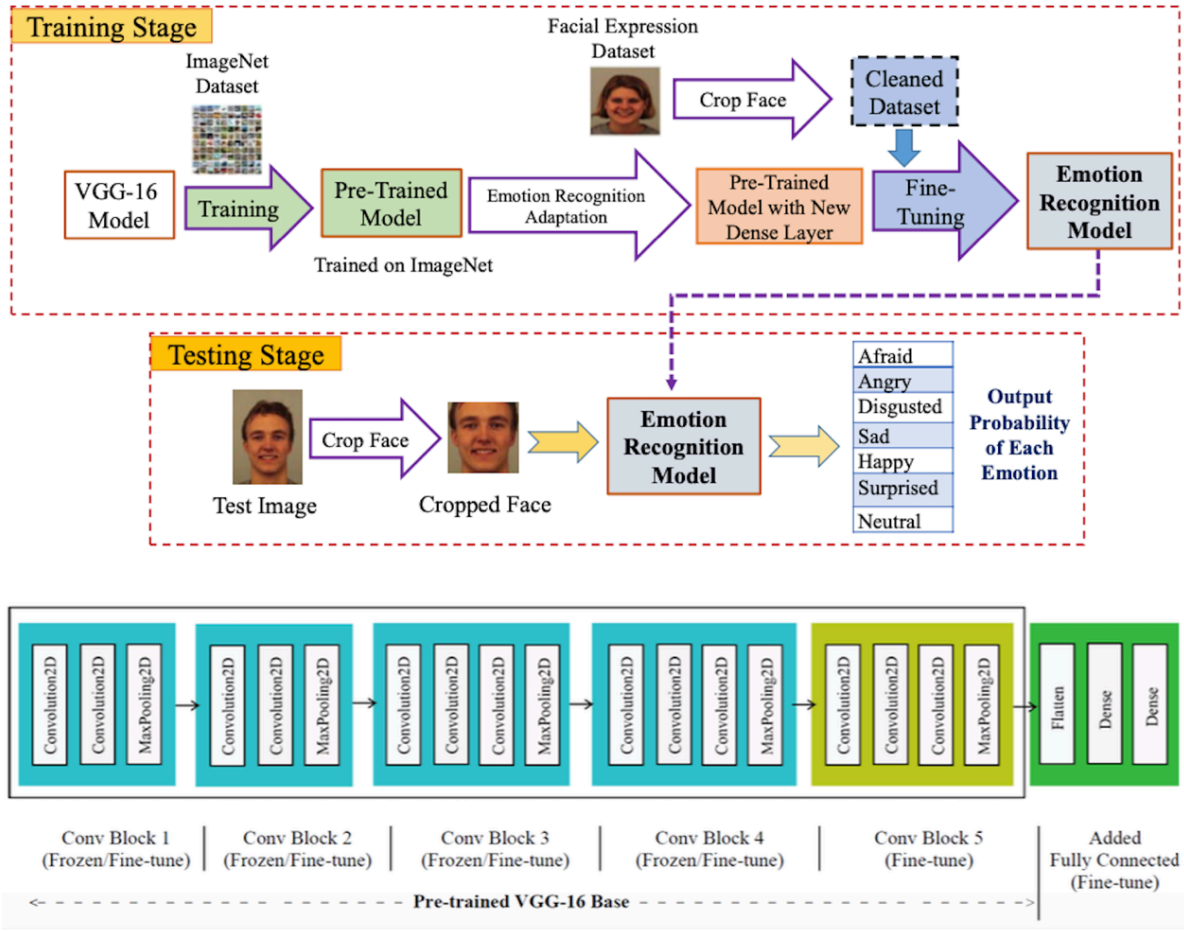

Figure 5: An architecture in which transfer learning is used

2. THE DATA PREPROCESSING AND THE FEATURES EXTRACTED

We trained our model(s) presented in the next section by the data from AffectNet. The dataset of AffectNet is divided into two groups, as mentioned in the previous section, one of which is manually annotated and of 420,299 images. We used the manually annotated dataset. In fact, we made use of a subset of the dataset. The reason for why we could not use the whole data is that the AffectNet has images labelled so that the number of images for each class is unbalanced and we ignored the categories 'none' and 'uncertain' and the features we want to extract restricted a little bit more for technical reasons explained later in this section.

Firstly, we extracted the 68 facial landmarks from the Python 'dlib' library for each image instead of using the facial landmarks provided in the AffectNet due to a comparative experiment between 'dlib' library and the given facial landmarks from AffectNet that 'dlib' seemed more accurate just by manually checking. An example of comparison is below presented.

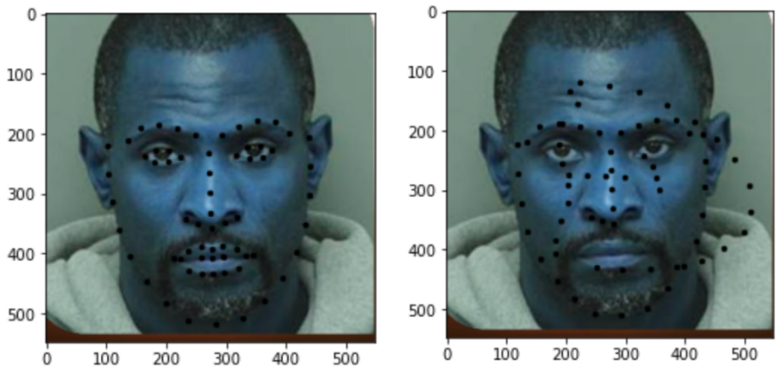

Figure 6: The picture on the left with landmarks from dlib library and the picture on the right with landmarks from the AffectNet.

We wanted to evaluate the angles of triangles that have corners as a triple of landmarks. However, there are, in total, $3\binom{68}{3}$ angles which are excessively much. We assumed that some of the landmarks are not flexible when relatively compared to other landmarks. In other words, when someone changes his/her face, some of the landmarks are thought to be constant, but the

others. For an instance to explain this issue let us have a look at the following landmarks plotted on a face which are supposed to be given by the 'dlib' library.

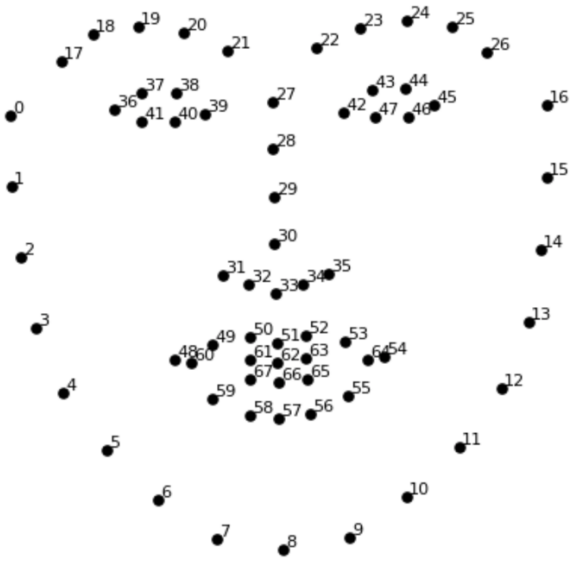

Figure 7: The 68 facial landmarks

In the figure above, we divided the set of all landmarks into two groups one of which is that {0, 1, 2, 7, 8, 9, 14, 15, 16, 27, 28, 29} considered to be constant and the other group of landmarks {17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 36, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59} considered to be active ones. The supposition is that emotions react to these defined landmarks more than the ones which are said to be constant. We obtained one angle by choosing two constant landmarks and one active landmark thinking of them as corners of a triangle. For example, if we choose the numbers 0 and 1 from the list of constant landmarks and the number 17 from the list of active landmarks, we consider the cosine of the angles between the line joining the landmarks 0 and 17 and the line segment joining the landmarks 1 and 17. Cosine values of the angles are calculated by the formula:

$$\cos(\theta) = < x, y > / (\left|\left|x\right|\right|.\left|\left|y\right|\right|),$$

where x is the vector with initial position at one landmark and terminal position at other landmark, similarly for y, $\left|\left|x\right|\right|$ and $\left|\left|y\right|\right|$ are the norms (lengths) of these vectors and θ is the angle between these two vectors.

In addition to angle features we also considered the 5 ratios of the distance between the landmarks 3 and 48 over the distance between the landmarks 3 and 60; the distance between the landmarks 13 and 54 over the distance between the landmarks 13 and 64; the distance between the landmarks 61 and 67 over the distance between the landmarks 50 and 58; the distance between the landmarks 63 and 65 over the distance between the landmarks 52 and 56;

the distance between the landmarks 62 and 66 over the distance between the landmarks 51 and 57.

The number of angle features and ratio features is in total 2385. We concatenated all these angle and ratio features and applied Principle Component Analysis (PCA) with output carrying 95% information of these features. PCA leads to the fact that the number of all of these features reduces to 33 features.

We observed that the number of the images labelled as 'contempt' is 4,250 which is the minimum among the other images (neutral: 75,374; happy: 134,915; sad: 25,959; surprise: 14,590; fear: 6,878; disgust: 4,303; anger: 25,382; contempt: 4,250; none: 33,588; uncertain: 12,145; non-face: 82,915). In order to prevent bias training, we randomly chose 4,250 for each of the other groups of labelled images. However, due to zero-value errors while calculating the cosine values of angles and ratios, we had to eliminate those with this error. The remaining ones without such an error occurred in evaluating angle and ratio features are listed as neutral: 3,543; happy: 3,630; sad: 3,486; surprise: 3,538; fear: 3,458; disgust: 3,588; anger: 3,458; contempt: 3,631; none: 3,524; uncertain: 3,489; non-face: 3,022. Furthermore, we ignored the categories 'none' and 'uncertain'. So, in total, we get 31354 images to train our model. On average, we have 3484 images per class. Similarly, the same reasons led to a decrease in the number of test data. The test data has, in total, 4223 images and the number of images in the test data varies as follows: neutral: 468; happy: 479; sad: 473; surprise: 475; fear: 463; disgust: 479; anger: 471; contempt: 485; non-face: 430 ignoring the cases labeled as 'none' and 'uncertain'.

We resized all the images to 100x100 pixels before using 'dlib' library. In order to get sufficient data we applied augmentation by shifting and rotating to the images randomly. We did not apply augmentation to the whole of the data. Instead, we have chosen any one of them with a 1/3 probability and applied shifting with ½ probability and rotation with ½ probability. The shifting and the rotation are also applied randomly within the procedure. We chose a random integer between -15 and 15 (both including) for the amount of shifting (in pixels) horizontally and another random integer between -15 and 15 (both including) for the amount of shifting (in pixels) vertically. Similarly, we chose a random integer between -30 and 30 for the amount of rotation in degrees (counterclockwise if the number is positive and clockwise if the number is negative). The resultant number of the images per class after the augmentation is as follows; neutral: 4,704; happy: 4827; sad: 4669; surprise: 4,717; fear: 4,602; disgust: 4,798; anger: 4,574; contempt: 4,873; non-face: 4,022.

3. MODELS CREATED TO BE TRAINED

We created numerous convolutional neural network models to get best results. We considered five of them which were comparatively successful. Each of these four models gave accuracies approximately (and very close to) 40% on the validation test. In each of the following sections we will present the details of these models. Lastly, another additional model will be provided. The last model is a combined version of these four models and leads to an accuracy of 43% on the validation set.

3.1. Model-1

The model-1 is figured below

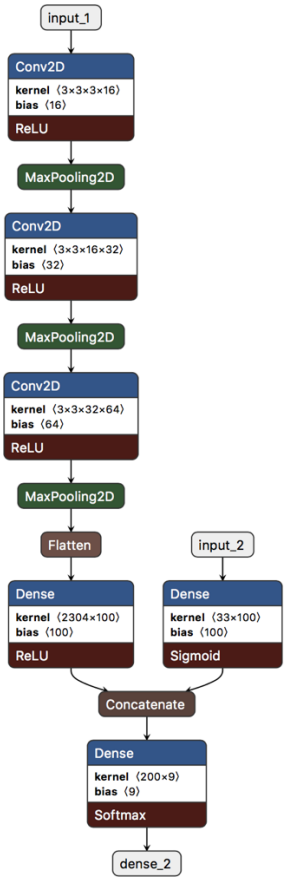

Figure 8: The Model-1

The next table shows the summary of Model-1.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 100, 100, 3)	0	
conv2d (Conv2D)	(None, 98, 98, 16)	448	input_1[0][0]
max_pooling2d (MaxPooling2D)	(None, 32, 32, 16)	0	conv2d[0][0]
conv2d_1 (Conv2D)	(None, 30, 30, 32)	4640	max_pooling2d[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 15, 15, 32)	0	conv2d_1[0][0]
conv2d_2 (Conv2D)	(None, 13, 13, 64)	18496	max_pooling2d_1[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 64)	0	conv2d_2[0][0]
flatten (Flatten)	(None, 2304)	0	max_pooling2d_2[0][0]
input_2 (InputLayer)	[(None, 33)]	0	
dense (Dense)	(None, 100)	230500	flatten[0][0]
dense_1 (Dense)	(None, 100)	3400	input_2[0][0]
concatenate (Concatenate)	(None, 200)	0	dense[0][0] dense_1[0][0]
dense_2 (Dense)	(None, 9)	1809	concatenate[0][0]

Total params: 259,293
Trainable params: 259,293
Non-trainable params: 0

Figure 9: Summary of the Model-1

The image file, which is the input-1 shown in the figure above, is given as input after it is converted to a 'numpy array' which is derived from the Python 'numpy' library. In fact, all the five models mentioned take the image in this object type and process it. The first layer is a convolution layer mapping to 16 neurons with the activation function 'Relu' defined as $Relu(x) = \max(x, 0)$ for any real number x. Then, a max pooling layer follows. The next two layers follow similarly mapping to 32 and 64 neurons, respectively. All of the max pooling layers in the Model-1 evaluate maximum value on 3x3 pixels. Input-2 in the figure above corresponds to the feature vector obtained lastly from PCA and has 33 components. This handcrafted feature vector is sent to a dense layer with 100 neurons with the activation function 'sigmoid' and the output of convolution layers is sent to a dense layer of 100 neurons. Afterwards, they are concatenated and sent to a dense layer of 9 neurons with the activation function 'softmax'. The sigmoid function is defined as follows.

$$sigmoid(x) = \frac{1}{1 + e^{-x}}$$

for any real number x and the softmax function is defined to be

$$softmax(x) = < \frac{e^{x_1}}{\sum_{i=1}^{n} e^{x_i}}, \frac{e^{x_2}}{\sum_{i=1}^{n} e^{x_i}}, \dots, \frac{e^{x_n}}{\sum_{i=1}^{n} e^{x_i}} >,$$

where $x = < x_1, x_2, \dots, x_n >$ is a vector. The components of $softmax(x)$ are the outputs of the layer which the layer is activated with. Observe that the components are added up to 1 and each component is nonnegative. This is interpreted as a probability value for the category. In

particular, the last layer in the Model-1 outputs a vector having 9 nonnegative components with a sum 1, the value of n in the formula of softmax function is set to 9. During the backpropagation of the process of training these probabilities are to be as close as possible to the real values which constitute actually a vector having 0 at all of its components except the one corresponding to the category of the input image, it is 1.

3.2. Model-2

The Model-2 is presented in the figure below.

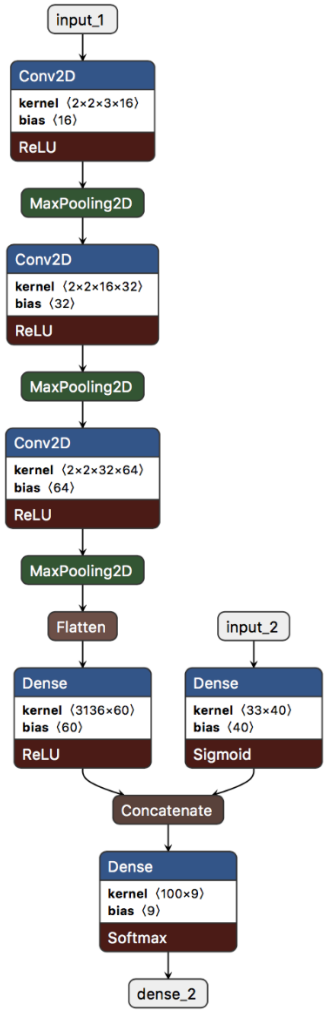

Figure 10: Model-2

The Model-2 differs from the Model-1 at the number of neurons in two layers. One is that the number of neurons in the first dense layer which the input-1 is sent (through the convolution layers) to has 60 neurons and the number of neurons in the first dense layer which the input-2 is sent (through the convolution layers) to has 40 neurons. As in the Model-1, then they are concatenated to be sent to the last dense layer with the softmax activation function.

Another distinction is that all of the convolution layers in the Model-2 apply convolution by 2x2 matrices. Below is the summary of the Model-2.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 100, 100, 3)	0	
conv2d (Conv2D)	(None, 99, 99, 16)	208	input_1[0][0]
max_pooling2d (MaxPooling2D)	(None, 33, 33, 16)	0	conv2d[0][0]
conv2d_1 (Conv2D)	(None, 32, 32, 32)	2080	max_pooling2d[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 16, 16, 32)	0	conv2d_1[0][0]
conv2d_2 (Conv2D)	(None, 15, 15, 64)	8256	max_pooling2d_1[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 7, 7, 64)	0	conv2d_2[0][0]
flatten (Flatten)	(None, 3136)	0	max_pooling2d_2[0][0]
input_2 (InputLayer)	[(None, 33)]	0	
dense (Dense)	(None, 60)	188220	flatten[0][0]
dense_1 (Dense)	(None, 40)	1360	input_2[0][0]
concatenate (Concatenate)	(None, 100)	0	dense[0][0] dense_1[0][0]
dense_2 (Dense)	(None, 9)	909	concatenate[0][0]

Total params: 201,033
Trainable params: 201,033
Non-trainable params: 0

Figure 11: Summary of the Model-2

3.3. Model-3

The Model-3 is presented in the figure below.

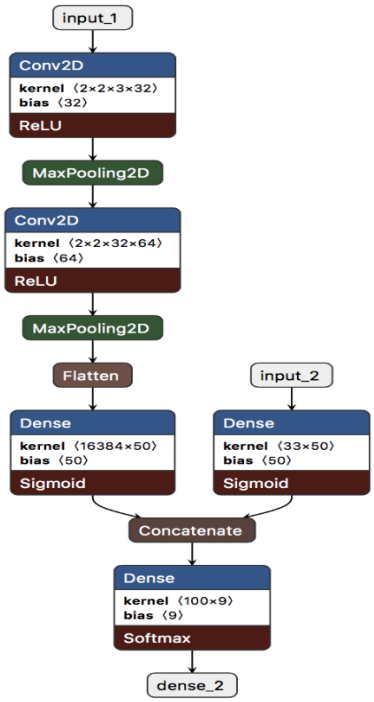

Figure 12: Model-3

The Model-3 has two convolution layers applyied by 2x2 matrices and the first convolution layer is followed by a max pooling over 3x3 pixels and the second convolution layer is followed by a max pooling over 2x2 pixels. Furthermore, the dense layer which takes the output of the last convolution and the dense layer which takes the handcrafted features has 50 nodes.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 100, 100, 3)	0	
conv2d (Conv2D)	(None, 99, 99, 32)	416	input_1[0][0]
max_pooling2d (MaxPooling2D)	(None, 33, 33, 32)	0	conv2d[0][0]
conv2d_1 (Conv2D)	(None, 32, 32, 64)	8256	max_pooling2d[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 16, 16, 64)	0	conv2d_1[0][0]
flatten (Flatten)	(None, 16384)	0	max_pooling2d_1[0][0]
input_2 (InputLayer)	[(None, 33)]	0	
dense (Dense)	(None, 50)	819250	flatten[0][0]
dense_1 (Dense)	(None, 50)	1700	input_2[0][0]
concatenate (Concatenate)	(None, 100)	0	dense[0][0] dense_1[0][0]
dense_2 (Dense)	(None, 9)	909	concatenate[0][0]

Total params: 830,531
Trainable params: 830,531
Non-trainable params: 0

Figure 13: Summary of Model-3

3.4. Model-4

The Model-4 is presented in the figure below.

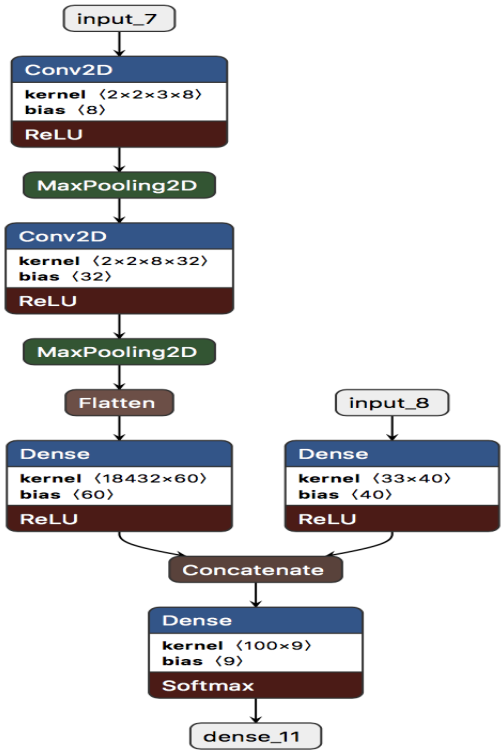

Figure 14: Model-4

The Model-4 differs from the Model-3 in that it has 8 and 32 nodes in the convolution layers and the dense layers which take the outputs from the convolutional network and the fully-connected layer which takes the handcrafted features has 60 and 40 nodes, respectively. Note that you may ignore the words ‘input_7’ and ‘input_8’ in the figure above, they are input_1 and input_2, respectively and they correspond to the same input values (images and features) as in the previous models. Below is the summary of the Model-4.

Layer (type)	Output Shape	Param #	Connected to
input_7 (InputLayer)	[(None, 100, 100, 3)	0	
conv2d_8 (Conv2D)	(None, 99, 99, 8)	104	input_7[0][0]
max_pooling2d_8 (MaxPooling2D)	(None, 49, 49, 8)	0	conv2d_8[0][0]
conv2d_9 (Conv2D)	(None, 48, 48, 32)	1056	max_pooling2d_8[0][0]
max_pooling2d_9 (MaxPooling2D)	(None, 24, 24, 32)	0	conv2d_9[0][0]
flatten_4 (Flatten)	(None, 18432)	0	max_pooling2d_9[0][0]
input_8 (InputLayer)	[(None, 33)]	0	
dense_9 (Dense)	(None, 60)	1105980	flatten_4[0][0]
dense_10 (Dense)	(None, 40)	1360	input_8[0][0]
concatenate_4 (Concatenate)	(None, 100)	0	dense_9[0][0] dense_10[0][0]
dense_11 (Dense)	(None, 9)	909	concatenate_4[0][0]

Total params: 1,109,409
Trainable params: 1,109,409
Non-trainable params: 0

Figure 15: The summary of the Model-4

3.5. The Combined Model

Suppose that we have an input x and say that x has two components x_1 and x_2 so that x_1 corresponds to the array of image and x_2 corresponds to the feature. We know that for each x there exist 4 outputs from the previous models (Model-1, Model-2, Model-3, Model-4). These outputs are not the categories, but the probability distributions obtained from the models. Let us put m_1, m_2, m_3, and m_4 for a given input x. So, we may think that we created 4 more features. We can consider that these features m_1, m_2, m_3, and m_4 are actually vectors which have coordinates carrying probabilities derived from the corresponding models. For instance, if $m_4 = (0.1, 0.1, 0.2, 0.3, 0.05, 0.03, 0.07, 0.05, 0.1)$, that means the Model-4 gives a probability distribution of these 9 numbers for the given input x. According to the argmax principle the Model-4 implies that x is labeled as 'surprise' (because the maximum probability 0.3 indicates the 4th component which corresponds to the probability of being labeled as 'surprise') though it may not actually be of that label. Although they may be not well distributed probabilities, as we will see that accuracies are low, they may give clues if we can use all of them at the same time. Hence, we may use these probability distributions as features for another model, say *the combined model*. The following figure presents this, indeed, simple model.

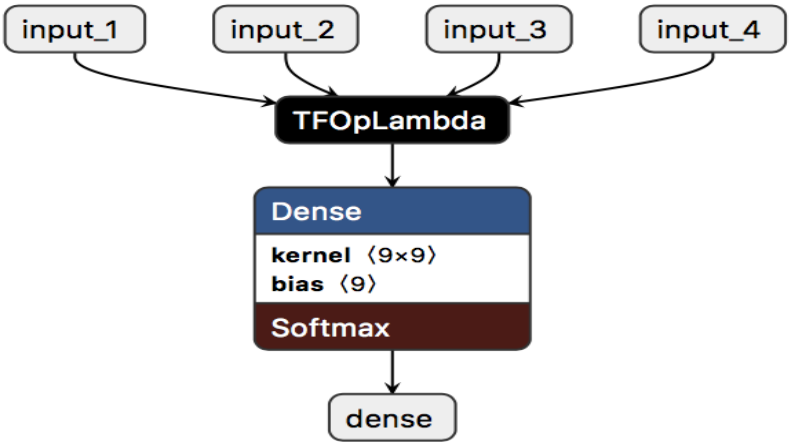

Figure 16: The combined model

The input_1, the input_2, the input_3, and the input_4 in the figure above correspond to the probability vectors derived from the previous models (Model-1, Model-2, Model-3, Model-4, respectively). Firstly, these input vectors are taken to be summed to be another vector of dimension 9 (the figure shows this step as TFOpLambda) and then sent to a dense layer with the softmax activation function.

5. TRAINING AND ACCURACIES

We used the programming language Python 3 and the libraries Keras and Tensorflow for modelling and training. The number of epochs for each of the first four models is 10. This number is 20 for the combined model. Optimization is customized by RMSprop with learning rate equal to 0.001. The optimization of the loss function is tuned for over batches of 100 inputs (the images and the features) for each of the five models. While training the models, the training data is split into two sets one of which is the validation set and the other one is the actual training data that the model learns from. The splitting ratio of validation data is 0.2 except that we split the training set with a ratio of 0.1 for the combined model. We used this method in order to see how well the training is processed. The accuracies of the models on the training dataset and the test dataset are given in the following table.

Table 1: Accuracies of the models

	Accuracy on the training dataset	Accuracy on the test dataset
Model-1	0.57	0.41
Model-2	0.49	0.41
Model-3	0.63	0.41
Model-4	0.64	0.38
The combined model	0.69	0.43

The following figures give an idea about how the training went through. The graphs on the left columns are for the training accuracies and the validation accuracies and the ones on the right columns are for the loss function values after each epoch of the model through the whole training dataset and the validation dataset.

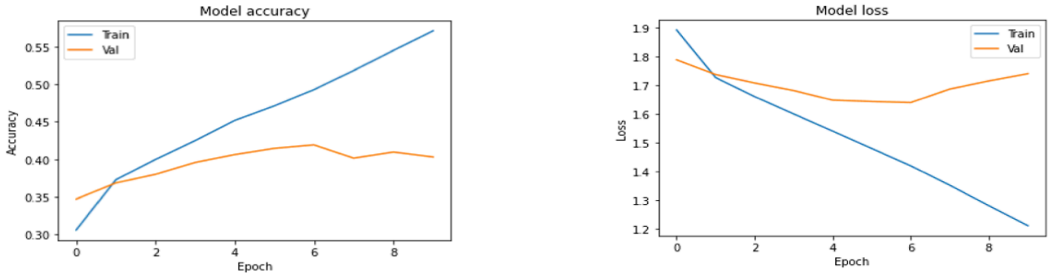

Figure 17: Accuracy and loss function graphs for Model-1

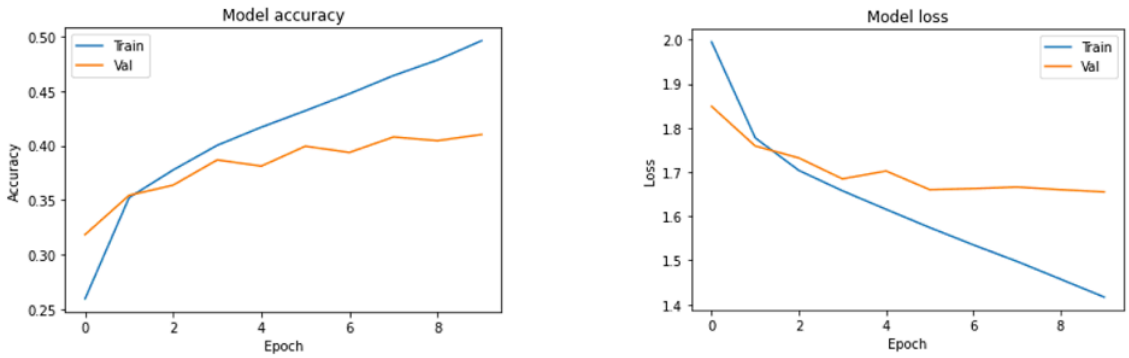

Figure 18: Accuracy and loss function graph for Model-2

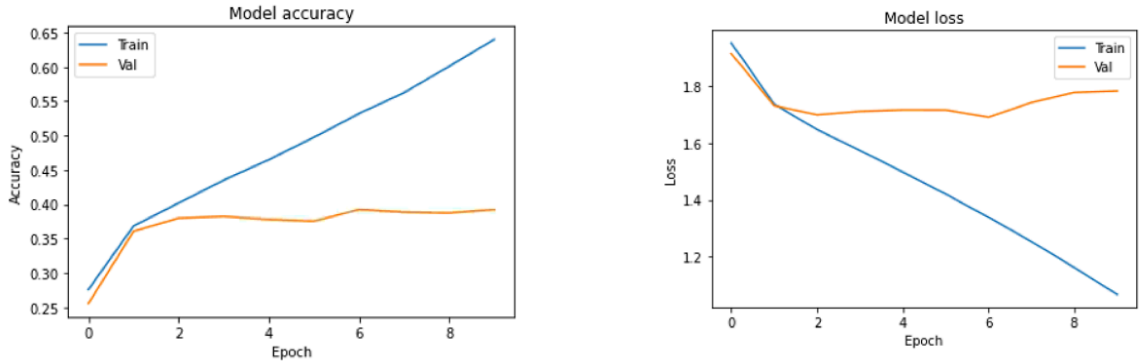

Figure 19: Accuracy and loss function for Model-3

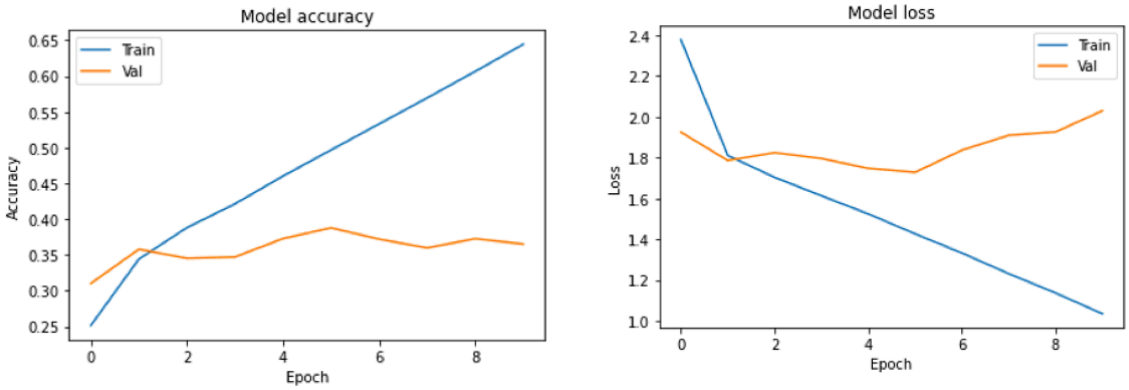

Figure 20: Accuracy and loss function for Model-4

5. CONCLUSION

The main intuition to develop a solution to the facial emotion recognition problem was due to the paper (G. Viswanatha Reddy et al, 2020). They used both hand-crafted features and images to train their model. We changed the idea of taking the distances between the facial landmarks into account, instead, we considered the ratios as we calculated the cosine values of the angles and some specific ratios of distances. The results were not that much successful as claimed in (G. Viswanatha Reddy et al, 2020). There are several reasons for such a difference. One of them is that they used the pretrained architectures (XceptionNet) for extracting features and a further advanced SVM (support vector machine) classifiers. Another reason is that they made use of more images than we used for this project.

The paper (M. A. Jalal et al, 2019) presents very nice results though that the dataset AffectNet has very wild images. What I have experienced from this project is that the number of nodes in dense layers does not change the results unless they are tuned extremely. Furthermore, the paper (M. A. Jalal et al, 2019) gives the intuition that the self-attention mechanisms are important for image classifications through convolutional neural networks. However, what I believe is that the hand-crafted feature extracting is also important so that we may be impressed by their unique effects in the paper (Dunau P. et al, 2019).

APPENDIX A

THE MODULE BELOW CREATES COLUMNS FOR FEATURES

```
import pandas as pd
import numpy as np
import csv

df_training=pd.read_csv('training.csv')
df_validation=pd.read_csv('validation.csv')

print('training.csv and validation.csv are loaded. lengths are:')
print(len(df_training))
print(len(df_validation))

# we drop columns got from the data to replace dlib information
df_training=df_training.drop(['facial_landmarks','face_x','face_y','face_width','face_height'],a
xis=1)
df_validation=df_validation.drop(['facial_landmarks','face_x','face_y','face_width','face_heigh
t'],axis=1)

# create 16 length columns
for i in range(67):
    '''create cols for
    lengths 0-1, 1-2,...,66-67 '''
    s='l_{}_{}'.format(i,i+1)

    df_training[s]=np.nan
    df_validation[s]=np.nan

# adding ratio information to training data.

df_training['r_3_48_3_60']=np.nan
df_training['r_13_54_13_64']=np.nan
df_training['r_61_67_50_58']=np.nan
df_training['r_63_65_52_56']=np.nan
df_training['r_62_66_51_57']=np.nan

# adding ratio information to validation data.

df_validation['r_3_48_3_60']=np.nan
df_validation['r_13_54_13_64']=np.nan
df_validation['r_61_67_50_58']=np.nan
df_validation['r_63_65_52_56']=np.nan
df_validation['r_62_66_51_57']=np.nan
```

```
column_names=list(df_training.columns)

print('now number of columns is {}'.format(len(column_names)))

# we add angles crosswise chosen between two groups of landmarks

constant_landmarks=[0,1,2,14,15,16,27,28,29,7,8,9]
active_landmarks=[17,18,19,20,21,22,23,24,25,26,36,37,38,39,40,41,42,43,44,45,46,47,48,49
,50,51,52,53,54,55,56,57,58,59]

print('constant landmark count is {}'.format(len(constant_landmarks)))
print('active landmarks count is {}'.format(len(active_landmarks)))

from itertools import combinations

comb = combinations(constant_landmarks, 2)

comb_list=list(comb)

for k in range(len(comb_list)):
    for j in range(len(active_landmarks)):
        t='a_{}_{}_{}'.format(active_landmarks[j],comb_list[k][0],comb_list[k][1])
        column_names+=[t]
        #df_training[t]=np.nan
        #df_validation[t]=np.nan
        #print('the column '+t+' created for df_training and df_validation.')

column_names_file_name='column_names.csv'

# writing to csv file
with open(column_names_file_name, 'w') as csvfile:
    # creating a csv writer object
    csvwriter = csv.writer(csvfile)

    # writing the fields
    csvwriter.writerow(column_names)
```

THE FOLLOWING MODULE CREATES FEATURES USING DLIB LIBRARY WHILE DETECTING POSSIBLE ERRORS FOR TRAINING DATASET

```
import os
import pandas as pd
import numpy as np
import math

import dlib
import cv2
from matplotlib import pyplot as plt
import csv
```

```
def angle_(points):
    '''points=[a,b,c]> gives the angle between
    the vectors ab and bc. '''
    v1_x=points[0][0]-points[1][0]
    v1_y=points[0][1]-points[1][1]
    v2_x=points[2][0]-points[1][0]
    v2_y=points[2][1]-points[1][1]
    r1=v1_x*v2_x+v1_y*v2_y
    r2=math.sqrt(v1_x**2+v1_y**2)*math.sqrt(v2_x**2+v2_y**2)

    if r2==0:
        return 'error'
    elif r1/r2<=1 and -1<=r1/r2:
        return math.acos(r1/r2)
    elif r1>r2:
        return 0
    else:
        return -math.pi

def length_(points):
    '''points=[a,b]> a ile b noktlari arasindaki
    uzaklik.'''
    d1=points[0][0]-points[1][0]
    d2=points[0][1]-points[1][1]
    d=math.sqrt(d1**2+d2**2)
    return d

def ratio(points):
    d1=length_([points[0],points[1]])
    d2=length_([points[2],points(N. Rodrigez-Diaz et al, 2021)])
    if d2==0:
        return 'error'
    else:
        return d1/d2

df_training=pd.read_csv('training.csv')
df_validation=pd.read_csv('validation.csv')


detector=dlib.get_frontal_face_detector()
predictor=dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")

column_names_df=pd.read_csv('column_names.csv')
column_names=list(column_names_df.columns)

row_indices_selected=np.load('row_indices_selected_from_training_data.npy')
row_indices_selected=row_indices_selected.tolist()
row_indices_selected.sort()

print('number of all rows is {}'.format(len(row_indices_selected)))
```

```
filename = "new_training_data_dlib_landmarks_and_features_v3.csv"



full_error_list=[]
angle_errors_list=[]
ratio_errors_list=[]

# writing to csv file
with open(filename, 'w') as csvfile:
    # creating a csv writer object
    csvwriter = csv.writer(csvfile)

    # writing the fields
    csvwriter.writerow(column_names)

    # writing the data rows
    for i in row_indices_selected:
        print('i={}'.format(i))
        try:
            s="/Volumes/macpart/capstone_project_files/Source_4/Manually_Annotated_Images"
            number_of_the_folder=df_training.iloc[i]['subDirectory_filePath'].split('/')[0]
            name_of_the_image_file=df_training.iloc[i]['subDirectory_filePath'].split('/')[1]
            os.chdir(s+'/'+number_of_the_folder)
            img=cv2.imread(name_of_the_image_file,1)
            img=cv2.resize(img,(100,100))
            face=detector(img)

            if len(face)!=1:
                continue
            if img.shape[2]!=3:
                continue
            for f in face:
                x_1=f.left()
            landmarks=predictor(img,f)
            landmarks_list=[]
            for q in range(68):
                landmarks_list.append([landmarks.part(q).x,landmarks.part(q).y])
            one_row=[]
            one_row.append(df_training.iloc[i]['subDirectory_filePath'])
            one_row.append(df_training.iloc[i]['expression'])
            one_row.append(df_training.iloc[i]['valence'])
            one_row.append(df_training.iloc[i]['arousal'])
            for j in range(len(column_names)-4):
                clm=column_names[j+4].split('_')
                if clm[0]=='l':
                    lm1=int(clm[1])
                    lm2=int(clm[2])
                    p1=landmarks_list[lm1]
```

```
                p2=landmarks_list[lm2]
                dist=length_([p1,p2])
                one_row.append(dist)
            elif clm[0]=='a':
                lm1=int(clm[1])
                lm2=int(clm[2])
                lm3=int(clm(N. Rodrigez-Diaz et al, 2021))
                p1=landmarks_list[lm1]
                p2=landmarks_list[lm2]
                p3=landmarks_list[lm3]
                ang=angle_([p1,p2,p3])
                if ang=='error':
                    angle_errors_list+=[(i,column_names[j+4])]
                one_row.append(ang)
            else:
                lm1=int(clm[1])
                lm2=int(clm[2])
                lm3=int(clm(N. Rodrigez-Diaz et al, 2021))
                lm4=int(clm([4]  Z. Kowalczk et al, 2019))
                p1=landmarks_list[lm1]
                p2=landmarks_list[lm2]
                p3=landmarks_list[lm3]
                p4=landmarks_list[lm4]
                rat=ratio([p1,p2,p3,p4])
                if rat=='error':
                    ratio_errors_list+=[(i,column_names[j+4])]
                one_row.append(rat)
        csvwriter.writerow(one_row)
    except:
        full_error_list+=[i]

print(full_error_list)
print('serious error occurs in {} rows'.format(len(full_error_list)))

print(angle_errors_list)
print('angle error occurs in {} rows'.format(len(angle_errors_list)))

print(ratio_errors_list)
print('ratio error occurs in {} rows'.format(len(ratio_errors_list)))
```

THE FOLLOWING MODULE CREATES FEATURES USING DLIB LIBRARY WHILE DETECTING POSSIBLE ERRORS FOR THE TEST DATASET

```
import os
import pandas as pd
import numpy as np
import math

import dlib
import cv2
import csv

def angle_(points):
    '''points=[a,b,c]> gives the angle between
    the vectors ba and bc.'''
    v1_x=points[0][0]-points[1][0]
    v1_y=points[0][1]-points[1][1]
    v2_x=points[2][0]-points[1][0]
    v2_y=points[2][1]-points[1][1]
    r1=v1_x*v2_x+v1_y*v2_y
    r2=math.sqrt(v1_x**2+v1_y**2)*math.sqrt(v2_x**2+v2_y**2)

    if r2==0:
        return 'error'
    elif r1/r2<=1 and -1<=r1/r2:
        return math.acos(r1/r2)
    elif r1>r2:
        return 0
    else:
        return -math.pi

def length_(points):
    '''points=[a,b]> gives the distance between two landmarks.'''
    d1=points[0][0]-points[1][0]
    d2=points[0][1]-points[1][1]
    d=math.sqrt(d1**2+d2**2)
    return d

def ratio(points):
    d1=length_([points[0],points[1]])
    d2=length_([points[2],points(N. Rodrigez-Diaz et al, 2021)])
    if d2==0:
        return 'error'
    else:
```

```
        return d1/d2

#df_training=pd.read_csv('training.csv')
df_validation=pd.read_csv('validation.csv')


detector=dlib.get_frontal_face_detector()
predictor=dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")

column_names_df=pd.read_csv('column_names.csv')
column_names=list(column_names_df.columns)



print('number of all rows is {}'.format(len(df_validation)))


filename = "new_test_data_dlib_landmarks_and_features_v1.csv"



full_error_list=[]
angle_errors_list=[]
ratio_errors_list=[]

# writing to csv file
with open(filename, 'w') as csvfile:
    # creating a csv writer object
    csvwriter = csv.writer(csvfile)

    # writing the fields
    csvwriter.writerow(column_names)

    # writing the data rows
    for i in range(len(df_validation)):
        print('i={}'.format(i))
        try:
            s="/Volumes/macpart/capstone_project_files/Source_4/Manually_Annotated_Images"
            number_of_the_folder=df_validation.iloc[i]['subDirectory_filePath'].split('/')[0]
            name_of_the_image_file=df_validation.iloc[i]['subDirectory_filePath'].split('/')[1]
            os.chdir(s+'/'+number_of_the_folder)
            img=cv2.imread(name_of_the_image_file,1)
            img=cv2.resize(img,(100,100))
            face=detector(img)

            if len(face)!=1:
                continue
            #if img.shape[0]!=img.shape[1]:
            #    nonsquare_pics+=[i]
            if img.shape[2]!=3:
```

```
            continue
        for f in face:
            x_1=f.left()
        landmarks=predictor(img,f)
        landmarks_list=[]
        for q in range(68):
            landmarks_list.append([landmarks.part(q).x,landmarks.part(q).y])
        one_row=[]
        one_row.append(df_validation.iloc[i]['subDirectory_filePath'])
        one_row.append(df_validation.iloc[i]['expression'])
        one_row.append(df_validation.iloc[i]['valence'])
        one_row.append(df_validation.iloc[i]['arousal'])
        for j in range(len(column_names)-4):
            clm=column_names[j+4].split('_')
            if clm[0]=='l':
                lm1=int(clm[1])
                lm2=int(clm[2])
                p1=landmarks_list[lm1]
                p2=landmarks_list[lm2]
                dist=length_([p1,p2])
                one_row.append(dist)
            elif clm[0]=='a':
                lm1=int(clm[1])
                lm2=int(clm[2])
                lm3=int(clm(N. Rodrigez-Diaz et al, 2021))
                p1=landmarks_list[lm1]
                p2=landmarks_list[lm2]
                p3=landmarks_list[lm3]
                ang=angle_([p1,p2,p3])
                if ang=='error':
                    angle_errors_list+=[(i,column_names[j+4])]
                one_row.append(ang)
            else:
                lm1=int(clm[1])
                lm2=int(clm[2])
                lm3=int(clm(N. Rodrigez-Diaz et al, 2021))
                lm4=int(clm([4]  Z. Kowalczk et al, 2019))
                p1=landmarks_list[lm1]
                p2=landmarks_list[lm2]
                p3=landmarks_list[lm3]
                p4=landmarks_list[lm4]
                rat=ratio([p1,p2,p3,p4])
                if rat=='error':
                    ratio_errors_list+=[(i,column_names[j+4])]
                one_row.append(rat)
        csvwriter.writerow(one_row)
    except:
        full_error_list+=[i]
```

```
print(full_error_list)
print('serious error occurs in {} rows'.format(len(full_error_list)))

print(angle_errors_list)
print('angle error occurs in {} rows'.format(len(angle_errors_list)))

print(ratio_errors_list)
print('ratio error occurs in {} rows'.format(len(ratio_errors_list)))
```

CLEANING OF THE TRAINING DATA BEFORE APPLYING PCA

```
import pandas as pd
import numpy as np
df=pd.read_csv('new_training_data_dlib_landmarks_and_features_v3.csv')

print(len(df))

col_names=df.columns.tolist()
err_list=[]


for col in col_names:
    print(col)
    err_list+=df[df[col]=='error'].index.values.tolist()

print(len(err_list))
e=set(err_list)
err_list_uniq=list(e)
print(len(err_list_uniq))
print(max(err_list_uniq))


indicies_to_drop=[df.index[j] for j in err_list_uniq]

df_updated=df.drop(indicies_to_drop)
```

```
df_updated.reset_index(inplace=False)

df_updated.to_csv('new_training_data_dlib_landmarks_and_features_v4.csv')
```

CLEANING OF THE TEST DATA BEFORE APPLYING PCA

```
import pandas as pd
import numpy as np
df=pd.read_csv('new_test_data_dlib_landmarks_and_features_v1.csv')


col_names=df.columns.tolist()
err_list=[]


for col in col_names:
    print(col)
    err_list+=df[df[col]=='error'].index.values.tolist()

print('err_list is: \n')
print(err_list)
print(len(err_list))
e=set(err_list)
err_list_uniq=list(e)
print(len(err_list_uniq))
print(max(err_list_uniq))


indicies_to_drop=[df.index[j] for j in err_list_uniq]

df_updated=df.drop(indicies_to_drop)

print(len(df_updated))

print(df_updated.head(10))

df_updated.reset_index(inplace=False)

df_updated.to_csv('new_test_data_dlib_landmarks_and_features_v2.csv', index=False)
```

SAVING (TRAINING AND TEST) IMAGES AS NUMPY ARRAYS

```
import os
import pandas as pd
import numpy as np
import cv2


print(os.getcwd())
```

```
#df_training=pd.read_csv('new_training_data_dlib_landmarks_and_features_v4.csv')
#df_test=pd.read_csv('new_test_data_dlib_landmarks_and_features_v2.csv')
print('train data are loaded.')


s="/Volumes/macpart/capstone_project_files/Source_4/Manually_Annotated_Images"


df_image=[]
not_RGB=[]
not_3_channel=[]
for i in range(len(df_test)):
    print('i={}'.format(i))
    number_of_the_folder=df.iloc[i]['subDirectory_filePath'].split('/')[0]
    name_of_the_image_file=df.iloc[i]['subDirectory_filePath'].split('/')[1]
    os.chdir(s+'/'+number_of_the_folder)
    #img=Image.open(name_of_the_image_file)
    img=cv2.imread(name_of_the_image_file,1)
    img=cv2.resize(img,(100,100))
    df_image.append(img)


os.chdir('/Users/ilkerarslan/Desktop/Capstone_venv_2')
df_test_image=np.asarray(df_test_image)

#np.save('test_images_as_arrays.npy',df_image)
#np.save('training_images_as_arrays.npy')
```

THE MODULE APPLYIES PCA

```
import os
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import math

drive_directory='/Volumes/macpart/capstone_project_files/Capstone_venv_2'
original_directory='/Users/ilkerarslan/Desktop/Capstone_venv_2'
os.chdir(drive_directory)

df_train=pd.read_csv('new_training_data_dlib_landmarks_and_features_v4.csv')
print(df_train.head(3))
```

```
z=df_train.columns.tolist()[0]
print(df_train.columns.tolist()[0])
df_train=df_train.drop([z],axis=1)
print(df_train.columns.tolist()[0])
print(df_train.head(3))

df_test=pd.read_csv('new_test_data_dlib_landmarks_and_features_v2.csv')
print(df_test.head(3))

df_x_train=df_train.drop(['subDirectory_filePath','valence','arousal','expression'],axis=1)
#df_y_train=df_train['expression']

df_x_test=df_test.drop(['subDirectory_filePath','valence','arousal','expression'],axis=1)
#df_y_test=df_test['expression']

c=df_x_train.columns.tolist()
for n in c:
    if n[0]=='l':
        print(n)
        df_x_train=df_x_train.drop([n],axis=1)
        df_x_test=df_x_test.drop([n],axis=1)

    elif n[0]=='a':
        df_x_train[n]=df_x_train[n].apply(lambda x: math.cos(x))
        df_x_test[n]=df_x_test[n].apply(lambda x: math.cos(x))

print('before PCA')
print(df_train.shape)


scaler = StandardScaler()

scaler.fit(df_x_train)

df_x_train=scaler.transform(df_x_train)
df_x_test=scaler.transform(df_x_test)


# Make an instance of the Model
pca = PCA(.95)

pca.fit(df_x_train)

df_x_train=pca.transform(df_x_train)
df_x_test=pca.transform(df_x_test)

np.save('x_train_angle_and_ratio_features_after_PCA_95.npy',df_x_train)
np.save('x_test_angle_and_ratio_features_after_PCA_95.npy.npy',df_x_test)
```

THE MODULE BELOW APPLYIES DATA AUGMENTATION FOR THE TRAINING DATASET

```
import imutils
import cv2
from imutils.convenience import translate
from matplotlib import pyplot as plt
import random
import numpy as np
from sklearn.utils import shuffle


img=cv2.imread('ilker_pic.png',3)

def random_augmentation(img):
    t_list=[-15+i for i in range(31)]
    r_list=[-30+j for j in range(61)]
    x=random.choice([0,1,2])
    if x==2:
        a=random.choice([0,1])
        if a==0:
            tx=random.choice(t_list)
            ty=random.choice(t_list)
            img_translated=imutils.translate(img,tx,ty)
            return img_translated
        else:
            r=random.choice(r_list)
            img_rotated=imutils.rotate(img,r)
            return img_rotated
    else:
        return 'nothing'

training_augmented_images=[]
training_augmented_angle_ratio_features=[]
training_augmented_length_features=[]
training_augmented_eyes=[]
training_augmented_mouth=[]
y_augmented_train=[]

original_training_images=np.load('training_images_as_arrays.npy')
original_x_train_angle_and_ratio_features=np.load('x_train_angle_and_ratio_features_after_
PCA_95.npy')
original_x_train_length_features=np.load('x_train_length_features.npy')
original_x_train_eyes=np.load('x_train_eyes.npy')
original_x_train_mouth=np.load('x_train_mouth.npy')
original_y_train=np.load('y_train.npy')



for i in range(len(original_training_images)):
```

```
    image_array=original_training_images[i]
    im=random_augmentation(image_array)
    if type(im)==str:
        training_augmented_images.append(image_array)

training_augmented_angle_ratio_features.append(original_x_train_angle_and_ratio_features[
i])

        y_augmented_train.append(original_y_train[i])
    else:
        training_augmented_images.append(image_array)

training_augmented_angle_ratio_features.append(original_x_train_angle_and_ratio_features[
i])

        y_augmented_train.append(original_y_train[i])

        training_augmented_images.append(im)

training_augmented_angle_ratio_features.append(original_x_train_angle_and_ratio_features[
i])
        y_augmented_train.append(original_y_train[i])

training_augmented_images=np.asarray(training_augmented_images)
training_augmented_angle_ratio_features=np.asarray(training_augmented_angle_ratio_featur
es)
y_augmented_train=np.asarray(y_augmented_train)


print(len(training_augmented_images))

images, angle_ratio, length, eyes, mouth, y_values
=shuffle(training_augmented_images,training_augmented_angle_ratio_features,training_aug
mented_length_features,training_augmented_eyes,training_augmented_mouth,y_augmented_
train)

np.save('augmented_training_images.npy',images)
np.save('augmented_training_angle_ratio_features.npy',angle_ratio)
np.save('augmented_y_train.npy',y_values)
```

MODEL-1

```
import numpy as np
import pandas as pd
from pandas.core.accessor import DirNamesMixin
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras

from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Dropout, Input,
concatenate
from tensorflow.keras import layers
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.optimizers import RMSprop

from sklearn.datasets import make_circles
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import confusion_matrix
from tensorflow.python.keras.engine.training import Model

from sklearn.metrics import classification_report
from tensorflow.keras.layers import Dropout

# we extract the data
x_train_images=np.load('augmented_training_images_v2.npy')
x_train_images=x_train_images/255
x_train_angle_and_ratio_features=np.load('augmented_training_angle_ratio_features_v2.npy'
)

x_train_i=Input(shape=x_train_images[0].shape)
x_train_f=Input(shape=x_train_angle_and_ratio_features[0].shape)

c=Conv2D(16, (3,3), activation='relu')(x_train_i)
c=MaxPooling2D(pool_size=(3,3))(c)
```

```
c=Conv2D(32, (3,3), activation='relu')(c)
c=MaxPooling2D(pool_size=(2,2))(c)
c=Conv2D(64, (3,3), activation='relu')(c)
c=MaxPooling2D(pool_size=(2,2))(c)
c=Flatten()(c)
c=Dense(100, activation='relu')(c)

d=Dense(100, activation='sigmoid')(x_train_f)

merged=concatenate([c,d])

m=Dense(9,activation='softmax')(merged)

model = Model(inputs=[x_train_i,x_train_f], outputs=m)

model.compile(loss='categorical_crossentropy',
            optimizer=RMSprop(learning_rate=0.001),
            metrics=['acc'])



y_train=np.load('augmented_y_train_v2.npy')
y_train=np.where(y_train==10, 8, y_train)

y = np.zeros((y_train.size, y_train.max()+1))
y[np.arange(y_train.size),y_train] = 1


# we train the model.
hist = model.fit([x_train_images,x_train_angle_and_ratio_features], y,
          batch_size=100, epochs=10, validation_split=0.2 )


# we observe

plt.plot(hist.history['acc'])
plt.plot(hist.history['val_acc'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper left')
plt.show()

plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper right')
plt.show()
```

```
# We save this model
model.save('model_30.h5')

model.summary()

x_test_images=np.load('test_images_v2.npy')
x_test_images=x_test_images/255
x_test_features_ar=np.load('test_angle_ratio_features_v2.npy')

y_test=np.load('y_test_v2.npy')
y_test=np.where(y_test==10,8,y_test)

y = np.zeros((y_test.size, y_test.max()+1))
y[np.arange(y_test.size),y_test] = 1

# predict probabilities for test set
pred = model.predict([x_test_images,x_test_features_ar], batch_size=100,verbose=1)
predicted=np.argmax(pred,axis=1)
report=classification_report(np.argmax(y, axis=1),predicted)
print(report)
```

MODEL-2

```
import numpy as np
import pandas as pd
from pandas.core.accessor import DirNamesMixin
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
#from tf.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Dropout, Input,
concatenate
from tensorflow.keras import layers
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.optimizers import RMSprop

from sklearn.datasets import make_circles
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import confusion_matrix
from tensorflow.python.keras.engine.training import Model

from sklearn.metrics import classification_report
```

```
from tensorflow.keras.layers import Dropout

# we extract the data
x_train_images=np.load('augmented_training_images_v2.npy')
x_train_images=x_train_images/255
x_train_angle_and_ratio_features=np.load('augmented_training_angle_ratio_features_v2.npy'
)

x_train_i=Input(shape=x_train_images[0].shape)
x_train_f=Input(shape=x_train_angle_and_ratio_features[0].shape)

c=Conv2D(16, (2,2), activation='relu')(x_train_i)
c=MaxPooling2D(pool_size=(3,3))(c)
c=Conv2D(32, (2,2), activation='relu')(c)
c=MaxPooling2D(pool_size=(2,2))(c)
c=Conv2D(64, (2,2), activation='relu')(c)
c=MaxPooling2D(pool_size=(2,2))(c)
c=Flatten()(c)
c=Dense(60, activation='sigmoid')(c)

d=Dense(40, activation='sigmoid')(x_train_f)

merged=concatenate([c,d])

m=Dense(9,activation='softmax')(merged)

model = Model(inputs=[x_train_i,x_train_f], outputs=m)

model.compile(loss='categorical_crossentropy',
            optimizer=RMSprop(learning_rate=0.001),
            metrics=['acc'])



y_train=np.load('augmented_y_train_v2.npy')
y_train=np.where(y_train==10, 8, y_train)

y = np.zeros((y_train.size, y_train.max()+1))
y[np.arange(y_train.size),y_train] = 1


# we train the model.
hist = model.fit([x_train_images,x_train_angle_and_ratio_features], y,
          batch_size=100, epochs=10, validation_split=0.2 )


# we observe

plt.plot(hist.history['acc'])
plt.plot(hist.history['val_acc'])
```

```
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper left')
plt.show()

plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper right')
plt.show()

# We save this model
model.save('model_32.h5')

model.summary()

x_test_images=np.load('test_images_v2.npy')
x_test_images=x_test_images/255
x_test_features_ar=np.load('test_angle_ratio_features_v2.npy')

y_test=np.load('y_test_v2.npy')
y_test=np.where(y_test==10,8,y_test)

y = np.zeros((y_test.size, y_test.max()+1))
y[np.arange(y_test.size),y_test] = 1

# predict probabilities for test set
pred = model.predict([x_test_images,x_test_features_ar], batch_size=100,verbose=1)
predicted=np.argmax(pred,axis=1)
report=classification_report(np.argmax(y, axis=1),predicted)
print(report)
```

MODEL-3

```
import numpy as np
import pandas as pd
from pandas.core.accessor import DirNamesMixin
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
#from tf.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Dropout, Input,
concatenate
from tensorflow.keras import layers
from tensorflow.keras.utils import to_categorical
```

```
from tensorflow.keras.optimizers import RMSprop

from sklearn.datasets import make_circles
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import confusion_matrix
from tensorflow.python.keras.engine.training import Model

from sklearn.metrics import classification_report
from tensorflow.keras.layers import Dropout

# we extract the data
x_train_images=np.load('augmented_training_images_v2.npy')
x_train_images=x_train_images/255
x_train_angle_and_ratio_features=np.load('augmented_training_angle_ratio_features_v2.npy'
)

x_train_i=Input(shape=x_train_images[0].shape)
x_train_f=Input(shape=x_train_angle_and_ratio_features[0].shape)

c=Conv2D(32, (2,2), activation='relu')(x_train_i)
c=MaxPooling2D(pool_size=(3,3))(c)
c=Conv2D(64, (2,2), activation='relu')(c)
c=MaxPooling2D(pool_size=(2,2))(c)
c=Flatten()(c)
c=Dense(50, activation='sigmoid')(c)

d=Dense(50, activation='sigmoid')(x_train_f)

merged=concatenate([c,d])

m=Dense(9,activation='softmax')(merged)

model = Model(inputs=[x_train_i,x_train_f], outputs=m)

model.compile(loss='categorical_crossentropy',
            optimizer=RMSprop(learning_rate=0.001),
            metrics=['acc'])



y_train=np.load('augmented_y_train_v2.npy')
y_train=np.where(y_train==10, 8, y_train)

y = np.zeros((y_train.size, y_train.max()+1))
y[np.arange(y_train.size),y_train] = 1
```

```
# we train the model.
hist = model.fit([x_train_images,x_train_angle_and_ratio_features], y,
          batch_size=100, epochs=10, validation_split=0.2 )


# we observe

plt.plot(hist.history['acc'])
plt.plot(hist.history['val_acc'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper left')
plt.show()

plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper right')
plt.show()

# We save this model
model.save('model_33.h5')

model.summary()

x_test_images=np.load('test_images_v2.npy')
x_test_images=x_test_images/255
x_test_features_ar=np.load('test_angle_ratio_features_v2.npy')

y_test=np.load('y_test_v2.npy')
y_test=np.where(y_test==10,8,y_test)

y = np.zeros((y_test.size, y_test.max()+1))
y[np.arange(y_test.size),y_test] = 1

# predict probabilities for test set
pred = model.predict([x_test_images,x_test_features_ar], batch_size=100,verbose=1)
predicted=np.argmax(pred,axis=1)
report=classification_report(np.argmax(y, axis=1),predicted)
print(report)
```

MODEL-4

```
import numpy as np
import pandas as pd
from pandas.core.accessor import DirNamesMixin
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
#from tf.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Dropout, Input,
concatenate
from tensorflow.keras import layers
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.optimizers import RMSprop

from sklearn.datasets import make_circles
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import confusion_matrix
from tensorflow.python.keras.engine.training import Model

from sklearn.metrics import classification_report
from tensorflow.keras.layers import Dropout

# we extract the data


x_train_images=np.load('augmented_training_images_v2.npy')
x_train_images=x_train_images/255
x_train_angle_and_ratio_features=np.load('augmented_training_angle_ratio_features_v2.npy'
)

x_train_i=Input(shape=x_train_images[0].shape)
x_train_f=Input(shape=x_train_angle_and_ratio_features[0].shape)

c=Conv2D(8, (2,2), activation='relu')(x_train_i)
c=MaxPooling2D(pool_size=(2,2))(c)
c=Conv2D(32, (2,2), activation='relu')(c)
c=MaxPooling2D(pool_size=(2,2))(c)
c=Flatten()(c)


c=Dense(60, activation='relu')(c)

d=Dense(40, activation='relu')(x_train_f)
```

```
merged=concatenate([c,d])

m=Dense(9,activation='softmax')(merged)

model = Model(inputs=[x_train_i,x_train_f], outputs=m)

model.compile(loss='categorical_crossentropy',
              optimizer=RMSprop(learning_rate=0.001),
              metrics=['acc'])



y_train=np.load('augmented_y_train_v2.npy')
y_train=np.where(y_train==10, 8, y_train)

y = np.zeros((y_train.size, y_train.max()+1))
y[np.arange(y_train.size),y_train] = 1


hist = model.fit([x_train_images,x_train_angle_and_ratio_features], y,
          batch_size=100, epochs=10, validation_split=0.2 )


# we observe

plt.plot(hist.history['acc'])
plt.plot(hist.history['val_acc'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper left')
plt.show()

plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper right')
plt.show()

# We save this model
model.save('model_34.h5')

model.summary()

x_test_images=np.load('test_images_v2.npy')
x_test_images=x_test_images/255
x_test_features_ar=np.load('test_angle_ratio_features_v2.npy')
```

```
y_test=np.load('y_test_v2.npy')
y_test=np.where(y_test==10,8,y_test)

y = np.zeros((y_test.size, y_test.max()+1))
y[np.arange(y_test.size),y_test] = 1

# predict probabilities for test set
pred = model.predict([x_test_images,x_test_features_ar], batch_size=100,verbose=1)
predicted=np.argmax(pred,axis=1)
report=classification_report(np.argmax(y, axis=1),predicted)
print(report)
```

THE COMBINED MODEL

```
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from tensorflow.keras.models import load_model
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Dropout, Input,
concatenate
from tensorflow.python.keras.backend import shape
from tensorflow.python.keras.engine.training import Model

from sklearn.datasets import make_circles
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

drive_directory='/Volumes/macpart/capstone_project_files/Capstone_venv_2'
original_directory='/Users/ilkerarslan/Desktop/Capstone_venv_2'
os.chdir(drive_directory)

# we extract the data
x_train_images=np.load('augmented_training_images_v2.npy')
x_train_images=x_train_images/255
x_train_angle_and_ratio_features=np.load('augmented_training_angle_ratio_features_v2.npy'
)

os.chdir(original_directory)

model_30=load_model('model_30.h5')
```

```
print('model 30 is loaded')
#model_31=load_model('model_31.h5')
model_32=load_model('model_32.h5')
print('model 32 is loaded')
model_33=load_model('model_33.h5')
print('model 33 is loaded')
model_34=load_model('model_34.h5')
print('model 34 is loaded')

os.chdir(drive_directory)

feature_30_x_train=np.load('feature_30_x_train.npy')
print('prediction 30 is done.')

#feature_31_x_train=model_31.predict([x_train_images,x_train_angle_and_ratio_features])
feature_32_x_train=np.load('feature_32_x_train.npy')
print('prediction 32 is done.')
feature_33_x_train=np.load('feature_33_x_train.npy')
print('prediction 33 is done.')
feature_34_x_train=np.load('feature_34_x_train.npy')
print('predictons are made.')
import tensorflow as tf


f30=Input(shape=feature_30_x_train[0].shape)
f32=Input(shape=feature_32_x_train[0].shape)
f33=Input(shape=feature_33_x_train[0].shape)
f34=Input(shape=feature_34_x_train[0].shape)


print(feature_33_x_train[0].shape)

merged=tf.math.add_n([f30,f32,f33,f34])


print(merged.shape)

y=Dense(9,activation='softmax')(merged)

model=Model(inputs=[f30,f32,f33,f34],outputs=y)

# we determine the loss, the optimizer, the metric.
model.compile(loss='categorical_crossentropy',
              optimizer=RMSprop(learning_rate=0.001),
              metrics=['acc'])


y_train=np.load('augmented_y_train_v2.npy')
y_train=np.where(y_train==10, 8, y_train)
```

```
y = np.zeros((y_train.size, y_train.max()+1))
y[np.arange(y_train.size),y_train] = 1


# we train the model.
hist =
model.fit([feature_30_x_train,feature_32_x_train,feature_33_x_train,feature_34_x_train], y,
          batch_size=100, epochs=20, validation_split=0.1 )


# we observe

plt.plot(hist.history['acc'])
plt.plot(hist.history['val_acc'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper left')
plt.show()

plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Val'], loc='upper right')
plt.show()

os.chdir(original_directory)
# We save this model
model.save('mixed_model_last_vesion.h5')

print(model.summary())
os.chdir(drive_directory)

test_images_v2=[]
test_angle_ratio_features_v2=[]
test_images_v1=np.load('test_images_as_arrays.npy')
test_angle_ratio_features_v1=np.load('x_test_angle_and_ratio_features_after_PCA_95.npy')

x_test_images=np.load('test_images_v2.npy')
x_test_images=x_test_images/255
x_test_features_ar=np.load('test_angle_ratio_features_v2.npy')

y_test=np.load('y_test_v2.npy')
print('test data is loaded')
y_test=np.where(y_test==10,8,y_test)

y = np.zeros((y_test.size, y_test.max()+1))
y[np.arange(y_test.size),y_test] = 1
```

```
feature_30_x_test=np.load('feature_30_x_test.npy')
feature_32_x_test=np.load('feature_32_x_test.npy')
feature_33_x_test=np.load('feature_33_x_test.npy')
feature_34_x_test=np.load('feature_34_x_test.npy')
print('test data is predicted.')
# predict probabilities for test set
pred =
model.predict([feature_30_x_test,feature_32_x_test,feature_33_x_test,feature_34_x_test],
batch_size=100,verbose=1)
predicted=np.argmax(pred,axis=1)
report=classification_report(np.argmax(y, axis=1),predicted)
print(report)
```

REFERENCES

[1] iMotions, [online] Available: https://imotions.com/contact-us/

[2] Pietschnig J., Aigner-Wöber R., Reischenböck N., Kryspin-Exner I., Moser D., Klug S., et al. “Facial emotion recognition in patients with subjective cognitive decline and mild cognitive impairment”, *Int Psychogeriatr* 2016; 28: 477-85.

[3] N. Rodrigez-Diaz, D. Apandi, F. Sukno, X. Binefa,“Machine Learning based Lie Detector applied to a Collected and Annotated Dataset” *arXiv*, [online] Available: https://arxiv.org/abs/2104.12345

[4] Z. Kowalczk, M. Czubenko, T. Merta, “Emotion monitoring system for drivers”, *IFAC-PapersOnLine,* Vol:52, Issue:8, p-200-205.

[5] S. Li and W. Deng, "Deep Facial Expression Recognition: A Survey," in *IEEE Transactions on Affective Computing* (2020) doi: 10.1109/TAFFC.2020.2981446.

[6] Dunau P., Bonny M., Huber M.F., Beyerer J. (2019) “Reduced Feature Set for Emotion Recognition Based on Angle and Size Information”, In: Strand M., Dillmann R., Menegatti E., Ghidoni S. (eds) Intelligent Autonomous Systems 15. IAS 2018. *Advances in Intelligent Systems and Computing, vol 867*. Springer, Cham. https://doi.org/10.1007/978-3-030-01370-7_46

[7] G. Viswanatha Reddy, C.V.R. Dharma Savarni, Snehasis Mukherjee, “Facial epression recognition in the wild, by fusion of deep learnt and hand-crafted features”, *Cognitive Systems Research*, Vol: 62, p-23-34, August 2020.

[8] M. A. Jalal, L. Mihaylova and R. K. Moore, "An End-to-End Deep Neural Network for Facial Emotion Classification," 2019 22th International Conference on Information Fusion (FUSION), 2019, pp. 1-7.

[9] King, Davis E., “Dlib-ml: A Machine Learning Toolkit.” *J. Mach. Learn. Res.* 10 (2009): 1755-1758.

[10] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1867-1874, doi: 10.1109/CVPR.2014.241.

[11] S. Ren, K. He, R. Girshick, J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks”, *IEEE Trans. Pattern Anal. Mach. Intell.*, 39 (2017), pp. 1137-1149.

[12] Akhand, M. A. H., Shuvendu Roy, Nazmul Siddque, Md A. S. Kamal, and Tetsuya Shimamura, "Facial Emotion Recognition Using Transfer Learning in the Deep CNN", *Electronics* 10, no. 9: 1036 (2021) https://doi.org/10.3390/electronics10091036.

[13] Ng, H., Nguyen V. D., Vonikakis, V., Winkler S., "Deep Learning for Emmotion Recognition on Small Datasets Using Transfer Learning", 2015. Proceedings of the 2015 ACM on International Conference on Multimodal Interaction, pages 443-449, *Association for Computing Machinery, New York, NY, USA*. https://doi.org/10.1145/2818346.2830593.

[14] Zhuang F., Qi Z., Duan K., Xi D., Zhu Y., Zhu H., Xiong H., He Q., "A comprehensive Survey on Transfer Learning" *arXiv*, [online] Available: arXiv:1911.02685 .