

VISUAL BMI ESTIMATION

SUKLESH S RAO (16Z355)

HAREESHWAR K (17Z317)

NISHANTH S (17Z332)

SHIVANI B (17Z348)

TEJAS H BADANI (17Z354)

15Z720 PROJECT WORK I

Dissertation submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF ENGINEERING

Branch: COMPUTER SCIENCE AND ENGINEERING

of Anna University



NOVEMBER 2020

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE – 641 004

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE – 641 004

VISUAL BMI ESTIMATION

Bona fide record of work done by

SUKLESH S RAO (16Z355)

HAREESHWAR K (17Z317)

NISHANTH S (17Z332)

SHIVANI B (17Z348)

TEJAS H BADANI (17Z354)

Dissertation submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF ENGINEERING

Branch: COMPUTER SCIENCE AND ENGINEERING

of Anna University

NOVEMBER 2020

.....
Dr.K. Sathiyapriya

Faculty guide

.....
Dr. G.Sudha Sadasivam

Head of the Department

Certified that the candidate was examined in the viva-voce examination held on

.....
(Internal Examiner)

.....
(External Examiner)

CERTIFICATE

Certified that this report titled “**VISUAL BMI ESTIMATION**,” for the Project Work 1 (15Z720) is a bonafide work of **SUKLESH S RAO(16Z355), HAREESHWAR K(17Z317), NISHANTH S (17Z332), SHIVANI B (17Z348), TEJAS H BADANI (17Z354)** who have carried out the work under my supervision for the partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science and Engineering. Certified further that to the best of my knowledge and belief, the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or an award was conferred on an earlier occasion.

Place: Coimbatore

Date: 19.11.2020



Dr.K. Sathiyapriya

Department of Computer Science and Engineering

PSG College of Technology

Coimbatore - 641004

COUNTERSIGNED



HEAD

Department of Computer Science and Engineering

PSG College of Technology, Coimbatore - 641004

ACKNOWLEDGEMENT

We would like to extend our sincere thanks to our Principal In-charge, **Dr. K. Prakasan**, for providing us with this opportunity to develop and complete our project in our field of study.

We sincerely express our sincere thanks to our Head of the Department, **Dr. G.Sudha Sadasivam**, for encouraging and allowing us to present the project at our department premises for the partial fulfillment of the requirements leading to the award of BE degree.

We take immense pleasure in conveying our thanks and a deep sense of gratitude to the Programme Coordinator, **Mr. Engels Rajangam**, Associate Professor, Department of Computer Science and Engineering, for his valuable suggestions and motivation all through the course of the project.

It is our privilege to express our sincere regards to our project guide, **Dr. K.Sathiyapriya**, Assistant Professor (SG), Department of Computer Science and Engineering, for her valuable inputs, guidance, encouragement, whole-hearted cooperation, and constructive criticism throughout the duration of our project.

We express our sincere thanks to all the panel members and faculty members of the department for helping us in resolving the problems incurred during the project work. We take this opportunity to thank all staff members of the department.

SYNOPSIS

In many of the scenarios of logging into a fitness application or entering data related to health, there exists a hindrance of entering the same data repeatedly. This process can be made easier if the individual's BMI is calculated automatically in these fitness applications to create a personalized fitness plan. The BMI estimator's objective is to efficiently estimate the BMI of an individual with just a visual input of their face.

This application will allow users to upload their images, from which facial recognition is applied to extract facial features. The BMI prediction system consists of a pre-trained face recognition model for extracting the facial features, and the BMI estimation is done by using a fully connected neural network. BMI estimation is used to find a numerical value corresponding to the BMI for the given facial image data in real-time.

The BMI estimation model is trained with 1026 images and validated to obtain a generalized fit over the data. The error rate is a maximum of ± 2 from the actual BMI. Here these predictive analytics are based on AI and ML programs from the vast volumes of data available from various sources. The proposed system can predict the BMI of an individual with an accuracy of 87.58% using facial recognition and deep learning.

CONTENTS

CHAPTER	Pg.No.
Acknowledgment.....	(i)
Synopsis.....	(ii)
List of Figures	(iii)
List of tables.....	(iv)
1. INTRODUCTION.....	1
1.1. Introduction	1
1.2. Motivation	1
1.3. Problem Statement	1
1.4. Objective	2
1.5. Scope and applications	2
2. SYSTEM STUDY.....	3
3. SYSTEM ANALYSIS	6
3.1. Functional Requirements	6
3.2. Non-functional Requirements	6
3.2.1. Usability Requirements	6
3.2.2. Reliability Requirements	6
3.2.3. Efficiency Requirements	6
3.3. Experimental setup	7
3.3.1. Hardware requirements	7
3.3.2. Software requirements	7
3.4. Feasibility analysis	7
3.4.1. Economic feasibility	7
3.4.2. Technical feasibility	7
3.4.3. Operational feasibility	8
4. SYSTEM DESIGN	9
4.1. Use case diagram	9
4.2. Activity Diagram	10
4.3. Architecture Diagram	10
4.4. Dataset Description	14

5.	SYSTEM IMPLEMENTATION.....	15
5.1.	Loading the data	15
5.2.	Pre-processing of image	16
5.3.	Facial Feature Extraction	17
5.4.	Facial feature extraction for images in the dataset	17
5.5	Splitting into train and test dataset	18
5.6	Building the fully connected estimator model	19
5.7	Training and Validation	20
5.8	Backend Integration	20
6.	TESTING.....	22
6.1.	Unit testing	22
6.1.1.	Image pre-processing	22
6.1.2.	Face detection	22
6.1.3.	BMI estimation	23
6.2	Integration testing	23
7.	RESULTS	25
7.1.	Login	25
7.2.	Upload images	25
7.3.	BMI prediction	26
7.4	Accuracy	27
8.	CONCLUSION.....	28
9.	BIBLIOGRAPHY.....	29
10.	APPENDIX.....	30

LIST OF FIGURES

Figure No	Figure name	page no.
4.1	Use case diagram	9
4.2	Activity diagram	10
4.3	Architecture diagram	11
4.4	Modified VGG-Face model	12
4.5	BMI estimation model	13
5.1	Loading the data	15
5.2	Pre-processing the data	16
5.3	Facial feature extraction model	17
5.4	Extracting features to a csv file	18
5.5	Splitting into train and test set	18
5.6	Fully connected estimator model	19
5.7	Training and validation	20
5.8	Backend integration	21
7.1	Placeholder (opening login screen)	25
7.2	Screen to upload images	26
7.3	BMI prediction screen	26
7.4	Final epochs of the estimation model	27
7.5	Training and validation loss	27

LIST OF TABLES

Table No.	Table name	page no.
2.1	Summary of system study	5
6.1	Summary of unit testing for Image processing	22
6.2	Summary of unit testing for face detection	22
6.3	Summary of unit testing for BMI estimation	23
6.4	Summary of integration testing	23

CHAPTER 1

INTRODUCTION

1.1.INTRODUCTION

The human face exhibits information pertaining to identity, a person's disposition, demeanour, as well as to attributes such as gender, age and ethnicity. From the perspective of biometrics, emphasis has predominantly been placed on facial recognition. Face images contain much biometric information, such as identity, gender, age, weight, etc. Decoding facial cues has attracted much attention from sociologists and computer scientists. Perceptions of people's weight based on photographs of their face is positively correlated with their body mass index. The Body Mass Index (BMI) is a general body fat indicator widely used in medical research.

The BMI is calculated by using the formula,

$$\text{Body Mass Index} = \frac{\text{weight} * 703}{\text{height}^2}$$

It can reveal various health and lifestyle issues. With the advanced algorithms and deep learning models trained on vast and diverse datasets available today, the capabilities of facial image analysis can be extended to estimate personal health characteristics such as BMI by extrapolations from the facial biometric data. This project aims to use specified and distinct facial features of a person to accurately predict the BMI of the said person.

Machine learning is an area of Artificial Intelligence (AI) that provides a system the ability to automatically learn and improve from experience without being explicitly programmed. Deep learning, where deep neural networks with several nodes and layers are involved, forms a major subset of machine learning and Computer Vision has always been a primary area of focus in this domain. The primary aim of facial image analysis in computer vision is to extract valuable information by interpreting perceived electronic data from face images.

1.2.MOTIVATION

Recent research shows that facial fatness is associated with perceived health and is highly correlated with body mass index (BMI) prediction. As a body fat indicator, BMI is widely used in health monitoring and health research. There are close connections between BMI and some diseases, such as cancers, unstable angina and type 2 diabetes and cardiovascular disease etc. So, it has now become common to know the BMI of a person for diagnosis. There are many applications to keep track of the BMI but all of them require manual input of BMI. The traditional method of calculating is a tedious process while the proposed solution can be used to estimate the BMI in a much faster way.

1.3.PROBLEM STATEMENT

It is difficult to evaluate the BMI accurately for an individual. Nowadays, a lot of fitness apps ask the user to provide his/her BMI details. Traditionally, a person would have to self-report their body-mass index use a special device to calculate it. Due to this difficulty, users tend to provide inaccurate BMI when it is needed. This leads to incorrect diagnosis and decreased accuracy of fitness applications.

1.4.OBJECTIVE

The objective of this project is to effectively estimate the BMI of the user with facial features and track BMI changes with the change in facial features over time along with the proper indications to the user in case of extremities such as being underweight or obesity thereby serving as a simple and easy to use BMI tracker.

1.5.SCOPE AND APPLICATIONS

The scope of this system is to reduce the difficulty in estimating BMI in several areas. Some of them are:

- The system can be integrated with fitness applications to allow users to estimate BMI by using just a selfie.
- The system can be used by individual users to keep track of their personal BMI changes with changes in their facial appearance over time.

CHAPTER 2

SYSTEM STUDY

This chapter discusses about the knowledge that is gained from the following materials which include substantive findings as well as theoretical contributions to BMI analysis from facial images.

Jiang et al. [1] used a two-stage learning framework, which consists of BMI-related facial features learning and BMI estimator learning. The BMI-related face model was learned based on a pre-trained deep face model. Then two different strategies were analyzed for modeling the BMI labels with probability distributions and a projection optimization was achieved by maximizing the correlation between the facial features and the assigned labels. Then the BMI estimator was learned from the projected features and assigned labels.

Jiang et al. [2] studied the visual BMI estimation problem systematically based on the facial representation or feature extraction. According to the inherent properties of representations, they grouped them into two types: geometric based and deep learning based. They experimented with two of the existing approaches (VGG-Face and PIGF), and five other facial approaches: PF, PIGF+PF, LightCNN, Centerloss and Arcface.

Dantcheva et al. [3] explored the possibility of estimating height, weight, and BMI from single-shot facial images by using a regression method based on a 50-layer ResNet architecture. They assembled a dataset call VIP_Attribute dataset which consists of 1026 subjects to facilitate the study.

Wen and Guo [4] proposed a computational method for automatically predicting BMI from 2D face images. This was based on the MORPH-II dataset, which obtained mean absolute errors (MAEs) for BMI in the range from 2.65–4.29 for different ethnic categories. The study explored handcrafted features for BMI-estimation and specifically in the method the face was detected, normalized, and an active shape model was fitted, based on which, geometry and ratio features were extracted (cheekbone to jaw width, width to upper facial height ratio, perimeter to area ratio, eye size, lower face to face height ratio, face width to lower face height ratio and mean of eyebrow height), normalized and finally subjected to support vector regression.

Kocabey et al. [5] employed the pre-trained VGG-Face model to extract facial representation for BMI estimation. Then a support vector regression model was learned

to map the facial representation to predicted BMIs. The above two works treated BMI prediction as a regression problem.

Lee et al. [6] proposed in a prediction method of normal and overweight females based on BMI using geometrical facial features only. The features, measured on 2D images, include Euclidean distances, angles and face areas defined by selected soft-tissue landmarks.

Wolffhechel et al. [7] employed a data-driven approach in which statistical models were built using principal components (PCs) derived from objectively defined shape and color characteristics in face images. The predictive power of these models was then compared with models based on previously studied facial proportions (perimeter-to-area ratio, width-to-height ratio, and cheek-to-jaw width). Models based on 2D shape-only PCs, color-only PCs, and 2D shape and color PCs combined each performed significantly and substantially better than models based on one or more of the previously studied facial proportions. They concluded that a non-linear PC model considering both 2D shape and color PCs was the best predictor of BMI.

Paper Name	Author	Key Points	Year
Visual-BMI estimation from face images using a label distribution-based method	Min Jiang, Guodong Guo, Guowang Mu	<ul style="list-style-type: none"> Two-stage learning framework. <ul style="list-style-type: none"> BMI-related facial features label distribution-based BMI estimator Models the single BMI value as a discrete probability distribution over a range of BMI 	2020
On visual BMI analysis from facial images	Min Jiang, Yuanyuan Shang, Guodong Guo	<ul style="list-style-type: none"> Based on the characteristics and performance of different facial representations Deep model-based methods perform better than geometry-based methods. Among them, the VGG-Face and Arcface show more robustness VGG-Face and PIGF are more robust than the others to head pose variations 	2019
Show me your face and I will tell you your height, weight and body mass index	Antitza Dantcheva, Francois Bremond, Piotr Bilinski	<ul style="list-style-type: none"> Regression pattern classification problems Proposes a regression method based on the 50-layers ResNet-architecture. Performs a gender-based analysis of the prediction of height, weight and BMI 	2018
Face-to-BMI: Using Computer Vision to Infer Body Mass Index on Social Media	Enes Kocabey, Mustafa Camurcu, Ferda Ofllli, Yusuf Aytar, Javier Marin, Antonio Torralba,	<ul style="list-style-type: none"> Uses computer vision techniques to infer a person's BMI from social media images This BMI prediction system is composed of two stages: <ul style="list-style-type: none"> deep feature extraction <ul style="list-style-type: none"> VGG-Net VGG-Face (Better one) training a regression model 	2017

	Ingmar Weber	▪ epsilon support vector regression	
Face morphology: Can it tell us something about body weight and fat?	Pascali, MA, Giorgi, D, Bastiani, L, Buzzigoli, E, Henriquez, P, Matuszewski, BJ, Morales, MA and Colantonio, S	<ul style="list-style-type: none"> Proposes a method for an automatic extraction of geometric features From 3D facial data acquired with low-cost depth scanners. Experimental results show that these measurements are highly correlated with weight, BMI which are markers of central obesity 	2016
Testing the Utility of a Data-Driven Approach for Assessing BMI from Face Images	Karin Wolffhechel, Amanda C. Hahn, Hanne Jarmer, Claire I. Fisher, Benedict C. Jones, Lisa M. DeBruine	<ul style="list-style-type: none"> data-driven approach statistical models were built using principal components Non-linear PC model considering both 2D shape and color PCs was the best predictor of BMI. 	2015

Table 2.1 Summary of system study

From this system study, it can be concluded that a two-stage model involving facial feature extraction and regression modelling works best for visual BMI estimation.

CHAPTER 3

SYSTEM ANALYSIS

This chapter describes the system analysis which is conducted for the purpose of studying a system or its parts in order to identify its objectives.

3.1 FUNCTIONAL REQUIREMENTS

1. Capturing of image of the user
 - Enabling users to click pictures using the camera of the mobile device used.
2. Allowing user to upload images
 - Enabling users to upload images directly from their gallery.
3. Face Detection Algorithm
 - Trained Deep Learning Model to detect the outlines of the human face from the user image.
4. Geometrical facial feature extraction from detected face
 - Trained Deep Learning Model that extracts geometrical facial features such as height to width ratio, circumference of eyes, and dimensions of lips, nose, cheeks and the chin.
5. Trained BMI estimator model
 - A trained BMI estimator model with optimization features to calculate the BMI of the user from the extracted facial features.
6. Detect BMI and classify users in real time
 - Classifying users into Categories of BMI from their image data in real time.

3.2 NON-FUNCTIONAL REQUIREMENTS

The non-functional requirements are as follows:

3.2.1 Usability Requirements

The system should be user friendly and doesn't require any guidance. In other words, the application has to be as simple as possible, so its users shall use it easily.

3.2.2 Reliability Requirements

The system should not have any unexpected failure and must be reliable at least 98% of the time. In order to avoid any failure's occurrence, the specifications have been respected and followed correctly. The failure of the system is based on the accuracy of

predicting the results.

3.2.3 Efficiency Requirements

The system response time should be adequate and sufficient enough, in order to increase the efficiency of the system. The application should be compatible with different versions of the libraries used.

3.3 EXPERIMENTAL SETUP

3.3.1 Hardware Requirements

The following are the details of the workstation used for implementation and evaluation.

- Processor - Two core processor
- RAM - 1 GB
- Disk Storage - 2 GB

The following hardware requirements are needed for deployment,

- Smart Phone – with Android OS 6.0 and above or iOS 11 and above
- RAM – 1 GB
- Disk Storage – 20 MB

3.3.2 Software Requirements

The following are the software requirements of the system:

1. Google Colaboratory
2. Jupyter Notebook
3. Python 3.7.6
4. Numpy, Pandas, Matplotlib, OpenCV libraries for Python
5. TensorFlow, ScikitLearn and PyTorch Machine Learning libraries for Python

3.4 FEASIBILITY ANALYSIS

A feasibility study is used to determine the practicality of an idea or proposed plan.

3.4.1 Economic Feasibility

As this project is implemented using only open source distributions of python and open source machine learning and image processing libraries, it is economically feasible.

3.4.2 Technical Feasibility

Python has numerous features like third-party modules (PyPi), extensive support for libraries, simplicity, enhanced text processing capabilities all of which makes it a viable option in terms of speed and productivity.

3.4.3 Operational Feasibility

All the functions performed by the system are valid and without conflict. All the functions and constraints specified in the requirements are operational.

CHAPTER 4

SYSTEM DESIGN

This chapter describes the design of the Visual BMI estimator. It talks about the functionalities of the Visual BMI estimator after analyzing the requirements.

4.1. USE CASE DIAGRAM

Fig.4.1 shows the use case diagram for the Visual BMI estimator.

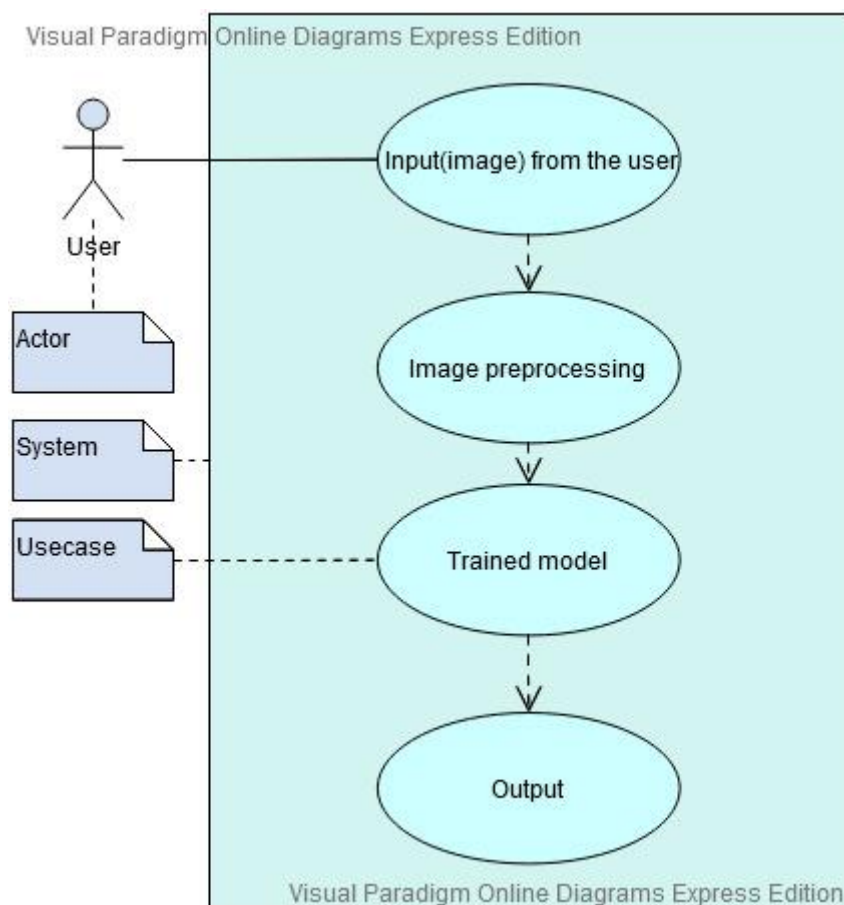


Figure 4. 1: Use case diagram

4.2. ACTIVITY DIAGRAM

Fig.4.2 shows the activity diagram for the Visual BMI estimator.

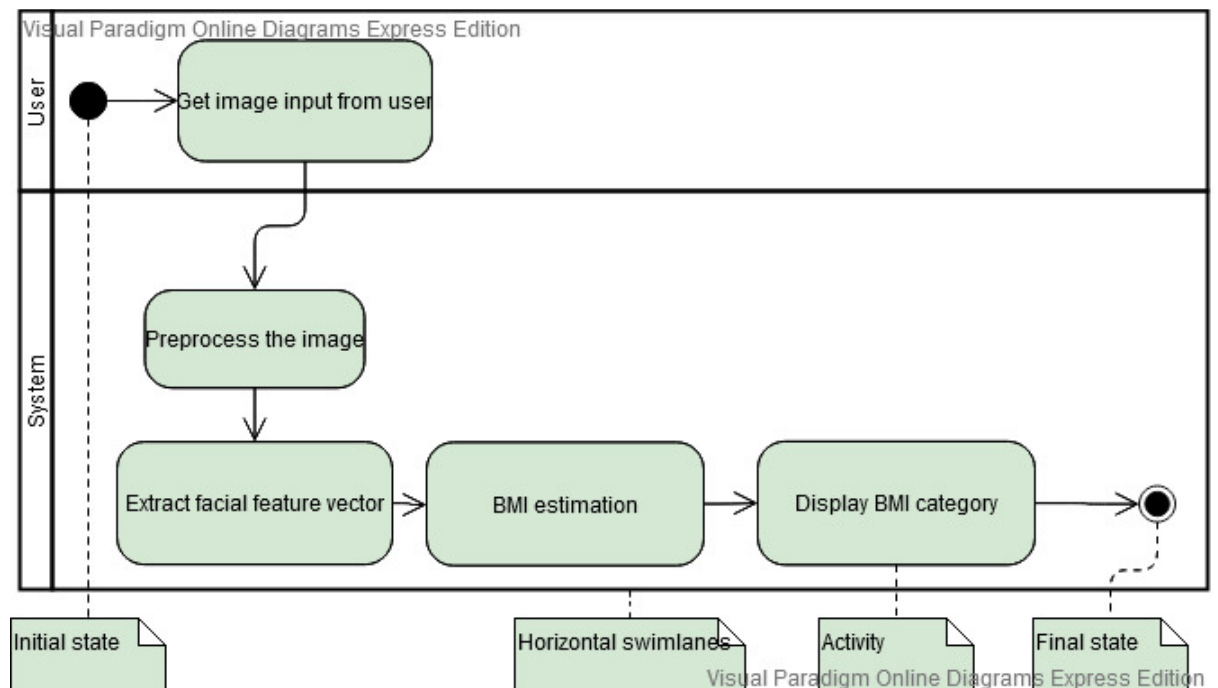


Figure 4. 2:Activity diagram

4.3. ARCHITECTURE DIAGRAM

Fig.4.3 shows the architecture diagram for the Visual BMI estimator. The description of the diagram is as follows,

- The user is asked to upload his or her image to the system
- Now, the pre-processing of the image is done, which includes
 - Converting the image to gray-scale
 - Cropping only the face from the image
- Now the data is passed through the modified VGG-face deep learning model to obtain the facial feature embedding
- Then the embeddings are given as input to the fully connected deep learning model.
- Then based on the trained model, the BMI is estimated.

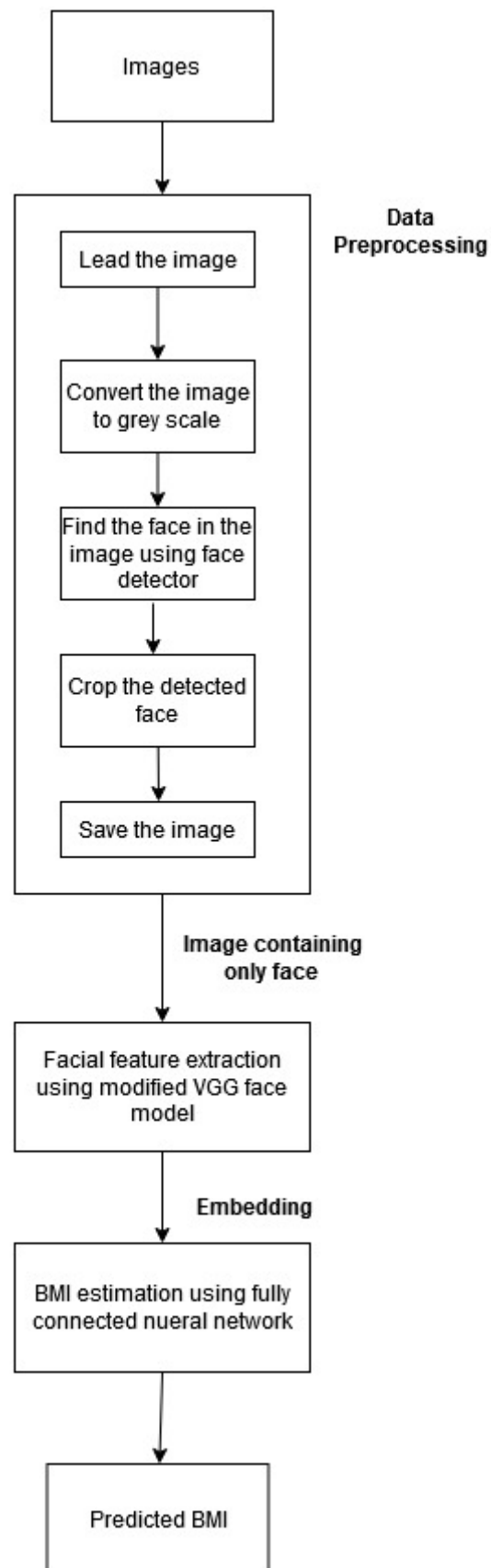
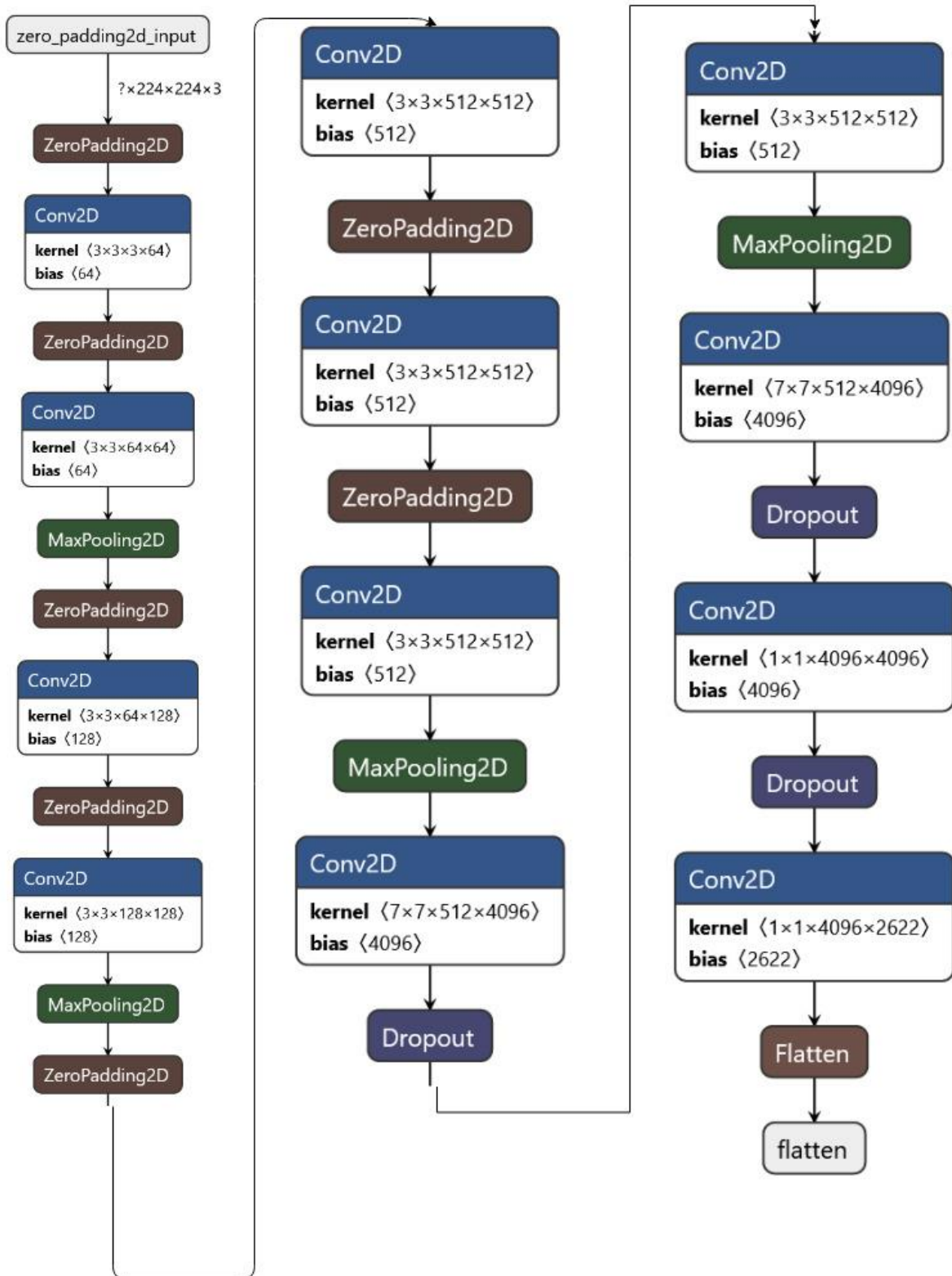


Figure 4. 3 Architecture diagram



Modified VGG-Face

Figure 4. 4 Modified VGG-Face model

Description of the layers used in the modified VGG-Face model depicted in figure 4.4

- Zero padding 2D
 - This layer can add rows and columns of zeros at the top, bottom, left, and right sides of an image tensor.
- Conv2D
 - This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs.
- Max pooling 2D
 - Downsamples the input representation by taking the maximum value over the window defined by pool size for each dimension along the features axis.
 - The window is shifted by strides in each dimension.
- Dropout
 - The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting.
 - Inputs not set to 0 are scaled up by $1/(1 - \text{rate})$ such that the sum over all inputs is unchanged.
- Flatten
 - Flattens the input.
 - Does not affect the batch size.

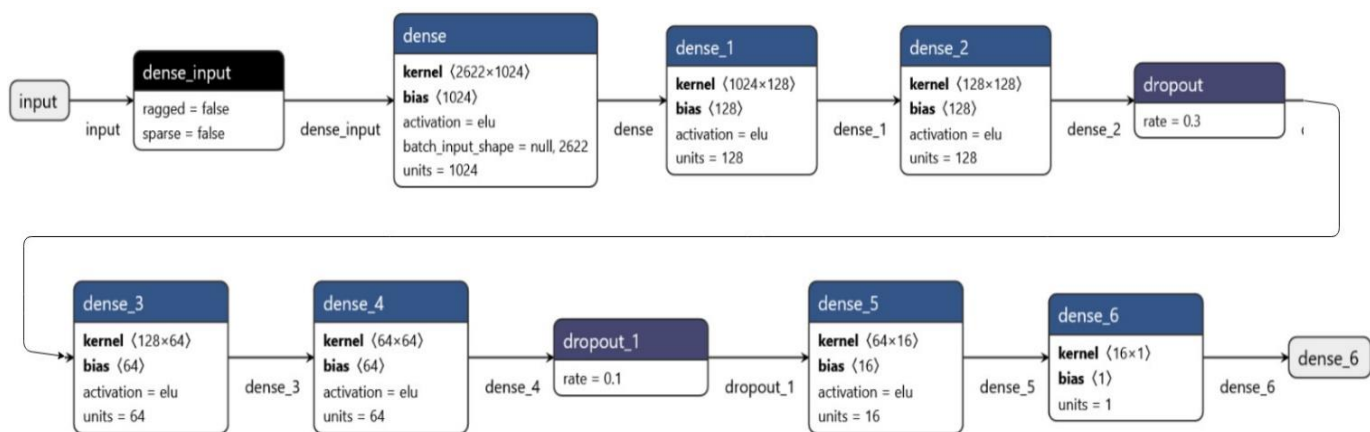


Figure 4. 5: BMI estimation model

Description of the layers used in the modified VGG-Face model depicted in figure 4.5

- Dense layer
 - Dense implements the operation: $\text{output} = \text{activation}(\text{dot}(\text{input}, \text{kernel}) + \text{bias})$ where activation is the element-wise activation function passed as the

activation argument, kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer (only applicable if use_bias is True).

- Dropout
 - The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting.
 - Inputs not set to 0 are scaled up by $1/(1 - \text{rate})$ such that the sum over all inputs is unchanged.
- ELU activation
 - Exponential Linear Unit.
 - The exponential linear unit (ELU) with $\alpha > 0$ is:
 - x if $x > 0$
 - $\alpha * (\exp(x) - 1)$ if $x < 0$
 - The ELU hyperparameter α controls the value to which an ELU saturates for negative net inputs. ELUs diminish the vanishing gradient effect.
 - ELUs have negative values which pushes the mean of the activations closer to zero.
 - Mean activations that are closer to zero enable faster learning as they bring the gradient closer to the natural gradient.
 - ELUs saturate to a negative value when the argument gets smaller.
 - Saturation means a small derivative which decreases the variation and the information that is propagated to the next layer.

4.4. DATASET DESCRIPTION

VIP_Attribute is a dataset comprising face images assembled to study height, weight, and body mass index (BMI) based on facial images. It has 1026 images of 513 female and 513 male celebrities (mainly actors, singers and athletes) collected from the internet. The images include the frontal pose of the subjects. Co-variables include illumination, expression, image quality, and resolution. Further challenges in this dataset are beautification (e.g., photoshop) of the images and the presence of makeup, plastic surgery, beard, and moustache. The annotations related to the subjects' body weight and height were obtained from websites.

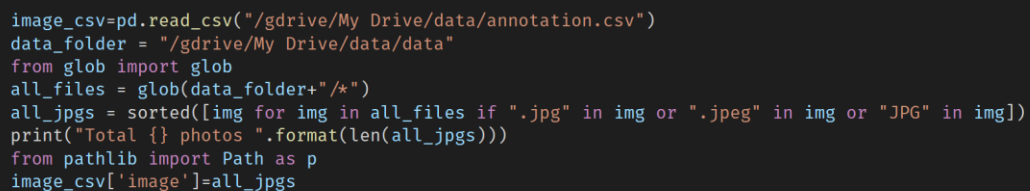
CHAPTER 5

SYSTEM IMPLEMENTATION

This chapter gives a brief explanation of the implementation of the visual BMI analysis system.

5.1 LOADING THE DATA

- Read the .csv files containing the BMI information
- Find the number of images in the data folder
- Add the column image path to a data frame



```
image_csv=pd.read_csv("/gdrive/My Drive/data/annotation.csv")
data_folder = "/gdrive/My Drive/data/data"
from glob import glob
all_files = glob(data_folder+"/*")
all_jpgs = sorted([img for img in all_files if ".jpg" in img or ".jpeg" in img or ".JPG" in img])
print("Total {} photos ".format(len(all_jpgs)))
from pathlib import Path as p
image_csv['image']=all_jpgs
```

Figure 5. 1: Loading the data

5.2 PREPROCESSING OF IMAGE

- Load cnn_face_detector with 'mmod_face_detector'
- Load image
- Convert to grayscale
- For each face, 'rect' provides face location in the image as a pixel location
- Save crop image



```
dnnFaceDetector=dlib.cnn_face_detection_model_v1("/gdrive/My
Drive/data/mmod_human_face_detector.dat")
img=cv2.imread('/DSC_6538.jpg')
print(img)
gray=cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
rects=dnnFaceDetector(gray,1)
left,top,right,bottom=0,0,0,0
for (i,rect) in enumerate(rects):
    left=rect.rect.left() #x1
    top=rect.rect.top() #y1
    right=rect.rect.right() #x2
    bottom=rect.rect.bottom() #y2
    width=right-left
    img_crop=img[top:top+height,left:left+width]
    cv2.imwrite('/gdrive/My Drive/data/sample.jpg',img_crop)
```

Figure 5. 2: Preprocessing the data

5.3 FACIAL FEATURE EXTRACTION

- Define VGG_FACE_MODEL architecture
- Load the pre-trained weights into the model
- The model needs not to classify whether the image consists of a face or not, so we remove the final softmax layer in the output to find out only the embeddings.

```

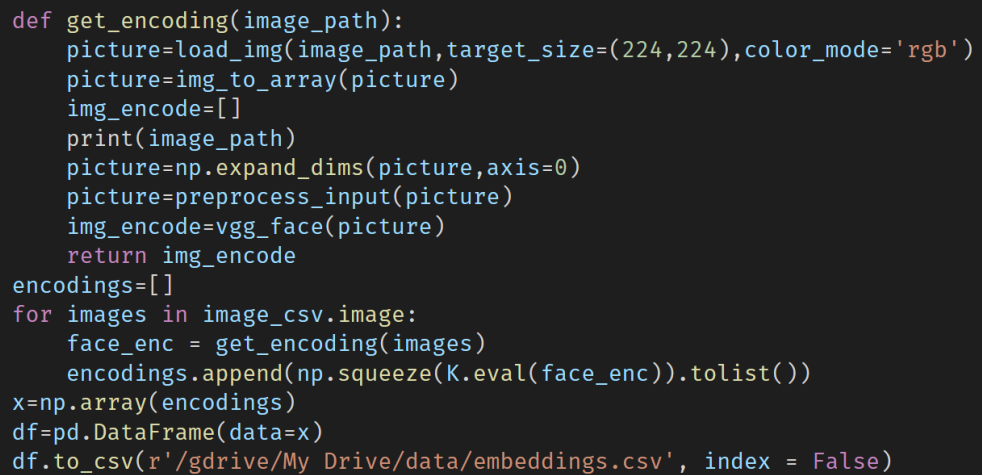
model = Sequential()
model.add(ZeroPadding2D((1,1),input_shape=(224,224, 3)))
model.add(Convolution2D(64, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(128, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(256, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(256, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(256, (3, 3), activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, (3, 3), activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, (3, 3), activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))
model.add(Convolution2D(4096, (7, 7), activation='relu'))
model.add(Dropout(0.5))
model.add(Convolution2D(4096, (1, 1), activation='relu'))
model.add(Dropout(0.5))
model.add(Convolution2D(2622, (1, 1)))
model.add(Flatten())
model.add(Activation('softmax'))
# Load VGG Face model weights
model.load_weights('/gdrive/My Drive/data/vgg_face_weights.h5')
vgg_face=Model(inputs=model.layers[0].input,outputs=model.layers[-2].output)

```

Figure 5. 3 Facial feature extraction model

5.4 FACIAL FEATURE EXTRACTION FOR IMAGES IN THE DATASET

- For every single image, we load it with size(224,224), and we find the embedding vector by passing it to the neural network.
- The embeddings are appended to a data frame, and that is converted to a .csv file for training the next fully connected model



```

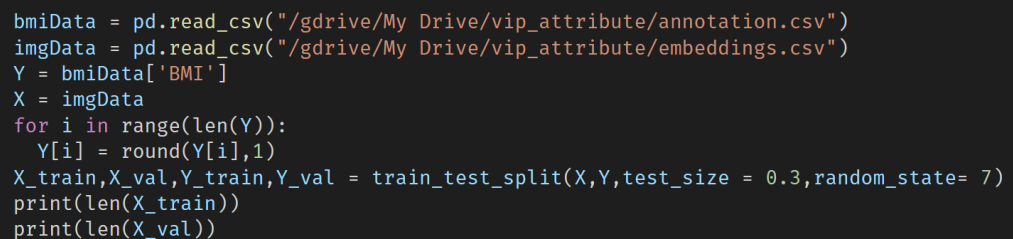
def get_encoding(image_path):
    picture=load_img(image_path,target_size=(224,224),color_mode='rgb')
    picture=img_to_array(picture)
    img_encode=[]
    print(image_path)
    picture=np.expand_dims(picture,axis=0)
    picture=preprocess_input(picture)
    img_encode=vgg_face(picture)
    return img_encode
encodings=[]
for images in image_csv.image:
    face_enc = get_encoding(images)
    encodings.append(np.squeeze(K.eval(face_enc)).tolist())
x=np.array(encodings)
df=pd.DataFrame(data=x)
df.to_csv(r'/gdrive/My Drive/data/embeddings.csv', index = False)

```

Figure 5. 4 Extracting features to a csv file

5.5 SPLITTING INTO TEST AND TRAIN DATASET

- The BMI datafile from the dataset is loaded as a data frame.
- The embeddings.csv file is also loaded as a data frame.
- Then the train test split is done (70% of data for training and 30% of data for validation)



```

bmiData = pd.read_csv("/gdrive/My Drive/vip_attribute/annotation.csv")
imgData = pd.read_csv("/gdrive/My Drive/vip_attribute/embeddings.csv")
Y = bmiData['BMI']
X = imgData
for i in range(len(Y)):
    Y[i] = round(Y[i],1)
X_train,X_val,Y_train,Y_val = train_test_split(X,Y,test_size = 0.3,random_state= 7)
print(len(X_train))
print(len(X_val))

```

Figure 5. 5:Splitting into train and test set

5.6 BUILDING THE FULLY CONNECTED ESTIMATOR MODEL

- With relu, a problem might occur called dead relu that feeds only 0 values to the next layer, and enough number of dead relus will make the loss stagnant and will not decrease with epochs
- Using elu activation, which has a non zero gradient in the negative section instead of 0 in relu when there were only two dropout layers
- A meager difference of 0.02 between the training and the validation loss; hence a decently good model is achieved
- However, the low value of the loss means overfitting. Accordingly, so more dropout layers are added
- But dropout layers were removed when training loss is significantly greater than validation loss.

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 1024)	2685952
dense_1 (Dense)	(None, 128)	131200
dense_2 (Dense)	(None, 128)	16512
dropout (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 64)	8256
dense_4 (Dense)	(None, 64)	4160
dropout_1 (Dropout)	(None, 64)	0
dense_5 (Dense)	(None, 16)	1040
dense_6 (Dense)	(None, 1)	17
Total params: 2,847,137		
Trainable params: 2,847,137		
Non-trainable params: 0		
None		

Figure 5. 6: Fully connected estimator model

5.7 TRAINING AND VALIDATION



Figure 5. 7: Training and validation

5.8 BACKEND INTEGRATION

- The deep learning models were run on a local server using the flask API with which the mobile application communicates

```

app = Flask(__name__)
app.config['MAX_CONTENT_LENGTH'] = 50 * 1024 * 1024
@app.route('/', methods=['GET'])
def hello():
    return 'Home'

@app.route('/test', methods=['POST'])
def test():
    try:
        f = request.files['image1']
        f.save(f.filename)
        print(f.filename)
        # FACE DETECTOR MODEL
        model = Sequential()
        model.add(ZeroPadding2D((1,1),input_shape=(224,224, 3)))
        model.add(Convolution2D(64, (3, 3), activation='relu'))
        model.add(ZeroPadding2D((1,1)))
        model.add(Convolution2D(64, (3, 3), activation='relu'))
        model.add(MaxPooling2D((2,2), strides=(2,2)))
        model.add(ZeroPadding2D((1,1)))
        model.add(Convolution2D(128, (3, 3), activation='relu'))
        model.add(ZeroPadding2D((1,1)))
        model.add(Convolution2D(128, (3, 3), activation='relu'))
        model.add(MaxPooling2D((2,2), strides=(2,2)))
        model.add(ZeroPadding2D((1,1)))
        model.add(Convolution2D(256, (3, 3), activation='relu'))
        model.add(ZeroPadding2D((1,1)))
        model.add(Convolution2D(256, (3, 3), activation='relu'))
        model.add(ZeroPadding2D((1,1)))
        model.add(Convolution2D(256, (3, 3), activation='relu'))
        model.add(MaxPooling2D((2,2), strides=(2,2)))
        model.add(ZeroPadding2D((1,1)))
        model.add(Convolution2D(512, (3, 3), activation='relu'))
        model.add(ZeroPadding2D((1,1)))
        model.add(Convolution2D(512, (3, 3), activation='relu'))
        model.add(ZeroPadding2D((1,1)))
        model.add(Convolution2D(512, (3, 3), activation='relu'))
        model.add(MaxPooling2D((2,2), strides=(2,2)))
        model.add(ZeroPadding2D((1,1)))
        model.add(Convolution2D(512, (3, 3), activation='relu'))
        model.add(ZeroPadding2D((1,1)))
        model.add(Convolution2D(512, (3, 3), activation='relu'))
        model.add(ZeroPadding2D((1,1)))
        model.add(Convolution2D(512, (3, 3), activation='relu'))
        model.add(MaxPooling2D((2,2), strides=(2,2)))
        model.add(Convolution2D(4096, (7, 7), activation='relu'))
        model.add(Dropout(0.5))
        model.add(Convolution2D(4096, (1, 1), activation='relu'))
        model.add(Dropout(0.5))
        model.add(Convolution2D(2622, (1, 1)))
        model.add(Flatten())
        model.add(Activation('softmax'))
        #change the weights file location
        model.load_weights('vgg_face_weights.h5')
        vgg_face=Model(inputs=model.layers[0].input,outputs=model.layers[-2].output)
        #cropping the face
        # Load cnn_face_detector with 'mmod_face_detector'
        dnnFaceDetector=dlib.cnn_face_detection_model_v1("mmod_human_face_detector.dat")
        # Load image
        img=cv2.imread(f.filename)
        # print(img)
        # Convert to gray scale
        gray=cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        # Find faces in image
        rects=dnnFaceDetector(gray,1)
        left,top,right,bottom=0,0,0,0
        # For each face 'rect' provides face location in image as pixel loaction
        for (i,rect) in enumerate(rects):
            left=rect.rect.left() #x1
            top=rect.rect.top() #y1
            right=rect.rect.right() #x2
            bottom=rect.rect.bottom() #y2
            width=right-left
            height=bottom-top
            # Crop image
            img_crop=img[top:top+height,left:left+width]
            #save crop image with person name as image name
            cv2.imwrite('C:\\Users\\Tejas\\Desktop\\Sample.png',img_crop)
        #image encoding
        image_path='sample.png'
        picture=load_img(image_path,target_size=(224,224),color_mode='rgb')
        picture=img_to_array(picture)
        img_encode=[]
        # print(image_path)
        picture=np.expand_dims(picture,axis=0)
        picture=preprocess_input(picture)
        img_encode=vgg_face(picture)
        encodings=[]
        encodings.append(np.squeeze(K.eval(img_encode)).tolist())
        newArr = np.reshape(encodings,(1,2622))
        print(newArr.shape)
        finalModel = load_model("bmiEstimatorModel.h5")
        finalVal = finalModel.predict(newArr)
        print(finalVal[0])
        return jsonify({'prediction': str(finalVal[0][0])})
    except Exception as err:
        return jsonify({'trace': traceback.format_exc()})

    return ('Oops')

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0',port = 5000)

```

Figure 5.8:Backend integration

CHAPTER 6

TESTING

This chapter describes the results of the testing of the modules after implementation.

6.1 UNIT TESTING

The tabulations below contain the summaries of the testing of the individual modules.

6.1.1 IMAGE PREPROCESSING

Test case No.	Inputs	Expected Output	Obtained Output	Test case Acceptance
1	Image with pixel size less than 224x224	Inputs out of range	Inputs out of range	Yes
2	Image with pixel size greater than 224x224	Image Inputs accepted and processed	Image Inputs accepted and processed	Yes

Table 6.1: Summary of testing for Image Preprocessing

6.1.2 FACE DETECTION

Test case No.	Inputs	Expected Output	Obtained Output	Test case Acceptance
1	224x224 pixel size image not containing a human face	Invalid Image Input message	Invalid Image Input message	Yes
2	224x224 pixel size image not containing multiple human faces	Invalid Image Input message	Invalid Image Input message	Yes
3	224x224 pixel size image containing a human face	Face detected and facial feature vector returned	Face detected and facial feature vector returned	Yes

Table 6.2: Summary of unit testing for face detection

6.1.3 BMI ESTIMATION

Test case No.	Inputs	Expected Output	Obtained Output	Test case Acceptance
1	Facial feature vector from face detector for image of person with normal BMI	BMI value between 18.5 to 24.9 and indication for Normal	BMI value between 18.5 to 24.9 and indication for Normal	Yes
2	Facial feature vector from face detector for image of an underweight person	BMI value below 18.5 indication for Underweight	BMI value below 18.5 indication for Underweight	Yes
3	Facial feature vector from face detector for image of an obese person	BMI value above 24.9 and indication for Overweight	BMI value above 24.9 and indication for Overweight	Yes

Table 6.3: Summary of unit testing for BMI estimation

6.2 INTEGRATION TESTING

All the modules passed unit testing. But when they were all integrated together, certain issues had to be resolved. In integrating the machine learning model with the mobile application, the model had to be hosted on the local server and the following errors/issues were identified during this process.

Issue No.	Description	Solution to resolve issue	Completion of resolving issue
1	Incompatibility of machine learning libraries in local machine to the versions with which the model was trained in Google Collaboratory.	Updation of the machine learning libraries – particularly cv2 and tensorflow - in the local machine to the latest versions.	Yes
2	Input dimensions not matching with the input dimensions required by the estimator model	Using image preprocessing functions from keras library to resize the image inputs to the desired dimensions	Yes
3	Passing input image to local server, processing	Hosting the model on a powerful web host like	No

	it, estimate BMI and returning the result to the mobile application run on the local emulator takes a significant amount of time	Amazon Cloud and using a physical mobile device would increase the performance.	
4	Unavailability of access to camera by emulator to facilitate BMI estimation from of live capturing of images of the user.	Hosting the model on a web host would eliminate the need for the emulator to be run on the same machine as local server and a physical mobile device with a camera can be used.	No

Table 6.4: Summary of Integration Testing

CHAPTER 7

RESULTS

This chapter contains a summary of the results achieved while implementing the system.

7.1 LOGIN

Figure 7.1 depicts a screenshot of the opening login screen of the application.

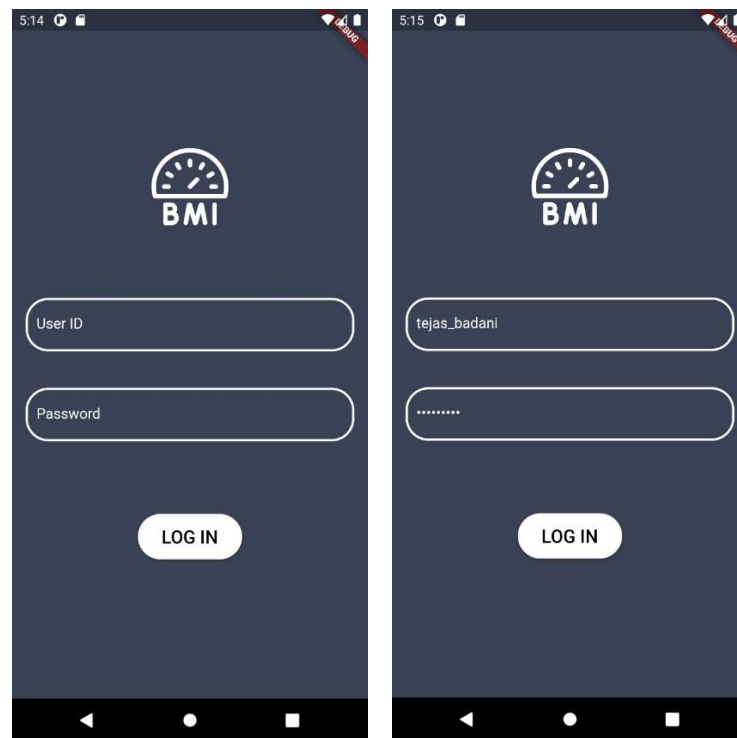


Figure 7.1: Placeholder(opening login screen)

7.2 UPLOAD IMAGES

Figure 7.2 depicts the image upload screen of the application.

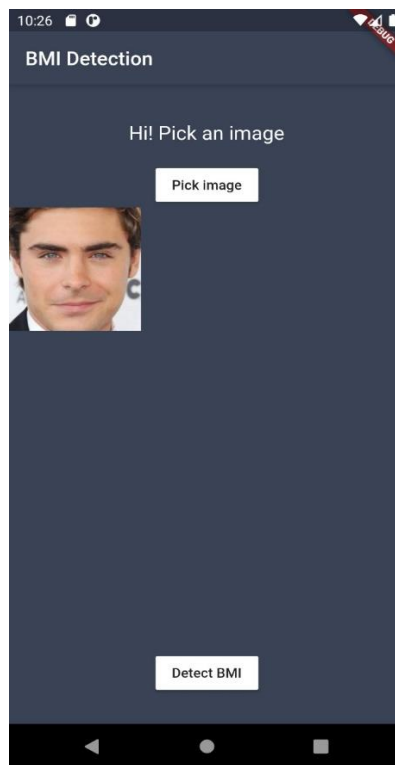


Figure 7.2 Screen to upload images

7.3 BMI PREDICTION

Figure 7.3 depicts the BMI prediction screen of the application.



Figure 7.3: BMI prediction screen

7.4 ACCURACY

Mean Square Error (MSE) is the most commonly used regression loss function. MSE is the sum of squared distances between our target variable and predicted values.

The loss function (Mean Square Error) is used to indicate how far predictions deviate from the target values.

```
Epoch 91/100
10/10 [=====] - 0s 29ms/step - loss: 3.1107 - val_loss: 8.7384
Epoch 92/100
10/10 [=====] - 0s 29ms/step - loss: 3.8070 - val_loss: 9.8703
Epoch 93/100
10/10 [=====] - 0s 27ms/step - loss: 3.3557 - val_loss: 12.2715
Epoch 94/100
10/10 [=====] - 0s 30ms/step - loss: 4.3436 - val_loss: 7.8959
Epoch 95/100
10/10 [=====] - 0s 30ms/step - loss: 3.7010 - val_loss: 12.3183
Epoch 96/100
10/10 [=====] - 0s 29ms/step - loss: 5.1107 - val_loss: 8.4285
Epoch 97/100
10/10 [=====] - 0s 28ms/step - loss: 5.9451 - val_loss: 11.3246
Epoch 98/100
10/10 [=====] - 0s 28ms/step - loss: 4.9032 - val_loss: 10.1510
Epoch 99/100
10/10 [=====] - 0s 29ms/step - loss: 5.2547 - val_loss: 13.0774
Epoch 100/100
10/10 [=====] - 0s 28ms/step - loss: 4.0521 - val_loss: 12.4113
```

Figure 7.4. Final epochs of the estimation model

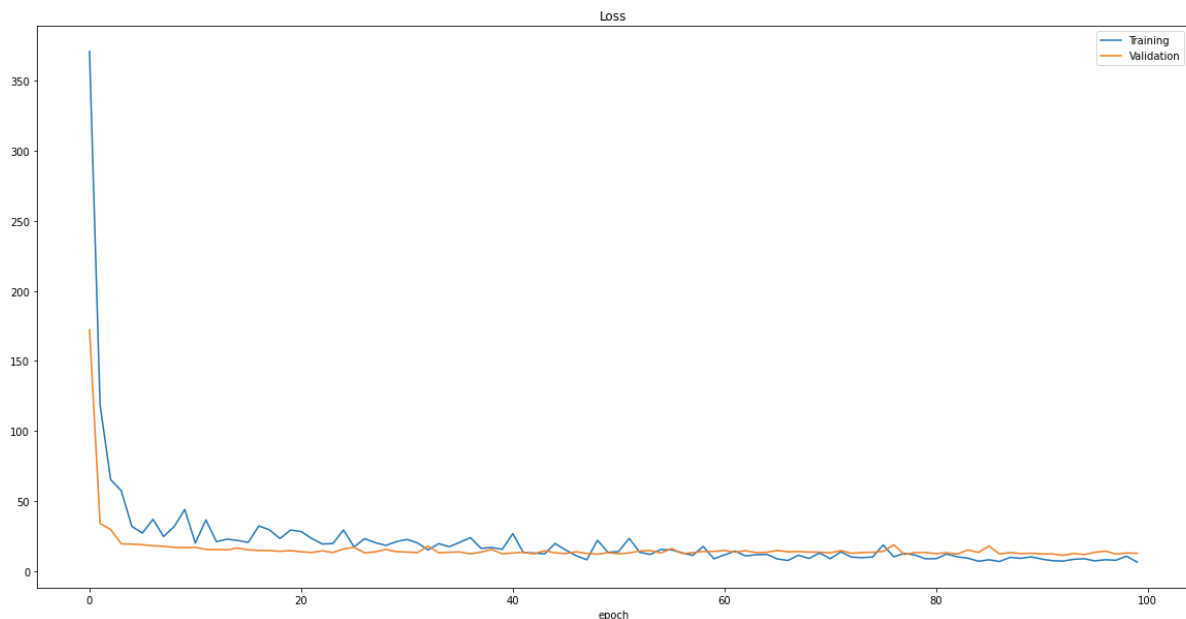


Figure 7.5 Training and validation loss

According to the loss graph, it can be seen that the model neither under-fits nor over-fits the data as the training and validation losses are low and very close to each other. The model was trained with the primary goal of obtaining the most generalised fit over the dataset of 1026 images and thus keeping the differences in the training and validation accuracy within a 10% percent variation whilst not compromising on the accuracy levels and maintaining them above 85%. From the loss values, the **accuracy is calculated to be 95.9479%** for the training of the estimator model on the whole of the dataset and the **validation accuracy is close to 87.5887%**.

CHAPTER 8

CONCLUSION

This deep-learning project will be available as a mobile application targeted at solving the problem of predicting the BMI of a person effectively using face recognition and regression techniques. These models have been in the server side connected to the application for an efficient prediction every time an image input is provided. With this application the user will be able to predict the BMI with an accuracy of 87.58%.

The following can be considered for future improvements upon the existing project:

- Training of the model on many more incorporated images of people. Also including images of people from diverse ethnicities ensures worldwide compatibility.
- Training the model with images of the same person over time towards capturing the progressive change of the face.
- Training of the model with images from different angles especially training with the pictures of the same person with lateral angles of the face.
- Hosting the model on a web server rather than local server as only local devices like emulators on the same PC will have access to the local server's port of communication.
- Creating this application as a plugin/library that can be released to the flutter store or to the python open source community. This can then be incorporated in health apps for seamlessly integration of BMI estimation with fitness tracking.

REFERENCES

1. Min Jiang, Guodong Guo, Guowang Mu, Visual BMI estimation from face images using a label distribution-based method, *Computer Vision and Image Understanding*, Volumes 197–198, 2020
2. Min Jiang, Yuanyuan Shang, Guodong Guo, On visual BMI analysis from facial images, *Image and Vision Computing*, Volume 89, 2019, Pages 183-196, ISSN 0262-8856.
3. Antitza Dantcheva, François Bremond, Piotr Bilinski. Show me your face and I will tell you your height, weight and body mass index. *International Conference on Pattern Recognition (ICPR)*, Aug 2018, Beijing, China.
4. Wen, L., Guo, G., 2013. A computational approach to body mass index prediction from face images. *Image Vis. Comput.* 31 (5), 392–400.
5. Kocabey, E., Camurcu, M., Ofli, F., Aytar, Y., Marin, J., Torralba, A., Weber, I., 2017. Face-to-BMI: using computer vision to infer body mass index on social media. In: 11th International AAAI Conference on Web and Social Media.
6. B. J. Lee, J. H. Do, J. K. Kim, A classification method of normal and overweight females based on facial features for automated medical applications, *Journal of Biomedicine and Biotechnology*
7. Wolffhechel K, Hahn AC, Jarmer H, Fisher CI, Jones BC, et al. (2015) Testing the Utility of a Data-Driven Approach for Assessing BMI from Face Images. *PLOS ONE* 10(10): e0140347.

APPENDIX

FLUTTER APPLICATION

Main.dart

```
import 'package:flutter/material.dart';
import 'Pages/login.dart';
import 'Pages/Primary.dart';
import 'Pages/Final_Display.dart';

void main() => runApp(MainPage());

class MainPage extends StatefulWidget {
  @override
  State<StatefulWidget> createState() {
    return _MainPageState();
  }
}

class _MainPageState extends State<MainPage> {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'BMI',
      theme: ThemeData(
        primaryColor: Color.fromRGBO(58, 66, 86, 1.0),
        accentColor: Colors.white,
        buttonColor: Colors.white,
        canvasColor: Color.fromRGBO(58, 66, 86, 1.0),
        fontFamily: 'Rubik'),
      routes: {
        '/Page2': (BuildContext context) => Primary(),
        '/Page3': (BuildContext context) => FinalDisplay(),
      },
      // home: _isLoggedIn ? Primary(): Login(),
      home: Login();
    )
  }
}
```

login.dart

```
import 'package:flutter/material.dart';

class Login extends StatefulWidget {
  @override
  State<StatefulWidget> createState() {
    return _LoginState();
  }
}

class _LoginState extends State<Login> {
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();

  Widget createRollField() {
    return TextFormField(
      validator: (value) {
        if (value.length < 4) {
          return 'Enter a valid ID';
        }
      },
      style: TextStyle(color: Colors.white),
    );
  }
}
```

```

decoration: InputDecoration(
  hintText: "User ID",
  hintStyle: TextStyle(color: Colors.white),
  focusedBorder: OutlineInputBorder(
    borderRadius: BorderRadius.all(Radius.circular(25)),
    borderSide: BorderSide(width: 2.5, color: Colors.white),
  ),
  disabledBorder: OutlineInputBorder(
    borderRadius: BorderRadius.all(Radius.circular(25)),
    borderSide: BorderSide(width: 2.5, color: Colors.white),
  ),
  enabledBorder: OutlineInputBorder(
    borderRadius: BorderRadius.all(Radius.circular(25)),
    borderSide: BorderSide(width: 2.5, color: Colors.white),
  ),
  border: OutlineInputBorder(
    borderRadius: BorderRadius.all(Radius.circular(25)),
    borderSide: BorderSide(
      width: 2.5,
    ),
  ),
  errorBorder: OutlineInputBorder(
    borderRadius: BorderRadius.all(Radius.circular(25)),
    borderSide: BorderSide(width: 2.5, color: Colors.white),
  ),
  focusedErrorBorder: OutlineInputBorder(
    borderRadius: BorderRadius.all(Radius.circular(25)),
    borderSide: BorderSide(width: 2.5, color: Colors.white),
  ),
),
);
}

Widget createPasswordField() {
  return TextFormField(
    validator: (value) {
      print("2");
      if (value.length < 5) {
        return 'Invalid Size of Password';
      }
    },
    obscureText: true,
    style: TextStyle(color: Colors.white),
    decoration: InputDecoration(
      hintText: "Password",
      hintStyle: TextStyle(color: Colors.white),
      focusedBorder: OutlineInputBorder(
        borderRadius: BorderRadius.all(Radius.circular(25)),
        borderSide: BorderSide(width: 2.5, color: Colors.white),
      ),
      disabledBorder: OutlineInputBorder(
        borderRadius: BorderRadius.all(Radius.circular(25)),
        borderSide: BorderSide(width: 2.5, color: Colors.white),
      ),
      enabledBorder: OutlineInputBorder(
        borderRadius: BorderRadius.all(Radius.circular(25)),
        borderSide: BorderSide(width: 2.5, color: Colors.white),
      ),
      border: OutlineInputBorder(
        borderRadius: BorderRadius.all(Radius.circular(25)),
        borderSide: BorderSide(
          width: 2.5,

```



```

    )),
    errorBorder: OutlineInputBorder(
      borderRadius: BorderRadius.all(Radius.circular(25)),
      borderSide: BorderSide(width: 2.5, color: Colors.white),
    ),
    focusedErrorBorder: OutlineInputBorder(
      borderRadius: BorderRadius.all(Radius.circular(25)),
      borderSide: BorderSide(width: 2.5, color: Colors.white),
    ),
  ),
);
}

Widget createButtonLogin() {
  return ButtonTheme(
    minWidth: 115,
    height: 50,
    child: RaisedButton(
      shape: new RoundedRectangleBorder(
        borderRadius: new BorderRadius.circular(25),
        side: BorderSide(color: Colors.white)),
      onPressed: () {
        if (_formKey.currentState.validate()) {
          _formKey.currentState.save();
          print("Executed");
          Navigator.pushReplacementNamed(context, '/Page2');
        }
      },
      color: Colors.white,
      textColor: Colors.black,
      child: Text("LOG IN".toUpperCase(), style: TextStyle(fontSize:
20)),
    ),
  );
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Color.fromRGBO(58, 66, 86, 1.0),
    body: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      crossAxisAlignment: CrossAxisAlignment.center,
      children: <Widget>[
        Container(
          child: Center(
            child: Image.asset(
              'assets/asset_1.png',
              height: 85,
              width: 85,
            ),
          ),
        ),
        Container(
          margin: EdgeInsets.only(top: 20),
          child: Center(
            // child: Text(
            //   'ESTIMATION',
            //   style: TextStyle(
            //     color: Colors.white,
            //     fontSize: 30,

```

```

        //          fontWeight: FontWeight.w500),
        // ),
    ),
    Container(
      margin: EdgeInsets.only(top: 60, left: 15, right: 15),
      child: Form(
        key: _formKey,
        child: Column(
          children: <Widget>[
            createRollField(),
            Container(margin: EdgeInsets.all(20)),
            createPasswordField(),
            Container(margin: EdgeInsets.all(40)),
            createButtonLogin()
          ],
        ),
      ),
    ),
    // _isLoading == true ? CustomLoading() : Container(),
  ]),
);
}
}

```

Primary.dart

```

import 'dart:convert';

import 'package:bmi_estimation/Pages/Final_Display.dart';
import 'package:flutter/material.dart';
import 'package:multi_image_picker/multi_image_picker.dart';
import 'dart:io';
import 'package:http/http.dart' as http;
import 'package:flutter_absolute_path/flutter_absolute_path.dart';

class Primary extends StatefulWidget {
  @override
  State<StatefulWidget> createState() {
    return _PrimaryState();
  }
}

class _PrimaryState extends State<Primary> {
  List<Asset> images = List<Asset>();
  List<File> imageFiles = List<File>();
  String _error = 'No Error Dected';
  @override
  void initState() {
    super.initState();
  }

  Widget buildGridView() {
    return GridView.count(
      crossAxisCount: 3,
      children: List.generate(images.length, (index) {
        Asset asset = images[index];

        return AssetThumb(
          asset: asset,
          width: 300,
          height: 300,
        );
      });
  }
}

```

```

    })),
  );
}

Future<void> loadAssets() async {
  List<Asset> resultList = List<Asset>();
  List<File> resultFiles = List<File>();
  String error = 'No Error Dectected';

  try {
    resultList = await MultiImagePicker.pickImages(
      maxImages: 1,
      enableCamera: true,
      selectedAssets: images,
      cupertinoOptions: CupertinoOptions(takePhotoIcon: "chat"),
      materialOptions: MaterialOptions(
        actionBarColor: "#abcdef",
        actionBarTitle: "Example App",
        allViewTitle: "All Photos",
        useDetailsView: false,
        selectCircleStrokeColor: "#000000",
      ),
    );
  } on Exception catch (e) {
    error = e.toString();
  }

  resultList.forEach((element) async {
    final filePath =
      await FlutterAbsolutePath.getAbsolutePath(element.identifier);
    File temp = File(filePath);
    if (temp.existsSync()) {
      resultFiles.add(temp);
      print("HERE");
      print(temp.path);
    }
  });
  // If the widget was removed from the tree while the asynchronous
platform
  // message was in flight, we want to discard the reply rather than
calling
  // setState to update our non-existent appearance.
  if (!mounted) return;

  setState(() {
    images = resultList;
    imageFiles = resultFiles;
    _error = error;
  });
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text("BMI Detection"),
    ),
    body: Column(
      mainAxisAlignment: MainAxisAlignment.max,
      //mainAxisAlignment: MainAxisAlignment.center,
      crossAxisAlignment: CrossAxisAlignment.center,

```

```

children: <Widget>[
  Container(
    margin: EdgeInsets.only(top: 40),
    child: Center(
      child: _error == "No Error Dected"
        ? Text(
            "Hi! Pick an image",
            style: TextStyle(color: Colors.white, fontSize: 20),
          )
        : Text(
            '$_error',
            style: TextStyle(color: Colors.white, fontSize: 15),
          ),
    ),
  ),
  Container(
    margin: EdgeInsets.only(top: 20),
    child: RaisedButton(
      child: Text("Pick image"),
      onPressed: loadAssets,
    ),
  ),
  Expanded(
    child: buildGridView(),
  ),
  images.length > 0
    ? Container(
        margin: EdgeInsets.only(bottom: 30),
        child: RaisedButton(
          child: Text("Detect BMI"),
          onPressed: () async {
            //Upload to Flask Server
            var url = 'http://192.168.0.110:5000/test';
            Map<String, String> headers = {
              "Content-type": "application/json"
            };
            var request =
              http.MultipartRequest("POST", Uri.parse(url));
            var pic = await http.MultipartFile.fromPath(
              "image1", imageFiles[0].path);
            // var pic2 = await http.MultipartFile.fromPath(
            //   "image2", imageFiles[1].path);
            // var pic3 = await http.MultipartFile.fromPath(
            //   "image3", imageFiles[2].path);
            request.files.add(pic);
            // request.files.add(pic2);
            // request.files.add(pic3);
            var response = await request.send();
            //Get the response from the server
            var responseData = await response.stream.toBytes();
            var responseString =
String.fromCharCode(responseData);
            print(responseString);
            var result = json.decode(responseString);
            // String result = jsonDecode(responseString);
            var resultString = result["prediction"];
            print(resultString);
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) =>

```

```

        FinalDisplay(bmi: resultString)));
    },
  ),
)
: Container(),
],
),
);
}
}

Finaldisplay.dart import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';

class FinalDisplay extends StatefulWidget {
  String bmi = "";
  FinalDisplay({Key key, @required this.bmi}) : super(key: key);
  @override
  State<StatefulWidget> createState() {
    return _FinalDisplayState();
  }
}

class _FinalDisplayState extends State<FinalDisplay> {
  Color textColor;
  double decimal;
  @override
  void initState() {
    decimal = double.parse(widget.bmi);
    if (decimal >= 18.5 && decimal < 25) {
      setState(() {
        textColor = Colors.green;
      });
    } else if (decimal < 18.5) {
      setState(() {
        textColor = Colors.yellow;
      });
    } else if (decimal >= 25) {
      setState(() {
        textColor = Colors.red;
      });
    }
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('BMI Result'),
      ),
      body: Column(
        children: [
          Row(
            children: [
              Container(
                margin: EdgeInsets.only(top: 40, left: 20),
                child: Text(
                  'Your BMI is',
                  style: GoogleFonts.montserrat(
                    fontSize: 30,
                    fontWeight: FontWeight.bold,

```

```

        color: Colors.white),
    ),
  ),
  Container(
    margin: EdgeInsets.only(top: 40, left: 20),
    child: Text(
      double.parse(widget.bmi).toStringAsFixed(2),
      style: GoogleFonts.montserrat(
        fontSize: 50,
        fontWeight: FontWeight.bold,
        color: textColor),
    ),
  ),
],
),
Row(
  children: [
    Container(
      margin: EdgeInsets.only(left: 20, top: 50),
      child: Text(
        "BMI 18.5 - 24.9 is",
        style: GoogleFonts.montserrat(
          fontSize: 20,
          fontWeight: FontWeight.w400,
          color: Colors.white),
      ),
    ),
    Container(
      margin: EdgeInsets.only(left: 10, top: 50),
      child: Text(
        "NORMAL",
        style: GoogleFonts.montserrat(
          fontSize: 20,
          fontWeight: FontWeight.bold,
          color: Colors.green),
      ),
    ),
  ],
),
Row(
  children: [
    Container(
      margin: EdgeInsets.only(left: 20, top: 50),
      child: Text(
        "BMI Below 18.5 is",
        style: GoogleFonts.montserrat(
          fontSize: 20,
          fontWeight: FontWeight.w400,
          color: Colors.white),
      ),
    ),
    Container(
      margin: EdgeInsets.only(left: 10, top: 50),
      child: Text(
        "UNDERWEIGHT",
        style: GoogleFonts.montserrat(
          fontSize: 20,
          fontWeight: FontWeight.bold,
          color: Colors.yellow),
      ),
    ),
  ],
),

```

```

    ],
),
Row(
    children: [
        Container(
            margin: EdgeInsets.only(left: 20, top: 50),
            child: Text(
                "BMI Above 25 is",
                style: GoogleFonts.montserrat(
                    fontSize: 20,
                    fontWeight: FontWeight.w400,
                    color: Colors.white),
            ),
        ),
        Container(
            margin: EdgeInsets.only(left: 10, top: 50),
            child: Text(
                "OVERWEIGHT",
                style: GoogleFonts.montserrat(
                    fontSize: 20,
                    fontWeight: FontWeight.bold,
                    color: Colors.red),
            ),
        ),
    ],
),
),
),
);
}
}

```

Backend.py

```

from flask import Flask, jsonify, request
import pandas as pd
import numpy as np
import joblib
import traceback
import sys
from flask_restful import reqparse

# Deep learning libraries
import numpy as np
import pandas as pd
import tensorflow as tf
import cv2
import dlib
from tensorflow import keras
from tensorflow.keras.models import load_model
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import
ZeroPadding2D, Convolution2D, MaxPooling2D
from tensorflow.keras.layers import
Dense, Dropout, Softmax, Flatten, Activation, BatchNormalization
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.applications.imagenet_utils import preprocess_input
import tensorflow.keras.backend as K

app = Flask(__name__)

```

```

app.config['MAX_CONTENT_LENGTH'] = 50 * 1024 * 1024
@app.route('/', methods=['GET'])
def hello():
    return 'Home'

@app.route('/test', methods=['POST'])
def test():
    try:
        # imagefile = request.files('file_field', '')
        # print(imagefile)
        f = request.files['image1']
        # f2 = request.files['image2']
        # f3 = request.files['image3']
        f.save(f.filename)
        # f2.save(f2.filename)
        # f2.save(f2.filename)
        print(f.filename)

        # FACE DETECTOR MODEL
        model = Sequential()
        model.add(ZeroPadding2D((1,1),input_shape=(224,224, 3)))
        model.add(Convolution2D(64, (3, 3), activation='relu'))
        model.add(ZeroPadding2D((1,1)))
        model.add(Convolution2D(64, (3, 3), activation='relu'))
        model.add(MaxPooling2D((2,2), strides=(2,2)))
        model.add(ZeroPadding2D((1,1)))
        model.add(Convolution2D(128, (3, 3), activation='relu'))
        model.add(ZeroPadding2D((1,1)))
        model.add(Convolution2D(128, (3, 3), activation='relu'))
        model.add(MaxPooling2D((2,2), strides=(2,2)))
        model.add(ZeroPadding2D((1,1)))
        model.add(Convolution2D(256, (3, 3), activation='relu'))
        model.add(ZeroPadding2D((1,1)))
        model.add(Convolution2D(256, (3, 3), activation='relu'))
        model.add(ZeroPadding2D((1,1)))
        model.add(Convolution2D(256, (3, 3), activation='relu'))
        model.add(MaxPooling2D((2,2), strides=(2,2)))
        model.add(ZeroPadding2D((1,1)))
        model.add(Convolution2D(512, (3, 3), activation='relu'))
        model.add(ZeroPadding2D((1,1)))
        model.add(Convolution2D(512, (3, 3), activation='relu'))
        model.add(ZeroPadding2D((1,1)))
        model.add(Convolution2D(512, (3, 3), activation='relu'))
        model.add(MaxPooling2D((2,2), strides=(2,2)))
        model.add(ZeroPadding2D((1,1)))
        model.add(Convolution2D(512, (3, 3), activation='relu'))
        model.add(ZeroPadding2D((1,1)))
        model.add(Convolution2D(512, (3, 3), activation='relu'))
        model.add(MaxPooling2D((2,2), strides=(2,2)))
        model.add(Convolution2D(4096, (7, 7), activation='relu'))
        model.add(Dropout(0.5))
        model.add(Convolution2D(4096, (1, 1), activation='relu'))
        model.add(Dropout(0.5))
        model.add(Convolution2D(2622, (1, 1)))
        model.add(Flatten())
        model.add(Activation('softmax'))
        #change the weights file location
        model.load_weights('vgg_face_weights.h5')

```



```

    vgg_face=Model(inputs=model.layers[0].input,outputs=model.layers[-
2].output)
        #cropping the face
        # Load cnn_face_detector with 'mmod_face_detector'

    dnnFaceDetector=dlib.cnn_face_detection_model_v1("mmod_human_face_d
etector.dat")
        # Load image
        img=cv2.imread(f.filename)
        # print(img)
        # Convert to gray scale
        gray=cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        # Find faces in image
        rects=dnnFaceDetector(gray,1)
        left,top,right,bottom=0,0,0,0
        # For each face 'rect' provides face location in image as
pixel location
        for (i,rect) in enumerate(rects):
            left=rect.rect.left() #x1
            top=rect.rect.top() #y1
            right=rect.rect.right() #x2
            bottom=rect.rect.bottom() #y2
            width=right-left
            height=bottom-top
            # Crop image
            img_crop=img[top:top+height,left:left+width]
            #save crop image with person name as image name

cv2.imwrite('C:\\Users\\Tejas\\Desktop\\Sample.png',img_crop)
        #image encoding
        image_path='sample.png'

    picture=load_img(image_path,target_size=(224,224),color_mode='rgb')
        picture=img_to_array(picture)
        img_encode=[]
        # print(image_path)
        picture=np.expand_dims(picture,axis=0)
        picture=preprocess_input(picture)
        img_encode=vgg_face(picture)
        encodings=[]
        encodings.append(np.squeeze(K.eval(img_encode)).tolist())
        newArr = np.reshape(encodings,(1,2622))
        print(newArr.shape)
        finalModel = load_model("bmiEstimatorModel.h5")
        finalVal = finalModel.predict(newArr)
        print(finalVal[0])
        return jsonify({'prediction': str(finalVal[0][0])})
except Exception as err:
    return jsonify({'trace': traceback.format_exc()})

    return ('Oops')

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0',port = 5000)

```

PLAGIARISM SCAN REPORT

Words 581 Date Novem

Characters 3781 Exclude URL

7%

Plagiarism

93%

Unique

2

Plagiarized Sentences

2

U

S

Content Checked For Plagiarism

INTRODUCTION CHAPTER 1

CHAPTER 1

introduction

1.1. INTRODUCTION

The human face exhibits information pertaining to identity, a person's disposition, demeanour, as well as to attributes such as gender, age, ethnicity. From the perspective of biometrics, emphasis has predominantly been placed on facial recognition. Face images contain much information, such as identity, gender, age, weight, etc. Decoding facial cues has attracted much attention from sociologists and computer scientists. Perceptions of people's weight based on photographs of their face is positively correlated with their body mass index. The Body Mass Index is a general body fat indicator widely used in medical research.

The BMI is calculated by using the formula,

It can reveal various health and lifestyle issues. With the advanced algorithms and deep learning models trained on vast and diverse data today, the capabilities of facial image analysis can be extended to estimate personal health characteristics such as BMI by extrapolations from facial biometric data. This project aims to use specified and distinct facial features of a person to accurately predict the BMI of the said person. Machine learning is an area of Artificial Intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Deep learning, where deep neural networks with several nodes and layers are involved, forms a major sub-field of machine learning and Computer Vision has always been a primary area of focus in this domain. The principal aim of facial image analysis in computer vision is to extract valuable information by interpreting perceived electronic data from face images.

1.2. MOTIVATION

Recent research shows that facial fatness is associated with perceived health and is highly correlated with body mass index (BMI). BMI is a general body fat indicator, BMI is widely used in health monitoring and health research. There are close connections between BMI and some diseases, such as cancers, unstable angina and type 2 diabetes and cardiovascular disease etc. So, it has now become common to know the BMI of a person for a health diagnosis. There are many applications to keep track of the BMI but all of them require manual input of BMI. The traditional method of calculating BMI is a tedious process while the proposed solution can be used to estimate the BMI in a much faster way.

1.3. PROBLEM STATEMENT

It is difficult to evaluate the BMI accurately for an individual. Nowadays, a lot of fitness apps ask the user to provide his/her BMI details. The user would have to self-report their body-mass index or use a special device to calculate it. Due to this difficulty, users tend to provide incorrect data when it is needed. This leads to incorrect diagnosis and decreased accuracy of fitness applications.

1.4. OBJECTIVE

The objective of this project is to effectively estimate the BMI of the user with facial features and track BMI changes with the change in facial features over time along with the proper indications to the user in case of extremities such as being underweight or obesity thereby serving as a simple and easy to use BMI tracker.

1.5. SCOPE AND APPLICATIONS

The scope of this system is to reduce the difficulty in estimating BMI in several areas. Some of them are:

* The system can be integrated with fitness applications to allow users to estimate BMI by using just a selfie.

* The system can be used by individual users to keep track of their personal BMI changes with changes in their facial appearance over time.

2

Sources

Machine Learning | Introduction to ML - YouTube

Machine learning is an application of artificial intelligence (AI) that provides systems with the ability to automatically learn and improve from experience without being explicitly programmed.

https://www.youtube.com/watch?v=N7sx9_nX8Ng

Body Mass Index, Metabolic Syndrome, and Risk of Type 2 Diabetes...

... as a body fat indicator, bmi is widely used in health monitoring and health research. there are close connections between bmi and some diseasesrecent research shows that facial adiposity is associated with bmi prediction. in this work, we investigate the problem of visual bmi estimation from...

https://www.researchgate.net/publication/7048325_Body_Mass_Index_Metabolic_Syndrome_and_Risk_of_Type_2_Diabetes_or_Cardiovascular_Disease

PLAGIARISM SCAN REPORT

Words 339 Date November 19,2020

Characters 2364 Exclude URL

12%

Plagiarism

88%

Unique

2

Plagiarized
Sentences

15

Unique Sentences

Content Checked For Plagiarism

CHAPTER 2 SYSTEM STUDY

CHAPTER 2

SYSTEM STUDY

This chapter discusses about the knowledge that is gained from the following materials which include substantive findings as well as theoretical contributions to BMI analysis from facial images.

Jiang et al. [1] used a two-stage learning framework, which consists of BMI-related facial features learning and BMI estimator learning. The BMI-related face model was learned based on a pre-trained deep face model. Then two different strategies were analyzed for modeling the BMI labels with probability distributions and a projection optimization was achieved by maximizing the correlation between the facial features and the assigned labels. Then the BMI estimator was learned from the projected features and assigned labels.

Jiang et al. [2] studied the visual BMI estimation problem systematically based on the facial representation or feature extraction. According to the inherent properties of representations, they grouped them into two types: geometric based and deep learning based. They experimented with two of the existing approaches (VGG-Face and PIGF), and five other facial approaches: PF, PIGF+PF, LightCNN, Centerloss and Arcface.

Dantcheva et al. [3] explored the possibility of estimating height, weight, and BMI from single-shot facial images by using a regression method based on a 50-layer ResNet architecture. They assembled a dataset call VIP_Attribute dataset which consists of 1026 subjects to facilitate the study.

Wen and Guo [4] proposed a computational method for automatically predicting BMI from 2D face images. This was based on the MORPH-II dataset. The study explored handcrafted features for BMI-estimation were normalized and finally subjected to support vector regression.

Kocabey et al. [5] employed the pre-trained VGG-Face model to extract facial representation for BMI estimation. Then a support vector regression model was learned to map the facial representation to predicted BMIs. The above two works treated BMI prediction as a regression problem.

Lee et al. [6] proposed in a prediction method of normal and overweight females based on BMI using geometrical facial features only. The features, measured on 2D images, include Euclidean distances, angles and face areas defined by selected soft-tissue landmarks.

Sources	Similarity
Multi-Scale Rotation-Invariant Convolutional Neural Networks ... The above two works treated BMI prediction as a regression problem. Recently, convolutional neural networks (CNN) have shown promising performance in ...	7%

https://www.researchgate.net/publication/315503397_Multi-Scale_Rotation-Invariant_Convolutional_Neural_Networks_for_Lung_Texture_Classification	
<p>All points in a facial image for feature extraction ((a): points and...</p> <p>The features, measured on 2D images, include Euclidean distances, angles and face areas defined by selected soft-tissue landmarks. ... Wize Mirror - a smart, ...</p> <p>https://www.researchgate.net/figure/All-points-in-a-facial-image-for-feature-extraction-a-points-and-areas-in-frontal_fig1_230735789</p>	6%

PLAGIARISM SCAN REPORT

Words 526 Date November 19,2020

Characters 3695 Exclude URL

4%

Plagiarism

96%

Unique

1

Plagiarized
Sentences

23

Unique Sentences

Content Checked For Plagiarism

CHAPTER 3

SYSTEM ANALYSIS

CHAPTER 3

SYSTEM ANALYSIS

This chapter describes the system analysis which is conducted for the purpose of studying a system or its parts in order to identify its objectives.

3.1 FUNCTIONAL REQUIREMENTS

1. Capturing of image of the user

* Enabling users to click pictures using the camera of the mobile device used.

2. Allowing user to upload images

* Enabling users to upload images directly from their gallery.

3. Face Detection Algorithm

* Trained Deep Learning Model to detect the outlines of the human face from the user image.

4. Geometrical facial feature extraction from detected face

* Trained Deep Learning Model that extracts geometrical facial features such as height to width ratio, circumference of eyes, and dimensions of lips, nose, cheeks and the chin.

5. Trained BMI estimator model

* A trained BMI estimator model with optimization features to calculate the BMI of the user from the extracted facial features.

6. Detect BMI and classify users in real time

* Classifying users into Categories of BMI from their image data in real time.

3.2 NON-FUNCTIONAL REQUIREMENTS

The non-functional requirements are as follows:

3.2.1 Usability Requirements

The system should be user friendly and doesn't require any guidance. In other words, the application has to be as simple as possible, so its users shall use it easily.

3.2.2 Reliability Requirements

The system should not have any unexpected failure and must be reliable at least 98% of the time. In order to avoid any failure's occurrence, the specifications have been respected and followed correctly. The failure of the system is based on the accuracy of predicting the results.

3.2.3 Efficiency Requirements

The system response time should be adequate and sufficient enough, in order to increase the efficiency of the system. The application should be compatible with different versions of the libraries used.

3.3 EXPERIMENTAL SETUP

3.3.1 Hardware Requirements

The following are the details of the workstation used for implementation and evaluation.

* Processor - Two core processor

* RAM - 1 GB

* Disk Storage - 2 GB

The following hardware requirements are needed for deployment,

* Smart Phone – with Android OS 6.0 and above or iOS 11 and above

* RAM – 1 GB

* Disk Storage – 20 MB

3.3.2 Software Requirements

The following are the software requirements of the system:

1. Google Colaboratory

2. Jupyter Notebook

3. Python 3.7.6

4. Numpy, Pandas, Matplotlib, OpenCV libraries for Python

5. TensorFlow, ScikitLearn and PyTorch Machine Learning libraries for Python

3.4 FEASIBILITY ANALYSIS

A feasibility study is used to determine the practicality of an idea or proposed plan.

3.4.1 Economic Feasibility

As this project is implemented using only open source distributions of python and open source machine learning and image processing libraries, it is economically feasible.

3.4.2 Technical Feasibility

Python has numerous features like third-party modules (PyPi), extensive support for libraries, simplicity, enhanced text processing capabilities all of which makes it a viable option in terms of speed and productivity.

3.4.3 Operational Feasibility

All the functions performed by the system are valid and without conflict. All the functions and constraints specified in the requirements are operational.

6

CHAPTER 3

SYSTEM ANALYSIS

6

Sources	Similarity
<p>OCR report synopsis Optical Character Recognition Mobile App</p> <p>in other words, the application has to be as simple as possible, so its users shall use it easily.why the time required for this application to response to its user's actions has been managed and controlled. but in order to maintain the performance of the application, the user must follow the...</p> <p>https://www.scribd.com/document/432007597/OCR-report-synopsis</p>	9%

PLAGIARISM SCAN REPORT

Words 672 Date November 20, 2020

Characters 3325 Exclude URL

0%

Plagiarism

100%

Unique

0

Plagiarized
Sentences

25

Unique Sentences

Content Checked For Plagiarism

CHAPTER 4

SYSTEM DESIGN

This chapter describes the design of the Visual BMI estimator. It talks about the functionalities of the Visual BMI estimator after analyzing the requirements.

4.1. USE CASE DIAGRAM:

Fig.4.1 shows the use case diagram for the Visual BMI estimator.

Figure 4. SEQ Figure_4. * ARABIC 1: Use case diagram

4.2. ACTIVITY DIAGRAM:

Fig.4.2 shows the activity diagram for the Visual BMI estimator.

Figure 4. SEQ Figure_4. * ARABIC 2: Activity diagram

4.3. ARCHITECTURE DIAGRAM:

Fig.4.3 shows the architecture diagram for the Visual BMI estimator. The description of the diagram is as follows,

The user is asked to upload his or her image to the system

Now, the pre-processing of the image is done, which includes

Converting the image to gray-scale

Cropping only the face from the image

Now the data is passed through the modified VGG-face deep learning model to obtain the facial feature embedding

Then the embeddings are given as input to the fully connected deep learning model.

Then based on the trained model, the BMI is estimated.

Figure 4. SEQ Figure_4. * ARABIC 3 Architecture diagram

Figure 4. SEQ Figure_4. * ARABIC 4 Modified VGG-Face model

Description of the layers used in the modified VGG-Face model depicted in figure 4.4

Zero padding 2D

This layer can add rows and columns of zeros at the top, bottom, left, and right sides of an image tensor.

Conv2D

This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs.

Max pooling 2D

Downsamples the input representation by taking the maximum value over the window defined by pool size for each dimension along the features axis.

The window is shifted by strides in each dimension.

Dropout

The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting.

Inputs not set to 0 are scaled up by $1/(1 - \text{rate})$ such that the sum over all inputs is unchanged.

Flatten

Flattens the input.

Does not affect the batch size.

Figure 4. SEQ Figure_4. * ARABIC 5:BMI estimation model

Description of the layers used in the modified VGG-Face model depicted in figure 4.5

Dense layer

Dense implements the operation: $\text{output} = \text{activation}(\text{dot}(\text{input}, \text{kernel}) + \text{bias})$ where activation is the element-wise activation function passed as the activation argument, kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer (only applicable if use_bias is True).

Dropout

The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting.

Inputs not set to 0 are scaled up by $1/(1 - \text{rate})$ such that the sum over all inputs is unchanged.

ELU activation

Exponential Linear Unit.

The exponential linear unit (ELU) with $\alpha > 0$ is:

x if $x \geq 0$

$\alpha * (\exp(x) - 1)$ if $x < 0$

The ELU hyperparameter α controls the value to which an ELU saturates for negative net inputs. ELUs diminish the vanishing gradient effect.

ELUs have negative values which pushes the mean of the activations closer to zero.

Mean activations that are closer to zero enable faster learning as they bring the gradient closer to the natural gradient.

ELUs saturate to a negative value when the argument gets smaller.

Saturation means a small derivative which decreases the variation and the information that is propagated to the next layer.

4.4. DATASET DESCRIPTION:

VIP_Attribute is a dataset comprising face images assembled to study height, weight, and body mass index (BMI) based on facial images. It has 1026 images of 513 female and 513 male celebrities (mainly actors, singers and athletes) collected from the internet. The images include the frontal pose of the subjects. Co-variates include illumination, expression, image quality, and resolution. Further challenges in this dataset are beautification (e.g., photoshop) of the images and the presence of makeup, plastic surgery, beard, and moustache. The annotations related to the subjects' body weight and height were obtained from websites.

PLAGIARISM SCAN REPORT

Words 445 Date November 20,2020

Characters 2842 Exclude URL

0%

Plagiarism

100%

Unique

0

Plagiarized
Sentences

18

Unique Sentences

Content Checked For Plagiarism

SYSTEM IMPLEMENTATION CHAPTER 5

CHAPTER 5

SYSTEM IMPLEMENTATION

This chapter gives a brief explanation of the implementation of the visual BMI analysis system.

5.1 LOADING THE DATA

- * Read the .csv files containing the BMI information
- * Find the number of images in the data folder
- * Add the column image path to a data frame

Figure 5. 1: Loading the data

5.2 PREPROCESSING OF IMAGE

- * Load cnn_face_detector with 'mmod_face_detector'
- * Load image
- * Convert to grayscale
- * For each face, 'rect' provides face location in the image as a pixel location
- * Save crop image

Figure 5. 2: Preprocessing the data

5.3 FACIAL FEATURE EXTRACTION

- * Define VGG_FACE_MODEL architecture
- * Load the pre-trained weights into the model
- * The model needs not to classify whether the image consists of a face or not, so we remove the final softmax layer in the output to find out only the embeddings.

Figure 5. 3 Facial feature extraction model

5.4 FACIAL FEATURE EXTRACTION FOR IMAGES IN THE DATASET

- * For every single image, we load it with size(224,224), and we find the embedding vector by passing it to the neural network.
- * The embeddings are appended to a data frame, and that is converted to a .csv file for training the next fully connected model

Figure 5. 4 Extracting features to a csv file

5.5 SPLITTING INTO TEST AND TRAIN DATASET

- * The BMI datafile from the dataset is loaded as a data frame.
- * The embeddings.csv file is also loaded as a data frame.
- * Then the train test split is done (70% of data for training and 30% of data for validation)

Figure 5. 5:Splitting into train and test set

5.6 BUILDING THE FULLY CONNECTED ESTIMATOR MODEL

- * With relu, a problem might occur called dead relu that feeds only 0 values to the next layer, and enough number of dead relus will make the loss stagnant and will not decrease with epochs
- * Using elu activation, which has a non zero gradient in the negative section instead of 0 in relu when there were only two dropout layers
- * A meager difference of 0.02 between the training and the validation loss; hence a decently good model is achieved
- * However, the low value of the loss means overfitting. Accordingly, so more dropout layers are added
- * But dropout layers were removed when training loss is significantly greater than validation loss.

Figure 5. 6:Fully connected estimator model

5.7 TRAINING AND VALIDATION

Figure 5. 7:Training and validation

5.8 BACKEND INTEGRATION

- * The deep learning models were run on a local server using the flask API with which the mobile application communicates

Figure 5.8:Backend integration

2

Sources

Similarity

PLAGIARISM SCAN REPORT

Words 445 Date November 19,2020

Characters 3926 Exclude URL

0%

Plagiarism

100%

Unique

0

Plagiarized
Sentences

21

Unique Sentences

Content Checked For Plagiarism

TESTING CHAPTER 6

CHAPTER 6

TESTING

This chapter describes the results of the testing of the modules after implementation.

6.1 UNIT TESTING

The tabulations below contain the summaries of the testing of the individual modules.

6.1.1 IMAGE PREPROCESSING

Test case No.

Inputs

Expected Output

Obtained Output

Test case Acceptance

1

Image with pixel size less than 224x224

Inputs out of range

Inputs out of range

Yes

2

Image with pixel size greater than 224x224

Image Inputs accepted and processed

Image Inputs accepted and processed

Yes

Table 6.1: Summary of testing for Image Preprocessing

6.1.2 FACE DETECTION

Test case No.

Inputs

Expected Output

Obtained Output

Test case Acceptance

1

224x224 pixel size image not containing a human face

Invalid Image Input message

Invalid Image Input message

Yes

2

224x224 pixel size image not containing multiple human faces

Invalid Image Input message

Invalid Image Input message

Yes

3

224x224 pixel size image containing a human face

Face detected and facial feature vector returned

Face detected and facial feature vector returned

Yes

Table 6.2: Summary of unit testing for face detection

6.1.3 BMI ESTIMATION

Test case No.

Inputs

Expected Output

Obtained Output

Test case Acceptance

1

Facial feature vector from face detector for image of person with normal BMI

BMI value between 18.5 to 24.9 and indication for Normal

BMI value between 18.5 to 24.9 and indication for Normal

Yes

2

Facial feature vector from face detector for image of an underweight person

BMI value below 18.5 indication for Underweight

BMI value below 18.5 indication for Underweight

Yes

3

Facial feature vector from face detector for image of an obese person

BMI value above 24.9 and indication for Overweight

BMI value above 24.9 and indication for Overweight

Yes

Table 6.3: Summary of unit testing for BMI estimation

6.2 INTEGRATION TESTING

All the modules passed unit testing. But when they were all integrated together, certain issues had to be resolved. In integrating the machine learning model with the mobile application, the model had to be hosted on the local server and the following errors/issues were identified during this process.

Issue No.

Description

Solution to resolve issue

Completion of resolving issue

1

Incompatibility of machine learning libraries in local machine to the versions with which the model was trained in Google Collaboratory.

Updation of the machine learning libraries – particularly cv2 and tensorflow - in the local machine to the latest versions.

Yes

2

Input dimensions not matching with the input dimensions required by the estimator model

Using image preprocessing functions from keras library to resize the image inputs to the desired dimensions

Yes

3

Passing input image to local server, processing it, estimate BMI and returning the result to the mobile application run on the local emulator takes a significant amount of time

Hosting the model on a powerful web host like Amazon Cloud and using a physical mobile device would increase

the performance.

No

4

Unavailability of access to camera by emulator to facilitate BMI estimation from of live capturing of images of the user.

Hosting the model on a web host would eliminate the need for the emulator to be run on the same machine as local server and a physical mobile device with a camera can be used.

No

Table 6.4: Summary of Integration Testing

24

Sources	Similarity
---------	------------

PLAGIARISM SCAN REPORT

Words 242 Date November 19,2020

Characters 1631 Exclude URL

0%

Plagiarism

100%

Unique

0

Plagiarized
Sentences

11

Unique Sentences

Content Checked For Plagiarism

RESULTS CHAPTER 7

CHAPTER 7

RESULTS

This chapter contains a summary of the results achieved while implementing the system.

7.1 LOGIN:

Figure 7.1 depicts a screenshot of the opening login screen of the application.

Figure 7.1: Placeholder(opening login screen)

7.2 UPLOAD IMAGES:

Figure 7.2 depicts the image upload screen of the application.

Figure 7.2 Screen to upload images

7.3 BMI PREDICTION:

Figure 7.3 depicts the BMI prediction screen of the application.

Figure 7.3: BMI prediction screen

7.4 ACCURACY:

Commonly used regression loss function is mean square error. It is the sum of squared distances between target variable and predicted values. The loss function is used to indicate how far predictions deviate from the target values.

Figure 7.4. Final epochs of the estimation model

Figure 7.5 Training and validation loss

According to the loss graph, it can be seen that the model neither under-fits nor over-fits the data as the training and validation losses are low and very close to each other. The model was trained with the primary goal of obtaining the most generalised fit over the dataset of 1026 images and thus keeping the differences in the training and validation accuracy within a 10% percent variation whilst not compromising on the accuracy levels and maintaining them above 85%. From the loss values, the accuracy is calculated to be 95.9479% for the training of the estimator model on the whole of the dataset and the validation accuracy is close to 87.5887%.

24

Sources

Similarity

PLAGIARISM SCAN REPORT

Words 227 Date November 19,2020

Characters 1472 Exclude URL

0%

Plagiarism

100%

Unique

0

Plagiarized
Sentences

11

Unique Sentences

Content Checked For Plagiarism

CONCLUSION CHAPTER 8

CHAPTER 8

CONCLUSION

This deep-learning project will be available as a mobile application targeted at solving the problem of predicting the BMI of a person effectively using face recognition and regression techniques. These models have been in the server side connected to the application for an efficient prediction every time an image input is provided. With this application the user will be able to predict the BMI with an accuracy of 87.58%.

The following can be considered for future improvements upon the existing project:

- * Training of the model on many more incorporated images of people. Also including images of people from diverse ethnicities ensures worldwide compatibility.
- * Training the model with images of the same person over time towards capturing the progressive change of the face.
- * Training of the model with images from different angles especially training with the pictures of the same person with lateral angles of the face.
- * Hosting the model on a web server rather than local server as only local devices like emulators on the same PC will have access to the local server's port of communication.
- * Creating this application as a plugin/library that can be released to the flutter store or to the python open source community. This can then be incorporated in health apps for seamlessly integration of BMI estimation with fitness tracking.

28

Sources

Similarity