



Mini Project Report

on

"FACE RECOGNITION"

Submitted in Partial Fulfillment for the Award of Degree of

Bachelor of Technology (3rd Sem)

GUIDED BY: - Manikant Kumar

-Sanjoy kumar Mohanta

-Abdul Kayom

SUBMITTED BY: -

HARSH SHARMA 1901227503

MANAS RANJAN BAISWAL 1901227507

MUSKAAN 1901227510

VIKASH MISHRA 1901227550



CERTIFICATE OF APPROVAL

This is to certify that we examined and approved the mini project 3rd semester in Electronics and Telecommunication Engineering entitled FACIAL RECOGNITION submitted by

| | |
|----------------------|------------|
| HARSH SHARMA | 1901227503 |
| MANAS RANJAN BAISWAL | 1901227507 |
| MUSKAAN | 1901227510 |
| VIKASH MISHRA | 1901227550 |

to C. V. Raman College of Engineering, Bhubaneswar. We here by accord our approval of it as a mini project work carried out and presented in a manner required for its acceptance for the partial fulfillment for the bachelor's degree of Technology in Electronics and Telecommunication Engineering for which it has been submitted. This approval does not necessarily endorse or accept every statement made, opinion expressed, or conclusions drawn as recorded in this mini project, it only signifies the acceptance of the mini project for the purpose it has been submitted.

(Project guide)

(External)

(HOD)

Acknowledgment

We would like to express our immense gratitude and sincere thanks to mini project guide “Manikant Kumar, Sanjoy Kumar Mohanta and Abdul Kayom” whose co-operative guidance has helped us in successful completion of this mini-project on “FACE RECOGNITION”.

| | |
|-----------------------------|-------------------|
| HARSH SHARMA | 1901227503 |
| MANAS RANJAN BAISWAL | 1901227507 |
| MUSKAAN | 1901227510 |
| VIKASH MISHRA | 1901227550 |



Department of ETC
C.V. Raman Global University, Bhubaneswar

CERTIFICATE

*This is to certify that the work in this project report entitled “FACE
RECOGNITION” submitted by*

| | |
|----------------------|------------|
| HARSH SHARMA | 1901227503 |
| MANAS RANJAN BAISWAL | 1901227507 |
| MUSKAAN | 1901227510 |
| VIKASH MISHRA | 1901227550 |

*in partial fulfillment of the requirements for the award of Bachelor of
Technology in Electronics & Telecommunication Engineering is carried out by
them under my supervision and guidance.*

Table of Contents

| | |
|--|--|
| List of Figures..... | |
| Abstract..... | |
| Chapter 1: Introduction..... | |
| Chapter 2: Requirement and Analysis..... | |
| Chapter 3: System Design | |
| Chapter 4: Source Code..... | |
| Chapter 5: Implementation and testing..... | |
| Chapter 6: Results and discussion..... | |
| Chapter 7: Conclusion..... | |
| Reference | |

List of Figures

Fig 1:-Block Diagram

Fig 2:-Neural Network

Fig 3:-Output of Project.....

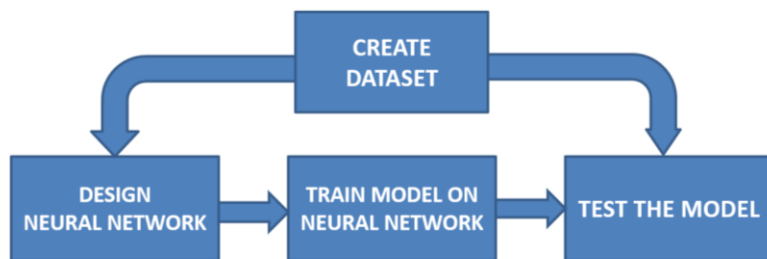


FIGURE 1

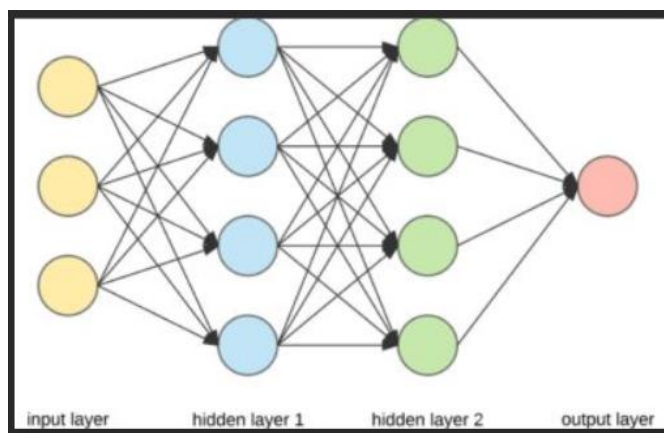


FIGURE 2

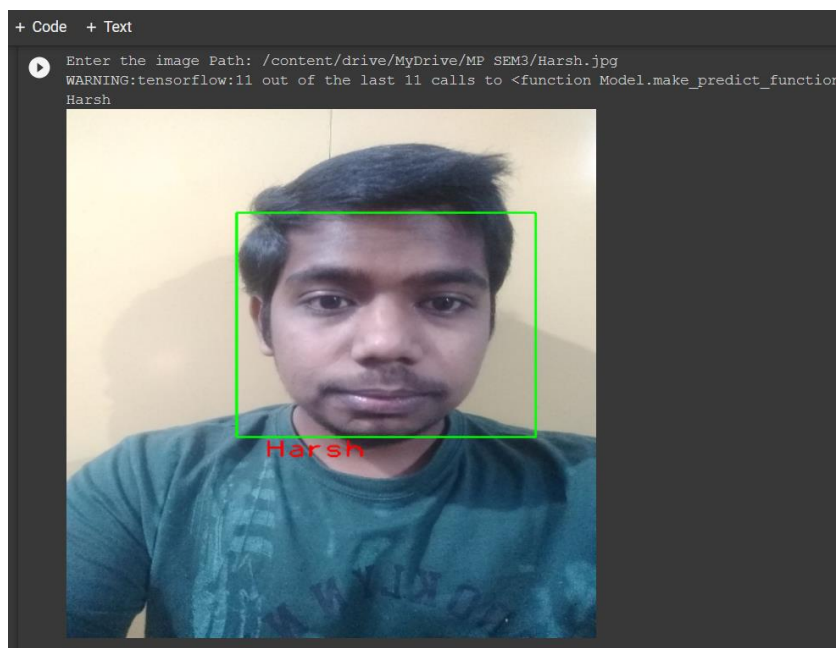


FIGURE 3

ABSTRACT

A face recognition system is one of the biometric information processes, its applicability is easier and working range is larger than others, i.e.; fingerprint, iris scanning, signature, etc. The system uses a combination of techniques in two topics; face detection and recognition. The face detection is performed on live acquired images without any application field in mind. Processes utilized in the system are white balance correction, skin like region segmentation, facial feature extraction and face image extraction on a face candidate. Then a face classification method that uses Feed Forward Neural Network is integrated in the system. The system is tested with a database generated in the laboratory. The tested system has acceptable performance to recognize faces within intended limits. System is also capable of detecting and recognizing multiple faces in live acquired images.

CHAPTER 1

INTRODUCTION

The main idea of our project “FACE RECOGNITION” is to build model inspired on how humans recognize faces. The process happens in a step by step where a person sees a face for the first time then remembers it and finally next time the person encounters the face it remembers from the past experience.

A face recognition system is one of the biometric information processes, its applicability is easier and working range is larger than others, i.e.; fingerprint, iris scanning, signature, etc.

CHAPTER 2

REQUIREMENTS AND ANALYSIS

COMPONENT REQUIREMENTS: -

Overview:

The coding is considered the most important stage in the development process of any software project; hence this step takes considerable time of the overall project lifecycle. In this chapter the design followed by the detailed coding of the project is being the center of attention for detailed discussion. BAs well as specifying the detailed functions of the project's units and interfaces implemented between them.

Body of the software:

Coding done for the project has 3 basic requirement: Programming language, Platform used to code and libraries used.

The language used was Python3a high-level, general-purpose, easy to understand programming language. The above language is used because of its nature of user friendly and having a wide range of feature. The platform used was Anaconda a free and open-source distribution of python language for scientific computing, that aims to simplify package management and development. The packaged is *jupyter-lab* and Google open source platform *Google Colab*. Both software are used for machine learning.

CONCEPT USED:

Deep Learning:

Deep learning is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised. **Deep learning** is an AI function that mimics the workings of the human brain in processing data for use in detecting objects, recognizing speech, translating languages, and making decisions. **Deep learning** AI is able to **learn** without human supervision, drawing from data that is both unstructured and unlabeled.

Supervised learning

Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from labeled training data consisting of a set of training examples.

Convolutional Neural Network:

In deep learning, a convolutional neural network is a class of deep neural networks, most commonly applied to analyzing visual imagery. They are also known as shift invariant or space invariant artificial neural networks, based on their shared-weights architecture and translation invariance characteristics.

LIBRARIES:

One of the most important part of this project was python libraries used. They are the heart of the program.

- OpenCV: Open source computer vision is a library aimed at real time computer vision. It is used to capture picture in this project with many other useful steps.
- OS module: For interacting with the operating system. For creation of text file.
- Keras: It is an open source library that provides a python interface for artificial neural networks.
- NumPy: It is a python library used for working with arrays.
- Matplotlib: It is a plotting library for python programming language and its numerical mathematics NumPy.

TECHNICAL SPECIFICATION:

- Python v3.7.4
- Anaconda v4.8.3
- Spyder v4.0.0
- Google Colab
- OpenCV v4.3.0
- Jupyter v2.1.5

COST ANAYSIS:

| SI. NO. | COMPONENT | UNIT PRICE | TOTAL PRICE |
|-------------|-----------------|-------------|-------------|
| 1 | Python language | Open source | 0 |
| 2 | Anaconda | Open source | 0 |
| 3 | Libraries | Open source | 0 |
| GRAND TOTAL | | | 0 |

CHAPTER 3

The project consists of 3 part:

- Creating dataset
- Training
- Implementation and testing

CREATING DATASET

The first thing is to import different module like :

OpenCV - For image manipulation, Numpy - For matrix manipulation and OS - For file handling and directory operations.

After the modules are imported we assign the path to the raw data and to the dataset of processed image. And then we open the Haarcascade file which is a pre trained classifier of Open CV used for face recognition.

The final dataset contains a nested loop that iterates over every image of the raw dataset. After that subfolders are created with different labels assigned to every subfolders.

Now the face is detected and a rectangle will be formed around the detected face. If the face is not detected so we will get an option for manual cropping also if the detected face is not appropriate, we can go for manual cropping.

TRAINING

In this part the image is converted in array format using NumPy, in the NumPy array labels are stored. We categorize the labels in the number of test cases for it to become binary as its easy and efficient. We normalize the data to reduce the size between 0 to 1 pixel.

A neural network model is then created which consist of 4 layers: an input layer, an output layer and two hidden layers. From these layers our face sample will pass and the face sample will be recognized.

CHAPTER 4

- *Creating Dataset*

```
Create_dataset.ipynb ☆
File Edit View Insert Runtime Tools Help Last edited on December 7
Code + Text Connect Editing

IMPORTING REQUIRED MODULE -
• OpenCV - For image manipulation
• Numpy - For matrix manipulation
• OS - For file handling and directory operations

import cv2 as cv # cv2 is openCV
import numpy as np
import os

[ ] raw_data = "../MP SEM3/Raw Data" # Path of raw data
    dataset = "../MP SEM3/Dataset" # Path of dataset where the processed image will be stored
    drawing = False
    ix,iy = -1,-1

    global pic_name, folder
    pic_name = 1

    ## Opening face detection module, Haarcascade file saved as xml ##
    face_cascade = cv.CascadeClassifier(r"../MP SEM3/haarcascades/haarcascade_frontalface_default.xml")

User defined functions for Manual cropping of Images Using mouse cursor -

## This function draws the rectangle on the passed image using cursor coordinates ##
def draw_rect(event,x,y,flags,params):

    global ix,iy,drawing, pic_name

    img2 = image.copy()
    text = '1.Draw a rectangle and press SPACE.'
    cv.putText(image, text, org=(10,25), fontFace=font, fontScale=1.5, color= (0,0,255),thickness=2)
    text = '2.Only press SPACE if MISSFIT.'
    cv.putText(image, text, org=(10,50), fontFace=font, fontScale=1.5, color= (0,0,255),thickness=2)

    if event == cv.EVENT_LBUTTONDOWN:

        drawing = True
        ix,iy=x,y

    elif event == cv.EVENT_MOUSEMOVE:

        if drawing == True:
            global x1,y1
            x1,y1=x,y
            #cv.rectangle(image, (ix,iy), (x1,y1), (0,255,0),1)

    elif event == cv.EVENT_LBUTTONUP:
        drawing = False
        cv.rectangle(image, (ix,iy), (x1,y1), (0,0,255),2)
        cropped = img2[iy:y1,ix:x1]
        cv.imwrite(os.path.join(folder,f"{pic_name}.jpg"),cropped)
        pic_name = pic_name+1

[ ] ## This function help us take cursor input directly from the user ##
def mouse(image):
    cv.namedWindow("face_crop") # A window will appear with name "face_crop"

    cv.setMouseCallback("face_crop",draw_rect) # Window named "face_crop" is passed to function draw_rect

    while True:

        cv.imshow('face_crop',image)

        key = cv.waitKey(1) & 0xFF
        if key == ord(' '):
            break

    cv.destroyAllWindows()
```


ALGORITHM FOR CREATING FINAL DATASET -

- A nested loop iterates over every image of every folder from 'raw_data'
- It creates subfolders in 'dataset' with the same name as in 'raw_data' folder

```
1 ## First for loop iterates over every folder in the "raw_data" ##
for name in os.listdir(raw_data):
    folder = os.path.join(dataset,name)

    ## Creates a subfolder(with same name in raw_data) inside the dataset folder ##
    if name not in os.listdir(dataset):
        os.mkdir(folder)
        pic_name = 1

    ## Second loop iterates over every images in the every subfolder of raw_data ##
    for pics in os.listdir(f"{raw_data}/{name}"):

        image = cv.imread(f"{raw_data}/{name}/{pics}",1) # Reads image in BGR format
        global img
        img = image.copy() # creating a copy of image

        x,y = 0,0
        gimg = cv.cvtColor(image, cv.COLOR_BGR2GRAY) # Converting the image to greyscale

        ## Detects the face and return a tuple of co-ordinates and dimension of face ##
        face_rect = face_cascade.detectMultiScale(gimg,scaleFactor = 1.21, minNeighbors = 7)

        ## Checking if the detected face is appropriate ##
        if np.shape(face_rect)[0] == 0:
            image = cv.resize(img, (500,500))
            mouse(image)
            continue
        elif len(face_rect)>1:
```

CO

Create_dataset.ipynb

☆

File Edit View Insert Runtime Tools Help Last edited on December 7

Comment Share Settings Help

Connect Editing

iii

+ Code + Text

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

538

539

540

541

542

543

544

545

546

547

548

549

550

551

552

553

554

555

556

557

558

559

560

561

562

563

564

565

566

567

568

569

570

571

572

573

574

575

576

577

578

579

580

581

582

583

584

585

586

587

588

589

590

591

592

593

594

595

596

597

598

599

600

601

602

603

604

605

606

607

608

609

610

611

612

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648

649

650

651

652

653

654

655

656

657

658

659

660

661

662

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702

703

704

705

706

707

708

709

710

711

712

713

714

715

716

717

718

719

720

721

722

723

724

725

726

727

728

729

730

731

732

733

734

735

736

737

738

739

740

741

742

743

744

745

746

747

748

749

750

751

752

753

754

755

756

757

758

759

760

761

762

763

764

765

766

767

768

769

770

771

772

773

774

775

776

777

778

779

780

781

782

783

784

785

786

787

788

789

790

791

792

793

794

795

796

797

798

799

800

801

802

803

804

805

806

807

808

809

810

811

812

813

814

815

816

817

818

819

820

821

822

823

824

825

826

827

828

829

830

831

832

833

834

835

836

837

838

839

840

841

842

843

844

845

846

847

848

849

850

851

852

853

854

855

856

857

858

859

860

861

862

863

864

865

866

867

868

869

870

871

872

873

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888

889

890

891

892

893

894

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

918

919

920

921

922

923

924

925

926

927

928

929

930

931

932

933

934

935

936

937

938

939

940

941

942

943

944

945

946

947

948

949

950

951

952

953

954

955

956

957

958

959

960

961

962

963

964

965

966

967

968

969

970

971

972

973

974

975

976

977

978

979

980

981

982

983

984

985

986

987

988

989

990

991

992

993

994

995

996

997

998

999

1000

if np.shape(face_rect)[0] == 0:

image = cv.resize(img, (500,500))

mouse(image)

continue

elif len(face_rect)>1:

image = cv.resize(img, (500,500))

mouse(image)

continue

Draw a rectangle denoting the detected image

for (x,y,w,h) in face_rect:

cv.rectangle(image, (x,y), (x+w,y+h), (0,255,0),10)

image = cv.resize(image, (500,500)) # resize the image to constant dimension

|

font = cv.FONT_HERSHEY_PLAIN

text1 = '1.Click Space for saving.'

cv.putText(image, text1, org=(10,30), fontFace=font, fontScale=1.5, color= (0,0,255),thickness=2)

text2 = '2.Click N for manual Cropping.'

cv.putText(image, text2, org=(10,50), fontFace=font, fontScale=1.5, color= (0,0,255),thickness=2)

cv.imshow('pics',image) # Display image

cropped_image = img[y-1:y+h+1, x-1:x+w] # Cropping out the face from full image

global key

key = cv.waitKey(0) & 0xFF # taking a input from keyboard as confirmation key

if key == ord(' '):

cv.imwrite(os.path.join(folder,f"{pic_name}.jpg"),cropped_image) # Saving the cropped face in dataset folder

pic_name = pic_name+1 #name of the image for convenience

elif key == ord('n'):

image = cv.resize(img, (500,500))

mouse(image) #calls function for cropping image manually

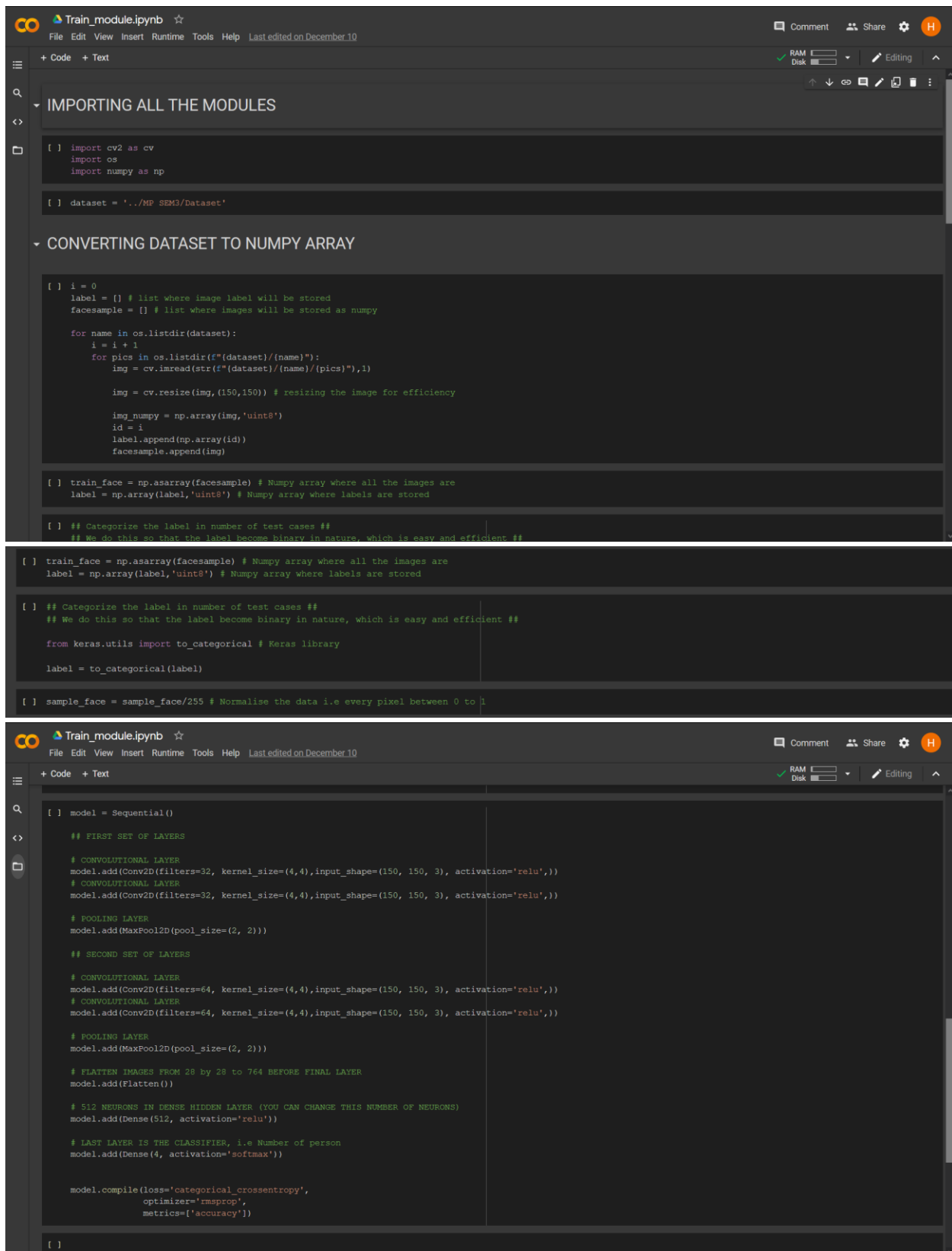
continue

elif key == ord('c'):

break

cv.destroyAllWindows()

- *Training Dataset*



```
Train_module.ipynb ☆
File Edit View Insert Runtime Tools Help Last edited on December 10

+ Code + Text
RAM Disk
Comment Share Settings H

IMPORTING ALL THE MODULES

[ ] import cv2 as cv
import os
import numpy as np

[ ] dataset = '../MP SEM3/Dataset'

CONVERTING DATASET TO NUMPY ARRAY

[ ] i = 0
label = [] # list where image label will be stored
facesample = [] # list where images will be stored as numpy

for name in os.listdir(dataset):
    i = i + 1
    for pics in os.listdir(f"{dataset}/{name}"):
        img = cv.imread(str(f"{dataset}/{name}/{pics}"),1)

        img = cv.resize(img,(150,150)) # resizing the image for efficiency

        img_numpy = np.array(img,'uint8')
        id = i
        label.append(np.array(id))
        facesample.append(img)

[ ] train_face = np.asarray(facesample) # Numpy array where all the images are
label = np.array(label,'uint8') # Numpy array where labels are stored

[ ] ## Categorize the label in number of test cases ##
## We do this so that the label become binary in nature, which is easy and efficient ##

[ ] train_face = np.asarray(facesample) # Numpy array where all the images are
label = np.array(label,'uint8') # Numpy array where labels are stored

[ ] ## Categorize the label in number of test cases ##
## We do this so that the label become binary in nature, which is easy and efficient ##

from keras.utils import to_categorical # Keras library

label = to_categorical(label)

[ ] sample_face = sample_face/255 # Normalise the data i.e every pixel between 0 to 1

Train_module.ipynb ☆
File Edit View Insert Runtime Tools Help Last edited on December 10

+ Code + Text
RAM Disk
Comment Share Settings H

[ ] model = Sequential()

## FIRST SET OF LAYERS

# CONVOLUTIONAL LAYER
model.add(Conv2D(filters=32, kernel_size=(4,4),input_shape=(150, 150, 3), activation='relu',))
# CONVOLUTIONAL LAYER
model.add(Conv2D(filters=32, kernel_size=(4,4),input_shape=(150, 150, 3), activation='relu',))

# POOLING LAYER
model.add(MaxPool2D(pool_size=(2, 2)))

## SECOND SET OF LAYERS

# CONVOLUTIONAL LAYER
model.add(Conv2D(filters=64, kernel_size=(4,4),input_shape=(150, 150, 3), activation='relu',))
# CONVOLUTIONAL LAYER
model.add(Conv2D(filters=64, kernel_size=(4,4),input_shape=(150, 150, 3), activation='relu',))

# POOLING LAYER
model.add(MaxPool2D(pool_size=(2, 2)))

# FLATTEN IMAGES FROM 28 by 28 to 764 BEFORE FINAL LAYER
model.add(Flatten())

# 512 NEURONS IN DENSE HIDDEN LAYER (YOU CAN CHANGE THIS NUMBER OF NEURONS)
model.add(Dense(512, activation='relu'))

# LAST LAYER IS THE CLASSIFIER, i.e Number of person
model.add(Dense(4, activation='softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])

[ ]
```

How to shuffle two NumPy arrays · X MP SEM3.ipynb - Colaboratory X Colab Notebooks - Google D... X Train_module.ipynb - Colaboratory X MP SEM3(final).ipynb - Colaboratory X

https://colab.research.google.com/drive/1OZ38cDej_9UkdC69qrbCV8D1TZFO_Qn#scrollTo=8dFuYrg08Mv

MP SEM3.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Files

- MyDrive
 - Colab Notebooks
 - MP SEM3
 - Dataset
 - Raw Data
 - haarcascades
 - Create_dataset.ipynb
 - Train_module.ipynb
 - face_data.npy
 - face_label.npy
 - New Dataset
 - Anikesk_test
 - Create_dataset.ipynb
 - GRP17 Face recognition...
 - chromedriver.exe
 - chromedriver_win32.zip
 - face_rec_test.h5
 - yolo (1).h5
 - yolo.h5
 - sample_data
 - 13.jpg
 - 2.jpg
 - 8.jpg

Disk 75.58 GB available

+ Code + Text

```
[43] Total params: 35,803,332
Trainable params: 35,803,332
Non-trainable params: 0

model.fit(final_face,final_label,verbose=1,epochs=10)

Epoch 1/10
8/8 [=====] - 32s 4s/step - loss: 6.4402 - accuracy: 0.2467
Epoch 2/10
8/8 [=====] - 32s 4s/step - loss: 1.6831 - accuracy: 0.3480
Epoch 3/10
8/8 [=====] - 32s 4s/step - loss: 1.5062 - accuracy: 0.3789
Epoch 4/10
8/8 [=====] - 32s 4s/step - loss: 2.2416 - accuracy: 0.5859
Epoch 5/10
8/8 [=====] - 32s 4s/step - loss: 0.7026 - accuracy: 0.6608
Epoch 6/10
8/8 [=====] - 32s 4s/step - loss: 1.1632 - accuracy: 0.7709
Epoch 7/10
8/8 [=====] - 32s 4s/step - loss: 0.5190 - accuracy: 0.8150
Epoch 8/10
8/8 [=====] - 32s 4s/step - loss: 0.1041 - accuracy: 0.9780
Epoch 9/10
8/8 [=====] - 32s 4s/step - loss: 0.0939 - accuracy: 0.9780
Epoch 10/10
8/8 [=====] - 32s 4s/step - loss: 0.2711 - accuracy: 0.9075
<tensorflow.python.keras.callbacks.History at 0x7f2c15efaid0>

[47] model.evaluate(sample_face,label)

8/8 [=====] - 7s 838ms/step - loss: 0.0496 - accuracy: 0.9780
[0.04956744238734245, 0.9779735803604126]
```

1432 11-12-2020

- *Testing the Trained module*

```
TEST.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
[21] import numpy as np
import os
from google.colab.patches import cv2_imshow as imshow
from keras.models import load_model

face = cv.CascadeClassifier(cv.data.harcascades+'haarcascade_frontalface_default.xml')

ind = load_model('/content/drive/MyDrive/MP SEM3/FINAL_SEM3.h5')

names = ['None','Harsh','Manas','Muskan','Vikash']

image_path = input('Enter the image Path: ')

image = cv.imread(image_path,1)
gimg = cv.cvtColor(image, cv.COLOR_BGR2GRAY)

rect = face.detectMultiScale(gimg,1.21,3)

for (x,y,w,h) in rect:
    cv.rectangle(image, (x,y), (x+w,y+h), (0,255,0), 3)

cropped = image[y-1:y+h+1, x-1:x+w]

img = np.array(cv.resize(cropped, (150,150)), 'uint8')
lis = []
lis.append(img)
lis = np.array(lis, 'uint8')

ans = ans = np.argmax(ind.predict(lis),-1)

print(names[ans[0]])

font = cv.FONT_HERSHEY_PLAIN
text1 = names[ans[0]]
cv.putText(image, text1, org=(x+50,y+h+50), fontFace=font, fontScale=4, color= (0,0,255),thickness=3)

image = cv.resize(image, (500,500))

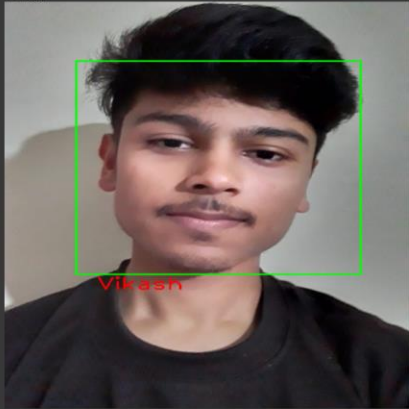
imshow(image)

cv.putText(image, text1, org=(x+50,y+h+50), fontFace=font, fontScale=4, color= (0,0,255),thickness=3)

image = cv.resize(image, (500,500))

imshow(image)

Enter the image Path: /content/drive/MyDrive/MP SEM3/Vikash.jpg
WARNING:tensorflow:11 out of the last 11 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f30eef28510> triggered tf.function retracing. Tr
Vikash
```



CHAPTER 5

IMPLEMENTATION AND TESTING

We create a program for testing our trained model. In this part first we take an image as input. The image is then processed and converted into numpy array. The newly formed array is then passed through the model which give a numeric as return type. The return type is nothing but the label against which we had trained the model earlier. Our model has an accuracy of 90.75% .

CHAPTER 6

RESULT AND DISCUSSION

Once the software has run as discussed in Chapter 3 , the setup is ready for testing. When we carried out implementations and testing the results were obtained and we can get the image i.e, the face, recognized and further the label of the image. Showing us the positive result of the experiment.

CHAPTER 6

CONCLUSION

The model that we implemented in this project is chosen after researches and testing results to confirm that its reliable. The model is tested under different conditions and the results were positive. Furthermore, it holds a large scope in the security system. The proposed model will help to reduce the security issues. It could also be used in various MNCs for security system, as attendance monitoring system, phone, PCs lock and so.

References

- 1) *Teachers*
- 2) <https://www.pyimagesearch.com/>
- 3) [geeksforgeeks.com](https://www.geeksforgeeks.com/)
- 4) deeplearning.ai
- 5) [google.com](https://www.google.com/)
- 6) opencv.org