CSY 3025 Assignment 2

Scan a face to register attendance.

By

Aayush Shrestha

A report of assignment 2 of CSY3025

Bsc Computing (Software Engineering)

Datasets link: https://github.com/Stha5665/AI\_assignment2.git

The University Northampton

2023

**1 Introduction**

The attendance system is one of the essential systems for any organization or institution. From university to different companies, it is required to track the present or absent status of every employee. This system identifies the performance of individuals, employees, and students. For internal evaluation, day-day attendance is the most essential part. The traditional attendance system is based on paper based and it needs to call all students or employees one by one to mark their status. This can be painful in terms of time consumption as well as storage. So, modern technology can be implemented to solve this problem. Artificial technology has a solution for this. Deep learning covers the area of face, biometric, object identification which can be implemented for the attendance system. Similarly, Convolution network (CNN) is most useful these days in terms of image processing and classification. It learns the features from images and identifies the patterns and finally gives the right classification of tested images. CNN model is most popular for its accuracy, so it is implemented in this system.

**2 Problem Analysis and Background Research:**

**2.1 Problem Description**

In the era of computer world, manual way of attendance and tracking record in paper-based system is very time consuming and inconsistent. Data provided through the present system cannot be believed blindly. The attendance records can be easily manipulated, which can affect the institution or company’s performance. The performance of individual employee can be stored in paper files, but it is not easy to handle or store the record for long time due to low capacity or environmental reasons(Yang & Han, 2020). Similarly in the context of university, schools or colleges, student or tutor attendance records can be easily cheated in many ways.(Pooja, 2023) It also consumes huge time by calling each student one by one, identifying them, and keeping record. As attendance details can be easily misplaced, the increasing rate of skipping classes, and the increasing numbers of absent student effects the moral value of the institution. This will present false performance representation of that student or organization.

So, face recognition-based attendance system is one of the trendy topics, many institutions need to shift to online-face recognition-based attendance system.(STMIK AKAKOM Yogyakarta et al., n.d.)

**2.2 Literature Review**

Attendance ensures that someone is physically present to the organization and has given their time to organization, which is their duty. The commercial value of the organization is related to the group of people they are bound with and their availability. According to author (Yang & Han, 2020), traditional check-in method is unbeneficial in present context as it is time consuming and needs separate physical spaces to store record. So, automated machines or program for keeping and managing attendance are very valuable. Dealing with automated process machine learning helps in most of the field and in this case the face recognition, face detection is integrated in a attendance system. Author mentioned that the accuracy rate of the face recognition system rises to 82%. Author had mentioned different proposed techniques for this system, face recognition is the first one. In this technique facial features of person are extracted from camera equipment and if the extracted pattern matches with model, then the system will recognize the image else it will not. Features extracted from LDA (Linear Discrimination Analysis), Neural network method and Support Vector Machine (SVM) methods are some of interesting fields. The author had done experiments with designing module for recognition and its database design.

(Pooja, 2023) had implemented face recognition-based attendance system which is useful to different universities or educational institutions. This system uses LBPH face recognizer to identify valid students. It simply consists of verification and recognition, utilizing hardware and software. Authors had mentioned that admin, tutors, students, and employees are the users for this system. The following had followed the data protection act for data protection. The author had given different modules of this system like check camera, train images, recognize faces, automatic mail. In check camera, it uses OpenCV library and detects the faces available in front of the camera, it converts the images into grayscale for detection or training. The grayscale images are separated by filename with id in train images module. After that it uses Haar-Cascade Classifier and uses LBPH algorithm to recognize and add recognized students into csv file and store it. Author had the automatic mail option through which the attendant record can be send into teacher or admin mail for the verification. This minimizes the time consumption for attendance and provides online storage of the attendance records along with student data protection and accuracy.

**3 Building Deep Learning Network**

The image classifier is the essential component for this system. Proper image classifier results in accurate image classification which results in actual face recognition. So, to train and build the best classifier, I have used Keras, Tensorflow, CNN model. For efficient training of model, datasets are the essential one. Dataset consists of raw face images. The building of the system or deep learning network are as follows:

**3.1 Dataset**

Collection of raw facial images was the first step for building this network. The collection of images of my friends was complex and time-consuming. For building my system, I have collected 10 raw images including my image. After collection, I renamed the images and structured them in different folders. I have used folders like train, test, and validation for training of model, inside which I have separated each image in each folder.

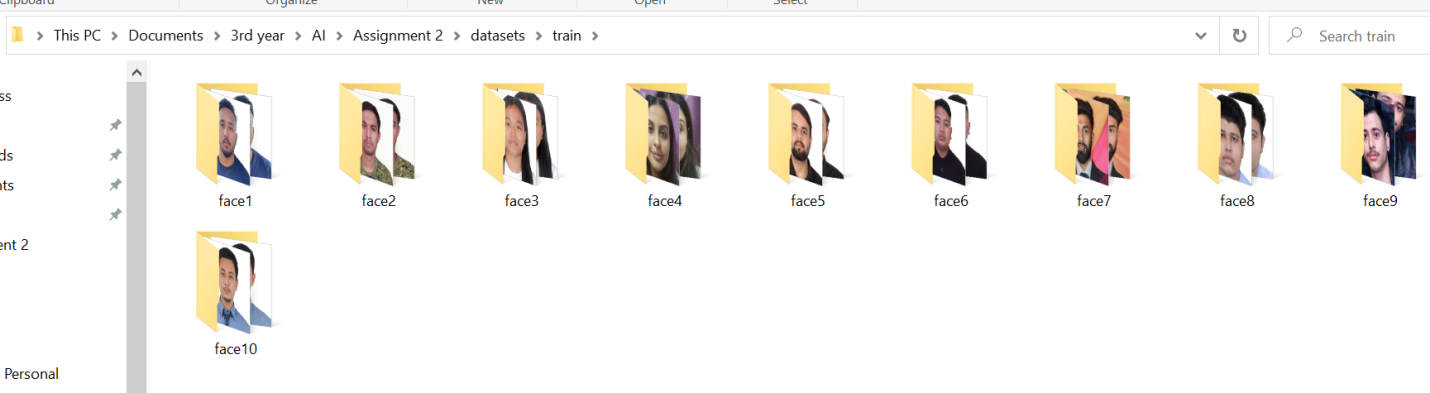


Figure: Images separated in different folders for training.

After building this folder structure, I preprocessed and augmented my images for higher accuracy. I detected each face through use of harcascade, resized into 64\*64 pixels and converted each image into gray scale.



This function is used to convert raw images into preprocess augmented images.

**3.2 Deep Learning Network**

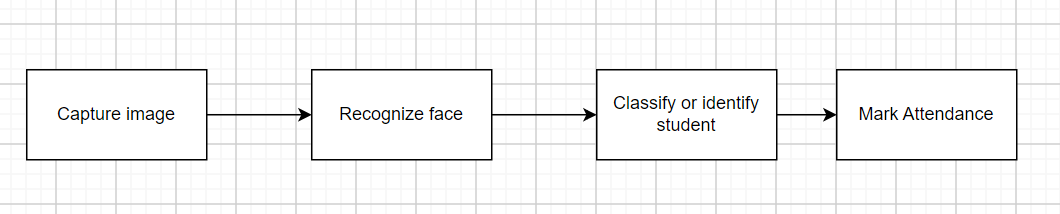
****

Figure: Basic flow of this system

The basic flow of the face recognition-based attendance system starts with capturing image, recognizing face, identifying face, and marking attendance. This consists of various input, hidden and output layers. This model is built using the CNN model. CNN model consists of different layers of convolution along with max-pooling layer (Khan et al., 2019) as convolution layer.



Figure: Architecture of CNN model for image prediction of MNIST digit classification

The above figure shows the working model of Convolution Neural Network. This type of network is called as convnet. Here, from raw images different features maps are extracted using stride (3\*3 or 5\*5 matrices). Each pattern, features edges are identified in the convolution layer by moving stride over the feature maps. The extracted feature map is passed to max-pooling layer for decreasing their size and passed to flatten layer in which 3-dimensional tensors are converted into 1-dimensional tensors. These one-dimensional tensors passed into fully connected layers (like Dense layer) which classified the images and final predictions is presented into output layer.

This system consists of an input layer (1st Convolution layer), hidden layers and output layer. The input layer takes input of shape 64\*64 matrices with RGB channels. Here, the activation function RELU is used as it is simple, the most common activation layer in deep learning. It helps to solve the vanishing gradient problem, by taking positive arguments of data.

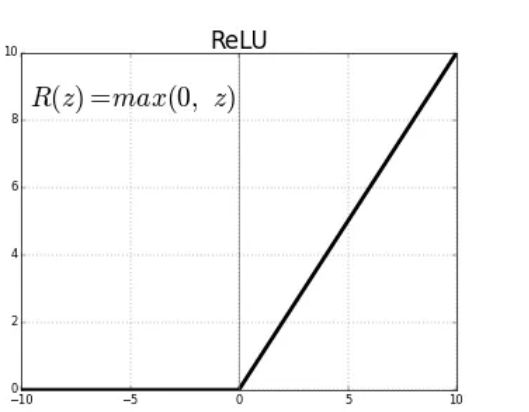
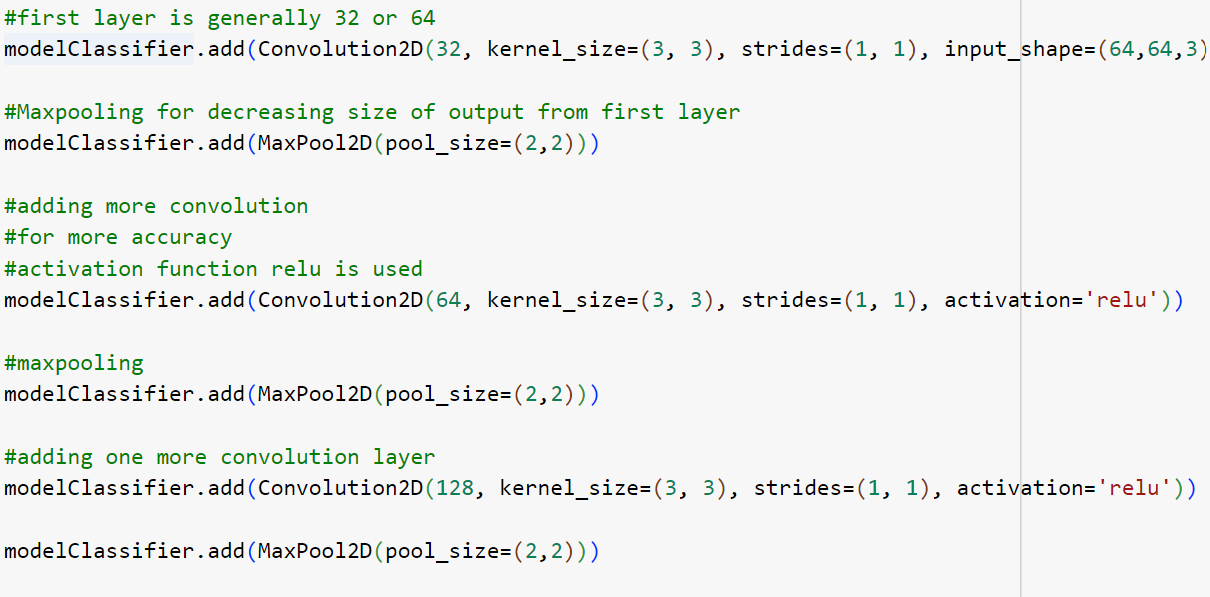


Figure: ReLU activation function

This plot shows that the range of ReLU activation function lies between 0 to infinity. Negative arguments are automatically set to zero.

The first layer of my convnet has 32 filters to be computed in features map with sliding window of 3\*3 size. The result is passed into max-pooling layer (2,2) where the size of the feature map is halved. Then, the extracted feature map is passed into a hidden layer where two convolution layers are used for exact feature extraction following by max pooling layer. Then it is flattened into one dimensional array and the result is passed into a dense layer with activation function ‘relu’. The result is passed into output layer which classifies and gives probabilities of each class label using ‘softmax’ function. Softmax is generally used in output layer for the providing probability of predicited image in each class or label. It is used in multi-class classification problem.

Figure: Code for building layers of convolution network

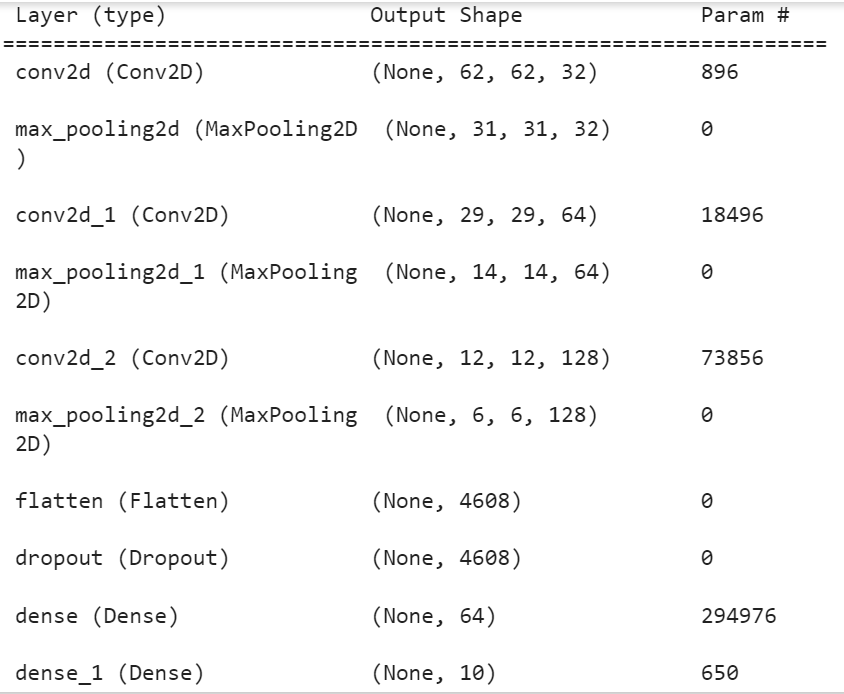


A picture containing text, screenshot, font, line

Description automatically generated

Figure: Code for dense layer, output layer

The network architecture can be viewed with the code ‘model.summary()’. The below figures present the CNN network architecture of this system or model.



A picture containing text, screenshot, receipt, line

Description automatically generated

Figure: CNN Network architecture for this system

Loss Functions and optimizer have an important role for evaluating output and generating exact output. For this system I have used Categorical crossentropy as the loss function and adam optimizer. Categorical Crossentropy is also known as Softmax loss, it is used for the binary or multi-class classification problem. As our system needs to predict different faces or classes, this function would be the best choice.

Optimizers are algorithms for increasing performance of system by minimizing the loss function. It helps to view the change in weight and learning rate of neural networks. Here this model uses adam optimizer, which is most popular gradient descent optimization algorithms. It combines the advantages of both RMS-Prop and Adadelta optimization algorithms. It is efficient , easy to implement and requires less memory in such neural networks.

**3.3 Training and evaluation**

A screenshot of a computer

Description automatically generated with medium confidence

Figure: Code to compile and train the model

After the result of output layer, model is compiled using categorical\_crossentropy loss function, adam optimizer and trained with training dataset and test dataset parallelly. Nine epochs are used for training of model. The obtained results are given below in figure.

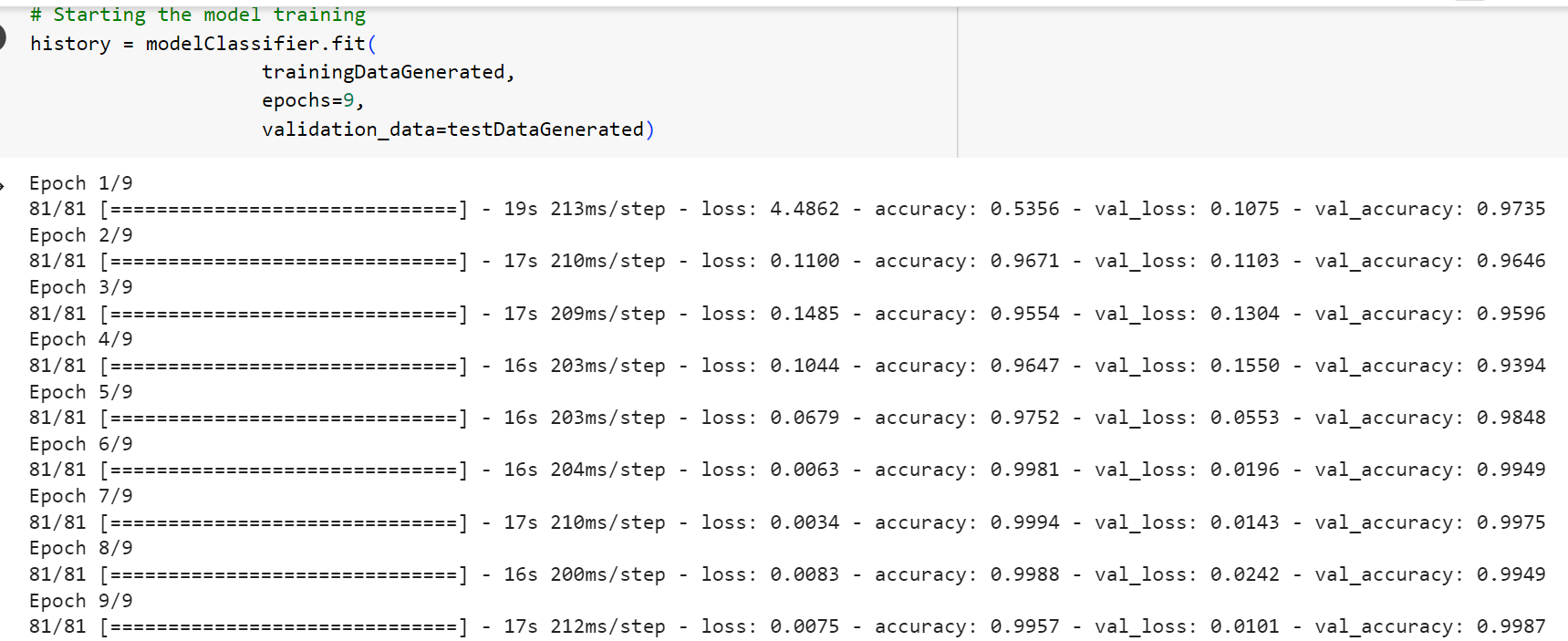


Figure: Output obtained while training model

Due to the convnet, augmented images, dropout function the training and validation accuracy seems to be in same level, which solves issues like overfitting and underfitting. The performance of model is drawn using matplot library.

A picture containing text, screenshot, line, plot

Description automatically generated

Figure: Plot of training and validation accuracy during training of model in each epoch

This is figure obtained by plotting the accuracy level of training and validation data during each epoch. Max 100% accuracy is available which is denoted by 1. The accuracy of training and validation increased upto 99%.

A picture containing text, screenshot, display, plot

Description automatically generated

Figure: Plot of training and validation loss during training of model

This is figure obtained by plotting the loss level of training and validation data during each epoch. Max 40% loss is available which is denoted by 4. The loss was lowered nearly up to 1%.

**Camera capture**: There is code snippet available in google collab for the capturing images through google collab. This system uses that snippet for capturing images.(*Google Colab: Access Webcam for Images and Video*, n.d.)

**4 Testing:**

The system was able to recognize the student and marked the student record. The result of testing conducted is given below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Testing** | **Expected Result** | **Result obtained** | **Screenshots** |
| Preprocessing Image | The model should preprocess images, detect face from image, crop and save image in grayscale. | Same as Expected |  |
| Make Convolution network | Code should make convolution layer, max pooling layer along with dense layer | Same as Expected |  |
| Map of Label | It should map the faces label and store it in array which would be useful for attendance and prediction | Same as Expected |  |
| Capture face image | The snippet should be able to capture image and store it | Same as expected |  |
| Prediction of different images | The trained model should classify the images used for training, validation, and new captured images | Same as expected |  |
| Write to the attendance | It should write the identified student details to attendance file or csv file | Same as expected |  |
| No Duplicate entry | No duplicate entry for the student attendance record in a attendance file | Same as expected |  |

Testing different images:

1. Testing Face 1 image:

A screenshot of a computer

Description automatically generated with medium confidence

Fig: Testing face 1 image by giving the image path to directory of face1

Result:

A close-up of a computer screen

Description automatically generated with low confidence

1. Testing face 3 image:

A screenshot of a computer

Description automatically generated with medium confidence

Figure: Image path to face 3 image

Result: After prediction:

A picture containing human face, text, screenshot, person

Description automatically generated

1. Testing Face 6

A screenshot of a computer

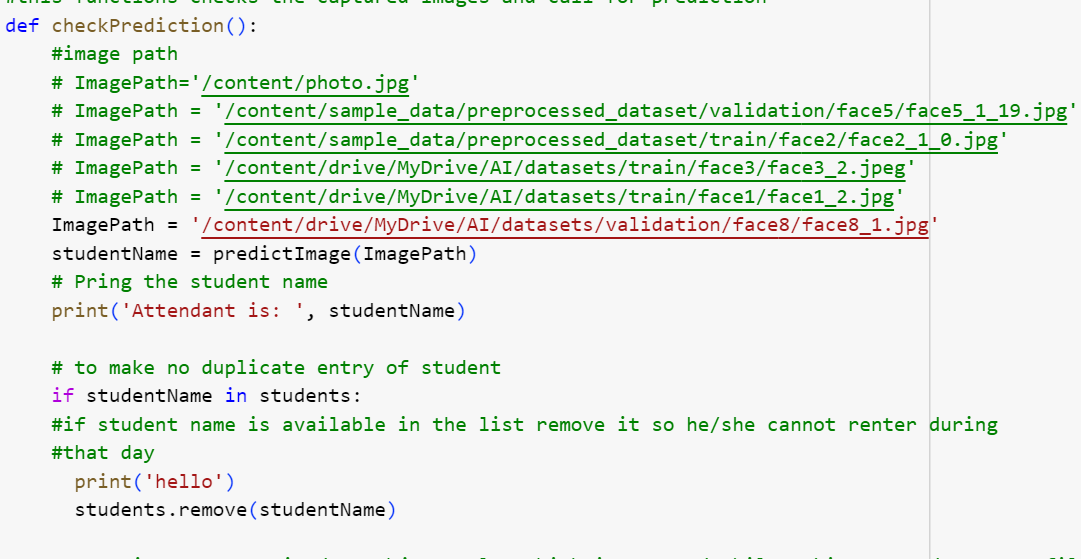
Description automatically generated with low confidence

Result:

A person's face with text

Description automatically generated with low confidence

1. Testing face 8 image:

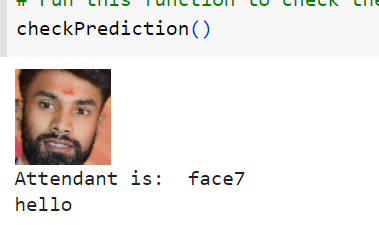


Result:

A screen shot of a computer

Description automatically generated with low confidence

1. Testing face 7 image:



**5 Discussion and Conclusion:**

This assignment was interesting for developing and learning. In the way of artificial intelligence, deep learning is one of the trendy topics and face recognition by using deep learning can be very useful in daily lives. Likewise converting manual paper-based attendance system to automated system will be efficient and time saving. Face recognition-based attendance system can be easily implemented in different institutions. This topic covers the face detection and face recognition too. The use of CNN model for face identification was helpful for me, the prediction was accurate making this system useful.

**References**

*Google Colab: Access Webcam for Images and Video*. (n.d.). Retrieved June 18, 2023, from https://colab.research.google.com/

Khan, M. Z., Harous, S., Hassan, S. U., Ghani Khan, M. U., Iqbal, R., & Mumtaz, S. (2019). Deep Unified Model for Face Recognition Based on Convolution Neural Network and Edge Computing. *IEEE Access*, *7*, 72622–72633. https://doi.org/10.1109/ACCESS.2019.2918275

Pooja, G. (2023). Face Recognition Based Attendance System. *International Journal for Research in Applied Science and Engineering Technology*, *11*(5), 5408–5412. https://doi.org/10.22214/ijraset.2023.52868

STMIK AKAKOM Yogyakarta, Institute of Electrical and Electronics Engineers. Indonesia Section, & Institute of Electrical and Electronics Engineers. (n.d.). *2nd ISRITI 2019 proceeding : the 2nd International Seminar on Research of Information Technology and Intelligent Systems 2019 : “The future & challenges of extended intelligence” : Yogyakarta, Indonesia, 05-06 December 2019*.

Yang, H., & Han, X. (2020). Face recognition attendance system based on real-time video processing. *IEEE Access*, *8*, 159143–159150. https://doi.org/10.1109/ACCESS.2020.3007205

**Appendix**

**Source code of my application:**

# mounting google drive

import cv2

from google.colab import drive

drive.mount("/content/drive")

import numpy as np

import cv2

#importing cascadeClassifier

from cv2 import CascadeClassifier

import os

#this for reading image

from cv2 import imread

#this for showing image

from google.colab.patches import cv2\_imshow

#random number

import random

#for generating images

from keras.preprocessing.image import ImageDataGenerator

# this functions preprocesses and augments the raw image

def preprocessImages(rawDataPath, preprocessedDataPath):

# setting attributes for new generated images

datagen = ImageDataGenerator(

rotation\_range=30,

width\_shift\_range=0.1,

height\_shift\_range=0.1,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True,

)

#size of image

targetSize = (128, 128)

# total no

#of images to generate

totalNumAugmentedImages = 100

#loops over raw images for augmentation

for imagesFileName in os.listdir(rawDataPath):

# path name of new image file

image\_path = os.path.join(rawDataPath, imagesFileName)

#shows the new path

print(imagesFileName)

#this loops over faces directory

for imageName in os.listdir(image\_path):

# Load the image

imagesDirectoryPath = os.path.join(image\_path, imageName)

#reads images with cv2.imread function

image = cv2.imread(imagesDirectoryPath)

#obtained image is resized to target size

resizedImageObtained = cv2.resize(image, targetSize)

# temporary variable to count total images

temp = 0

#changing image to gray scale

inputImage = cv2.cvtColor(resizedImageObtained, cv2.COLOR\_BGR2RGB)

#convert into numpy array

inputImage = np.array(inputImage)

inputImage = inputImage.reshape((1,) + inputImage.shape)

# data augumentation with face detection

for i in range(totalNumAugmentedImages):

# generate total images

temp +=1

# ony generate upto required size

if temp == totalNumAugmentedImages:

break

# get augmented images

augmentedImages = datagen.flow(inputImage, batch\_size=1, save\_format='jpg')

# find images

# convert them

obtainedAugmentedImage = next(augmentedImages)[0]

# convert them into following type

obtainedAugmentedImage = obtainedAugmentedImage.astype(np.uint8)

# faceDetection with using haarcascade

faceDetect = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml')

# detect faces

faces = faceDetect.detectMultiScale(obtainedAugmentedImage, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

# it loops over faces and cops them

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

# creating

# new path for augmented images

augmentedProcessedImagePath = os.path.join(preprocessedDataPath, imagesFileName)

# print the path

print(augmentedProcessedImagePath)

# make directiory if not exists

os.makedirs(augmentedProcessedImagePath, exist\_ok=True)

# get cropped images of detected face

cropped\_image = obtainedAugmentedImage[y:y+h, x:x+w]

# Save

# augmented images in following path

obtainedAugmentedImage\_filename = f'{imageName.split(".")[0]}\_{i}.jpg'

# get path

obtainedAugmentedImage\_path = os.path.join(augmentedProcessedImagePath, obtainedAugmentedImage\_filename)

# create or write image in the following path

cv2.imwrite(obtainedAugmentedImage\_path, cropped\_image)

# he directory containing the raw images

raw\_train\_data\_dir = '/content/drive/MyDrive/AI/datasets/train'

# raw image directory for raw images

raw\_validation\_data\_dir = '/content/drive/MyDrive/AI/datasets/validation'

# preprocessed data directory

preprocessed\_train\_data\_dir = '/content/sample\_data/preprocessed\_dataset/train'

# for validation path

preprocessed\_validation\_data\_dir = '/content/sample\_data/preprocessed\_dataset/validation'

# Create the output directory if it doesn't exist

os.makedirs(preprocessed\_train\_data\_dir, exist\_ok=True)

# create the validation directory

os.makedirs(preprocessed\_validation\_data\_dir, exist\_ok=True)

# calling function for augmentign raw train images

preprocessImages(raw\_train\_data\_dir, preprocessed\_train\_data\_dir)

# augmenting validation images

# available in given path

preprocessImages(raw\_validation\_data\_dir, preprocessed\_validation\_data\_dir)

#make currnet attendance file

from datetime import datetime

import csv

# directory of training image to store faces names

directory='/content/sample\_data/preprocessed\_dataset/train'

# store name in following directory

students = []

# loops over each directory and stores name

for file in os.listdir(directory):

# append name

# into array

students.append(file)

# print all names

print(students)

# get current time

currentTimeDate = datetime.now()

# in format of y-m-d

currentDateVal = currentTimeDate.strftime("%Y-%m-%d")

# create new csv file

attendanceFile = open(currentDateVal + '.csv', 'w', newline='')

# create writer objects which writes to the file

attendanceWriter = csv.writer(attendanceFile)

# write the following header in the file

attendanceWriter.writerow(['Student Name', 'Arrived Time'])

# save to csv file

attendanceFile.flush()

# Deep Learning CNN model

# paths containing preprocessed dataset

trainingImagePath = '/content/sample\_data/preprocessed\_dataset/train'

# path for validation of image

validationImagePath = '/content/sample\_data/preprocessed\_dataset/validation'

# No transformations made on the testing images

testDataGenerator = ImageDataGenerator()

trainDataGenerator = ImageDataGenerator()

# Data generating for testing

testDataGenerated = testDataGenerator.flow\_from\_directory(

validationImagePath,

target\_size=(64, 64),

batch\_size=20,

class\_mode='categorical')

# "

# Data generated for training

# with 20 batchsize

trainingDataGenerated = trainDataGenerator.flow\_from\_directory(

trainingImagePath,

batch\_size=20,

target\_size=(64, 64),

class\_mode='categorical')

# "

# labels for face

trainingDataGenerated.class\_indices

#this variable stores has tag in number for all faces

trainDataClasses=trainingDataGenerated.class\_indices

#This stores faces names which will be useful

MapForFaces={}

# looping through values and storing as json object

for faceValue,faceName in zip(trainDataClasses.values(),trainDataClasses.keys()):

MapForFaces[faceValue]=faceName

#this map to faces value with its file name

print("array Mapping of Face Name, faces:\n",MapForFaces)

# total output layer is equal to total faces

totalNeurons=len(MapForFaces)

print('\n Total output neurons or layers: ', totalNeurons)

# importing layers packages for creating CNN layers

from keras.layers import Dropout

# improting dense layer

from keras.layers import Dense

# max pool layer

from keras.layers import MaxPool2D

# for flatten layer

# "

from keras.layers import Flatten

# sequention package

from keras.models import Sequential

# package for convolution

from keras.layers import Convolution2D

# as there is small no of images for training and testing

# we will use 3 layers of convulation

#2 layers of maxpool

#and dense layers

#initializing

modelClassifier= Sequential()

''' first layer of Convolution

In tensorflow we use format(64,64,3)

# accepts 3 matrix of size (64X64) pixels with RGB components

'''

#first layer is generally 32 or 64

modelClassifier.add(Convolution2D(32, kernel\_size=(3, 3), strides=(1, 1), input\_shape=(64,64,3), activation='relu'))

#Maxpooling for decreasing size of output from first layer

modelClassifier.add(MaxPool2D(pool\_size=(2,2)))

#adding more convolution

#for more accuracy

#activation function relu is used

modelClassifier.add(Convolution2D(64, kernel\_size=(3, 3), strides=(1, 1), activation='relu'))

#maxpooling

modelClassifier.add(MaxPool2D(pool\_size=(2,2)))

#adding one more convolution layer

modelClassifier.add(Convolution2D(128, kernel\_size=(3, 3), strides=(1, 1), activation='relu'))

# halved the feature map

modelClassifier.add(MaxPool2D(pool\_size=(2,2)))

#flattening the result to single dimenstion array

modelClassifier.add(Flatten())

modelClassifier.add(Dropout(0.5))

#adding dense layer

modelClassifier.add(Dense(64, activation='relu'))

#adding output layer with softmax activation function

modelClassifier.add(Dense(totalNeurons, activation='softmax'))

# this displays the layer structure

modelClassifier.summary()

#compiling model for CNN

modelClassifier.compile(loss='categorical\_crossentropy', metrics=["accuracy"], optimizer = 'adam')

# Starting the model training

history = modelClassifier.fit(

trainingDataGenerated,

epochs=9,

validation\_data=testDataGenerated)

# This is the code snippet for camera capture of google collab

#(Google Colab: Access Webcam for Images and Video, n.d.)

#Reference: Google Colab: Access Webcam for Images and Video. (n.d.). Retrieved June 12, 2023, from https://colab.research.google.com/

from IPython.display import display, Javascript

from google.colab.output import eval\_js

from base64 import b64decode

#code for caputring images

#(Google Colab: Access Webcam for Images and Video, n.d.)

#Reference: Google Colab: Access Webcam for Images and Video. (n.d.). Retrieved June 12, 2023, from https://colab.research.google.com/

def onlineTakePhoto(fileName='photo.jpg', quality=0.8):

js = Javascript('''

async function takePhoto(quality) {

const div = document.createElement('div');

const capture = document.createElement('button');

capture.textContent = 'Capture';

div.appendChild(capture);

const video = document.createElement('video');

video.style.display = 'block';

const stream = await navigator.mediaDevices.getUserMedia({video: true});

document.body.appendChild(div);

div.appendChild(video);

video.srcObject = stream;

await video.play();

// Resize the output to fit the video element.

google.colab.output.setIframeHeight(document.documentElement.scrollHeight, true);

// Wait for Capture to be clicked.

await new Promise((resolve) => capture.onclick = resolve);

const canvas = document.createElement('canvas');

canvas.width = video.videoWidth;

canvas.height = video.videoHeight;

canvas.getContext('2d').drawImage(video, 0, 0);

stream.getVideoTracks()[0].stop();

div.remove();

return canvas.toDataURL('image/jpeg', quality);

}

''')

display(js)

data = eval\_js('takePhoto({})'.format(quality))

binary = b64decode(data.split(',')[1])

with open(fileName, 'wb') as f:

f.write(binary)

return fileName

#(Google Colab: Access Webcam for Images and Video, n.d.)

#Reference: Google Colab: Access Webcam for Images and Video. (n.d.). Retrieved June 12, 2023, from https://colab.research.google.com/

#imread function

from cv2 import imread

# import cascade

from cv2 import CascadeClassifier

def predictImage(imagePath):

# load the photograph from path

takenImage = imread(imagePath)

# load the harcascade model

imageClassifier = CascadeClassifier('/content/drive/MyDrive/AI/haarcascade\_frontalface\_default.xml')

# perform face detection

faceBoxes = imageClassifier.detectMultiScale(takenImage,scaleFactor=1.1, minNeighbors=2, minSize=(30, 30))

# print bounding box for each detected face

for faceBox in faceBoxes:

# extract faces region

x, y, width, height = faceBox

x2, y2 = x + width, y + height

# draw a rectangle over the pixels

# rectangle(pixels, (x, y), (x2, y2), (0,0,255), 1)

#extract the detected image

takenImage = takenImage[y:y2, x:x2]

#resize the obtained image

takenImage = cv2.resize(takenImage, (64,64))

# displays that image

cv2\_imshow(takenImage)

# show the image

#" convert into array

test\_image=image.img\_to\_array(takenImage)

#expands the dimenstion

test\_image=np.expand\_dims(test\_image,axis=0)

# prediction is performed here

result=modelClassifier.predict(test\_image,verbose=0)

# print(training\_set.class\_indices)

#get the name of student from result

studentName = MapForFaces[np.argmax(result)]

#return student name

return studentName

#this functions checks the captured images and call for prediction

def checkPrediction():

#image path

ImagePath='/content/photo.jpg'

# ImagePath = '/content/sample\_data/preprocessed\_dataset/validation/face5/face5\_1\_19.jpg'

# ImagePath = '/content/sample\_data/preprocessed\_dataset/train/face2/face2\_1\_0.jpg'

# ImagePath = '/content/drive/MyDrive/AI/datasets/train/face3/face3\_2.jpeg'

# ImagePath = '/content/drive/MyDrive/AI/datasets/train/face6/face6\_2.jpg'

studentName = predictImage(ImagePath)

# Pring the student name

print('Attendant is: ', studentName)

# to make no duplicate entry of student

if studentName in students:

#if student name is available in the list remove it so he/she cannot renter during

#that day

print('hello')

students.remove(studentName)

#getting current timedate object value which is created while making attendance.csv file

currentTimeDate = datetime.now()

# getting time value

currentTimeVal = currentTimeDate.strftime("%H-%M-%S")

# write the name and time in csv file

attendanceWriter.writerow([studentName, currentTimeVal])

# save the record

attendanceFile.flush()

import numpy as np

import keras.utils as image

from IPython.display import Image

try:

#call the camera

imageTaken = onlineTakePhoto()

# chedck prediction

checkPrediction()

# Show the image which was just taken.

except Exception as error:

# errors will be displayed

print(str(error))

# run this function to check the current student captured photo

checkPrediction()

#total students names after attendance

print(students)