Facial Recognition for Computer Labs
Final Report

Group 39
Diqing Zuo
Rohan Sreenivasan

1. Introduction

1.1 Statement of Purpose

The objective of this project is to grant students the access to a computer lab based on facial recognition. It would compare the real-time input and the pictures of students we have in our database to decide if or not access is granted to the certain student, and assign those with access to a desktop computer in the lab. It will be useful to automate the way students use school resources all while holding full accountability.

1.2 Features and benefits

a. Features:

The product is able to detect faces through real-time inputs of a USB camera connected to the Jetson Nano. It would be placed at the entrance of computer labs such that every student that wants access would have to request it by getting their face scanned before entering the lab.

Our product would then crop the video inputs into pictures of faces with dimensions 96*96. These images of student faces would then be compared with the face images of students stored in the database to see if they match the face of a student that has been granted access to the computer lab. If access is granted to them, the LED board connected to Jetson Nano should display the location of their assigned computer as well as the name and netID of the student that's allowed into the computer lab. The step motor that is also connected to

Jetson Nano would then automatically open a gate that signals access to the computer lab is granted to the student scanned.

b. Benefits:

This product allows the university to automate the process of granting students access to school resources and also being able to keep track of detailed information on the use of school resources, such as which student was using the computer at what time. This would be helpful in the cases of unexpected damage to public resources because the university could then track down the person that's responsible for them.

Another benefit this would bring is that it brings convenience to students who want to access computer labs because they no longer have to bring an I-card with them every time they want to enter one.

2. Design

2.1 System overview

The project utilizes jetson nano as the core control to run the code based on an pretrained resnet model from network for facial recognition purposes. It is then connected to an LED board which would act as the display board for directions for assigned computer and personal information of students. After the system decides that the student is granted access to the lab, we would then use Jetson Nano to control the step motor that would raise a mini gate to indicate that access is granted.

2.2 Design Details

Our project utilizes MTCNN (Multi-Task Cascading Convolutional Neural Network) inside the facenet pytorch package to create a face tracking system. The facial recognition process is divided into a few stages.

The first stage was to crop the video inputs of students' faces. We decided to use the pretrained model “vggfaces2”. Since the system would work better if applied to the images that are resize to the same dimensions of the images it was trained on (Esler, 2021), and pretrained model we’ve decided to use was trained on 96*96 pixels images, we would have to crop the camera inputs into images of only faces with dimensions of 96 * 96 pixels.

Secondly, we would need to apply the cropped images of faces to resnet (The version we’re using is Inception Resnet 1). In the case of our project, we are trimming the last two layers of the resnet because they are not needed.

Then we look for the distance between the image we cropped and images of different student faces we have in the database. After we get all the distance values, we compare them and decide that the face with the smallest distance value would be the matched student in the database.

Below is our python code for facial recognition to be implemented on Jetson Nano:

```
#import facenet_pytorch
from facenet_pytorch import MTCNN, InceptionResnetV1
from PIL import Image
import numpy
import torch
import cv2
import matplotlib.pyplot as plt
#from scipy import ndimage, misc
import os
import serial
import time
arduino = serial.Serial(port='COM3', baudrate=115200, timeout=.1)

# python function to write data to the arduino through the serial


def write_read(x):
    arduino.write(bytes(x, 'utf-8'))
    time.sleep(0.5)
    return arduino.read(6)
```

```
resnet = InceptionResnetV1(pretrained='vggface2').eval()
resnet.last_bn = torch.nn.Identity()  # take away last 2 layers
resnet.logits = torch.nn.Identity()

resnet.classify = True

# If required, create a face detection pipeline using MTCNN:
capture = cv2.VideoCapture(0, cv2.CAP_DSHOW)
detector = MTCNN(image_size=96, margin=10)
count = 0

while capture.isOpened():
    ret, frame = capture.read()
  #   cv2.imshow('window-name', frame)
    x = 0
    result = detector.detect(frame)
    #img = numpy.array(result)
    coords, conf = result
    if coords is None:
        continue
    x1, y1, x2, y2 = [int(x) for x in coords[0]]
    #print(x1, y1, x2, y2)

    cv2.imshow('frame', frame[y1:y2, x1:x2])
    # cv2.imwrite("frame%d.jpg" % count, frame)
    count = count + 1
   # print(result)
    if (conf.any() > .98):

        outPath = r"C:\Users\rohan\Documents\Honors Lab\CV
project\res"
        path = r"C:\Users\rohan\Documents\Honors Lab\CV
project\images"
        img_embedding = result
    # iterate through the names of contents of the folder

        data = []
        paths = []
        for image in os.listdir(path):
            with Image.open(image) as im:
```

```
                img_cropped = detector(
                    im, save_path=r'C:\Users\rohan\Documents\Honors
Lab\CV project\res\test.jpg')
                img_embedding = resnet(img_cropped.unsqueeze(0))

                img2 = frame[y1:y2, x1:x2]
                img2 = torch.from_numpy(img2).permute(
                    [2, 0, 1]).unsqueeze(0).float()
                img2_embedding = resnet(img2)
                dist = torch.cdist(img_embedding, img2_embedding, 1)
                data.append(dist)
                paths.append(image)

                dist = torch.cdist(img_embedding, img2_embedding, 1)
               # print(dist)
        closest_path = paths[torch.argmin(torch.tensor(data))]
        print(closest_path)
        val = torch.tensor([dist])
        num = val.item()
        # print(num)
        if (dist < 900):
            num = input("Enter your password: ")  # Taking input
from user
            print(write_read(num))  # printing the value
        exit(0)

    if cv2.waitKey(10) & 0xFF == ord('q'):
        break

capture.release()
cv2.destroyAllWindows()
model = InceptionResnetV1(pretrained='vggface2').eval()
cv2.destroyAllWindows()
```

Arduino Code:

```
#include <Servo.h>
Servo myservo;
 int x;
void setup() {

    Serial.begin(115200);

   myservo.attach(9);
    myservo.write(0);
    delay(1000);
    myservo.detach();
   pinMode(13, OUTPUT);
   pinMode(12, OUTPUT);

}

void loop() {
    // send data only when you receive data:

while (!Serial.available());{
 x = Serial.readString().toInt();

 if (x == 1234){
    Serial.print("match");
    myservo.attach(9);
    delay(1000);
     myservo.write(90);
    delay(3000);
    myservo.write(0);
    delay(1000);
    myservo.detach();

    x = 0;
 }
 }


 }
```

3. Results

Below are our results for facial recognition (comparing cropped images of faces from video feed with faces in the database).

Because we encountered a problem with using Jetson Nano, we had to use a laptop to run the python codes instead, and for the motor that controls the mini gate, an arduino was used to read signals generated by the laptop after the facial recognition process and operate the gate accordingly.
We also had to change the motor we use because we placed a wrong order and bought a motor driver breakout board instead of a stepper motor.

Arduino Codes would be attached in the appendix.

4. Problems & Challenges

Problems that we've encountered include:

- Finding faces in real-time video inputs
- Getting the relatively right distance value for face comparison so that the system could recognize faces.
- Jetson Nano required Wifi, so we decided to use the laptop to run python code and an arduino board to control the motor instead.

5. Future Plans

We plan to find a way of implementing Jetson Nano instead of needing to use both laptop and arduino to control the facial recognition system and motor separately.

6. References

1. Esler, T., 2021. *facenet-pytorch*. [online] PyPI. Available at: <https://pypi.org/project/facenet-pytorch/> [Accessed 11 December 2021].

2.

7. Appendix

Arduino code: CV-Project/motor.ino at main · rohansreenivasan5/CV-Project (github.com)

Python code: [CV-Project/model.py at main · rohansreenivasan5/CV-Project (github.com)]

test1 | Arduino 1.8.16

File Edit Sketch Tools Help


```
/*

*/
#include <Servo.h>
Servo myservo;
 int x;
void setup() {

    Serial.begin(115200);

   myservo.attach(9);
    myservo.write(0);
    delay(1000);
    myservo.detach();
   pinMode(13, OUTPUT);
   pinMode(12, OUTPUT);

}

void loop() {
    // send data only when you receive data:
    digitalWrite(13, HIGH);
    digitalWrite(12, HIGH);
while (!Serial.available());{
 x = Serial.readString().toInt();

 if (x = 1234){
 myservo.attach(9);
    Serial.print("match\n");
    delay(1000);
    digitalWrite(13, HIGH);
    delay(1000);
     digitalWrite(12, HIGH);
    delay(1000);
     myservo.write(90);
    delay(1000);
    myservo.write(0);
    delay(1000);
   // digitalWrite(13, LOW);
    delay(1000);
    myservo.detach();

    x = 0;
 }
 }


 }
```