Hệ thống Face Recognition

**1. Face-api**

Face-api là một thư viện giúp cho chúng ta thực hiện các công việc như phát hiện khuôn mặt và nhận diện khuôn mặt trên trình duyệt và nó được triển khai trên lõi của[*tensorFlow.js*](https://www.tensorflow.org/js)

Trước khi bắt đầu với việc làm demo thì mình muốn giới thiệu một chút về các model của thư viện Face-api.js. Hiện tại thì chúng ta có thể sử dụng được các model mà face-api cung cấp đó là:

Face Detection: Nó sẽ trả về một ô vuông giới hạn khuôn mặt và xác suất cho mỗi gương mặt mà nó phát hiệnđược

Face Landmark Detection: nó giúp phát hiện 68 điểm trên khuôn mặt

Face Recognition: Xử dụng cho việc nhận diện khuôn mặt, một kiến trúc gần giống như ResNet-34 được triển khai để tính toán một bộ mô tả khuôn mặt (mộ vectơ đặc trưng có 128 giá trị) từ các hình ảnh khuôn mặt đã được cung cấp

**2. Sử dụng face-api trong server của NodeJS:**

Training dữ liệu:

Import thư viện face-api

*const* faceapi = *require*('face-api.js');

Tiếp theo chúng ta sẽ phải load các models của thư viện cần dùng như ở đây chúng ta cần model faceRecognitionNet, faceLandmark68Net và ssdMobilenetv1. Ở đây chúng ta dùng Promise là để đợi cho đến khi load xong cả 3 models kia thì mới thực hiện tiếp.

 await Promise.all([

            faceapi.nets.ssdMobilenetv1.loadFromDisk('./modelsTraining'),

            faceapi.nets.faceLandmark68Net.loadFromDisk('./modelsTraining'),

            faceapi.nets.faceRecognitionNet.loadFromDisk('./modelsTraining')

        ])

Tiến hành training dữ liệu có trong folder uploads, với mỗi folder là id của khách hàng và bên trong là dữ liệu hình .jpeg

async *function* loadTrainingData(){

*const* faceDescriptors = []

    for(*const* file of files){

        console.log(`training ${file}...`)

*const* descriptors = []

*const* imagefolders = fs.readdirSync(`./uploads/${file}`)

        try {

            for(*const* imagefolder of imagefolders){

*const* image = await canvas.loadImage(`./uploads/${file}/${imagefolder}`);

*const* detection = await faceapi.detectSingleFace(image).withFaceLandmarks().withFaceDescriptor()

                descriptors.push(detection.descriptor)

            }

*const* newLabeledFaceDescriptors = new faceapi.LabeledFaceDescriptors(file, descriptors)

            faceDescriptors.push(newLabeledFaceDescriptors)

            console.log(`${file} done`)

        } catch (error) {

            console.log(error)

        }

    }

    return faceDescriptors

}

*const* trainingData = await loadTrainingData()

Dữ liệu được training xong sẽ được lưu lại dưới dạng JSON:

*let* data = *JSON*.*stringify*(trainingData);

        fs.writeFileSync('./faceRecognition/trainingData.json', data);

Sử dụng dữ liệu đã đã và nhận diện khuông mặt:

Đọc dữ liệu định dạng JSON được lưu và format lại theo chuẩn của face-api là Float32Array, sau đó gán vào biến tên là ‘faceMatcher’ :

async *function* init() {

        await Promise.all([

            faceapi.nets.ssdMobilenetv1.loadFromUri('/models'),

            faceapi.nets.faceLandmark68Net.loadFromUri('/models'),

            faceapi.nets.faceRecognitionNet.loadFromUri('/models')

        ])

*const* content = faceData

        for (*var* x = 0; x < Object.keys(content).length; x++) {

            for (*var* y = 0; y < Object.keys(content[x].descriptors).length; y++) {

*var* results = Object.values(content[x].descriptors[y]);

                content[x].descriptors[y] = new Float32Array(results);

            }

        }

        faceMatcher = await createFaceMatcher(content)

        setSubFaceMatcher(faceMatcher)

    }

Nhận diện khuông mặt từ faceMatcher:

*const* detection = await faceapi.detectSingleFace(e).withFaceLandmarks().withFaceDescriptor()

*const* face = faceMatcher.findBestMatch(detection.descriptor)

biến ‘face’ là kết quả trả về sau khi nhận diện dưới dạn một object với property ‘\_label’ chính là id của khách hàng