face-api.js : A way to build a Face Recognition system in the browser.

Jeeva Saravanan · Follow
Published in TheLeanProgrammer
4 min read · Jul 28, 2021

A **facial recognition system** is a technology capable of matching a human face from a digital image or a video frame against a database of faces, typically employed to authenticate users through ID verification services, works by pinpointing and measuring facial features from a given image.

This article will focus on How to implement a Face Recognition System in Browser using JavaScript. To implement this we are gonna use face-api JavaScript Library.

Pinterest

What is face-api.js ?

face-api is a JavaScript library created by Vincent Mühler., to detect faces via browser. It is built over tensorflow.js core API. It supports Face Detection, Face Recognition, Face Expression, Age, and Gender Detection.

Installing face-api.js

You can clone the following face-api repository

```
git clone https://github.com/justadudewhohacks/face-api.js.git
```

justadudewhohacks/face-api.js

JavaScript face recognition API for the browser and nodejs implemented on top of tensorflow.js core...

github.com

if you have npm installed in your system

```
npm i face-api.js
```

or you can use

GitHub - JeevaSaravanan/Face-Recognition-With-Face-Api: Face Recognition system with Face API in...

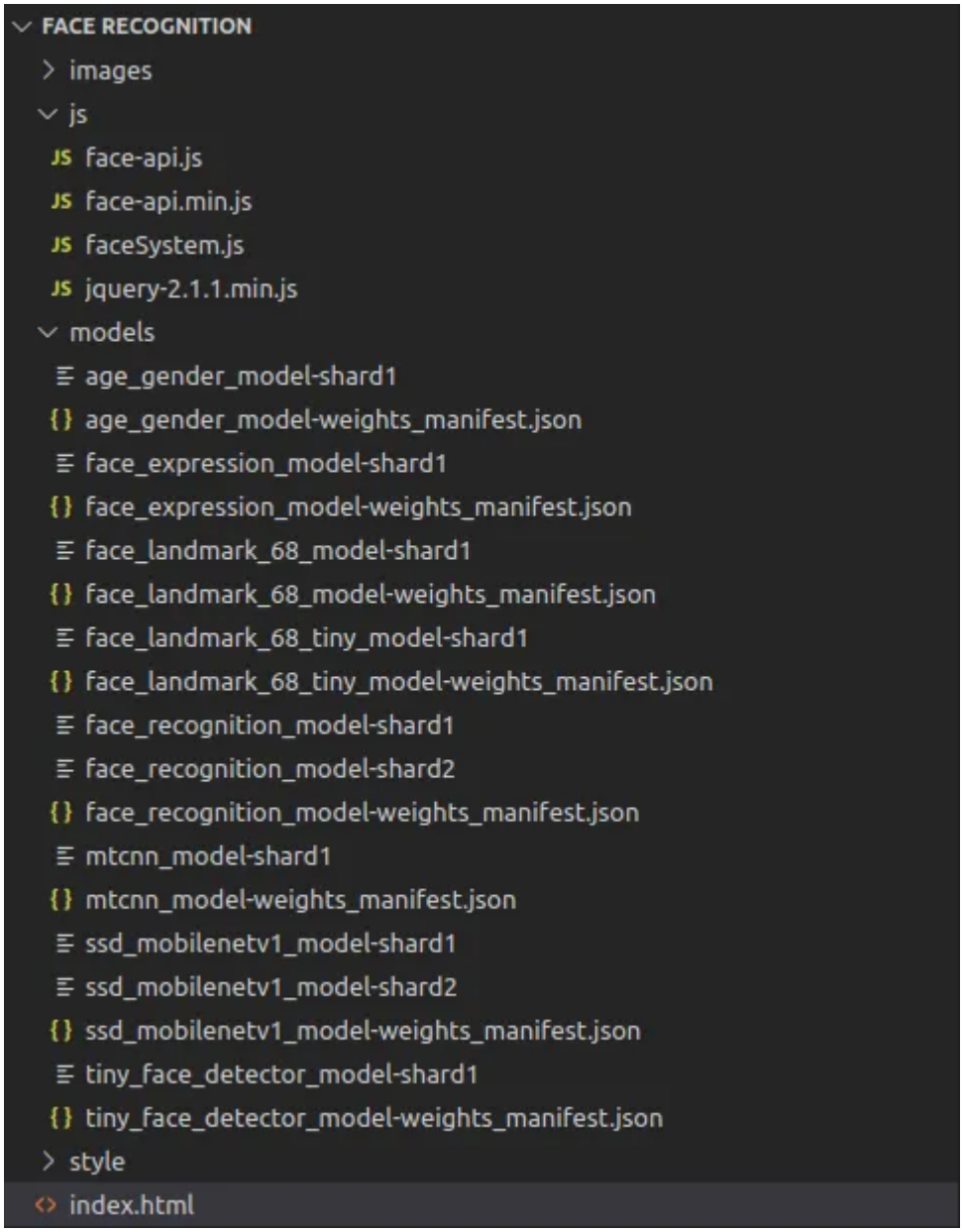
Face Recognition system with Face API in browser. Contribute to JeevaSaravanan/Face-Recognition-With-Face-Api...

github.com

Let's get started

Folder Structure

Before diving into code, let's organize the folder and files nice and clean so that it is more maintainable.



Folder Structure

- images — contains images
- js — contains js file

i) face-api.js / face-api.min.js — face detection and recognition javascipt library

ii) faceSystem.js — application js file

iii) jquery-2.1.1.min.js — jquery Library

- models — contains models from face-api.js

- styles — contains css file
- index.html

If you have cloned face-api repo then, models are present in weights folder and you can find face-api.js/face-api.min.js file under dist folder.

Here we begin,

Creating HTML Page

Create an HTML page with image and canvas. Canvas is used to draw in the browser using HTML and JavaScript, this is required to draw the bounding box and the facial landmarks.

```
1       <body>
2           <img src="images/friendsreunion.jpg" id="originalImg"/>
3           <canvas id="reflay" class="overlay"></canvas>
4       </body>
```

image.html hosted with ❤ by GitHub view raw

Include the necessary script

```
1   <!-- JS Library -->
2   <script src="js/jquery-2.1.1.min.js"></script>
3   <script src="js/face-api.js"></script>
4
5   <!-- Application JS -->
6   <script src="js/faceSystem.js"></script>
```

script.html hosted with ❤ by GitHub view raw

Final index.html,

```
1   <html>
2       <head>
3           <title>Face App</title>
4           <style>
5           #overlay, .overlay {
6               position: absolute;
7               top: 0;
```

```
            top: 0;
            left: 0;
            }
        </style>
    </head>
    <body>
        <img src="images/friendsreunion.jpg" id="originalImg" />
        <canvas id="reflay" class="overlay"></canvas>

        <script src="js/jquery-2.1.1.min.js"></script>
        <script src="js/face-api.js"></script>
        <script src="js/faceSystem.js"></script>
    </body>
</html>
```

index.html hosted with ❤ by GitHub view raw

index.html

Load the models

The next would be loading the model. `loadSsdMobilenetv1Model` is used to load the `ssd` object detection model, `loadFaceLandmarkModel` to detect the facial landmark, `loadFaceRecognitionModel` to detect the facial recognition and `loadFaceExpressionModel` to detect the facial expression

```
const MODEL_URL = '/models' //model directory

await faceapi.loadSsdMobilenetv1Model(MODEL_URL)
await faceapi.loadFaceLandmarkModel(MODEL_URL) // model to detect face landmark
await faceapi.loadFaceRecognitionModel(MODEL_URL) //model to Recognise Face
await faceapi.loadFaceExpressionModel(MODEL_URL) //model to detect face expression
```

loadModel.js hosted with ❤ by GitHub view raw

One of the advantages of using face-api library is that it has already trained model(pre-trained Model).

Adding Bounding Box

After loading the models, get the image and canvas. To get the Description of the face, pass it via model.

```
const img= document.getElementById('originalImg')
let faceDescriptions = await faceapi.detectAllFaces(img).withFaceLandmarks().withFaceDescriptors().
const canvas = $('#reflay').get(0)
faceapi.matchDimensions(canvas, img)
```

faceDescription.js hosted with ❤ by GitHub view raw

Draw bounding box and landmarks for the recognized face with `drawDetections` and `drawFaceLandmarks` . `drawFaceExpressions` to recognize facial expressions. Before drawing make sure to resize the Face Description.

```
faceDescriptions = faceapi.resizeResults(faceDescriptions, img)
faceapi.draw.drawDetections(canvas, faceDescriptions) //to draw box around detection
faceapi.draw.drawFaceLandmarks(canvas, faceDescriptions) //to draw face landmarks
faceapi.draw.drawFaceExpressions(canvas, faceDescriptions) //to mention face expression
```

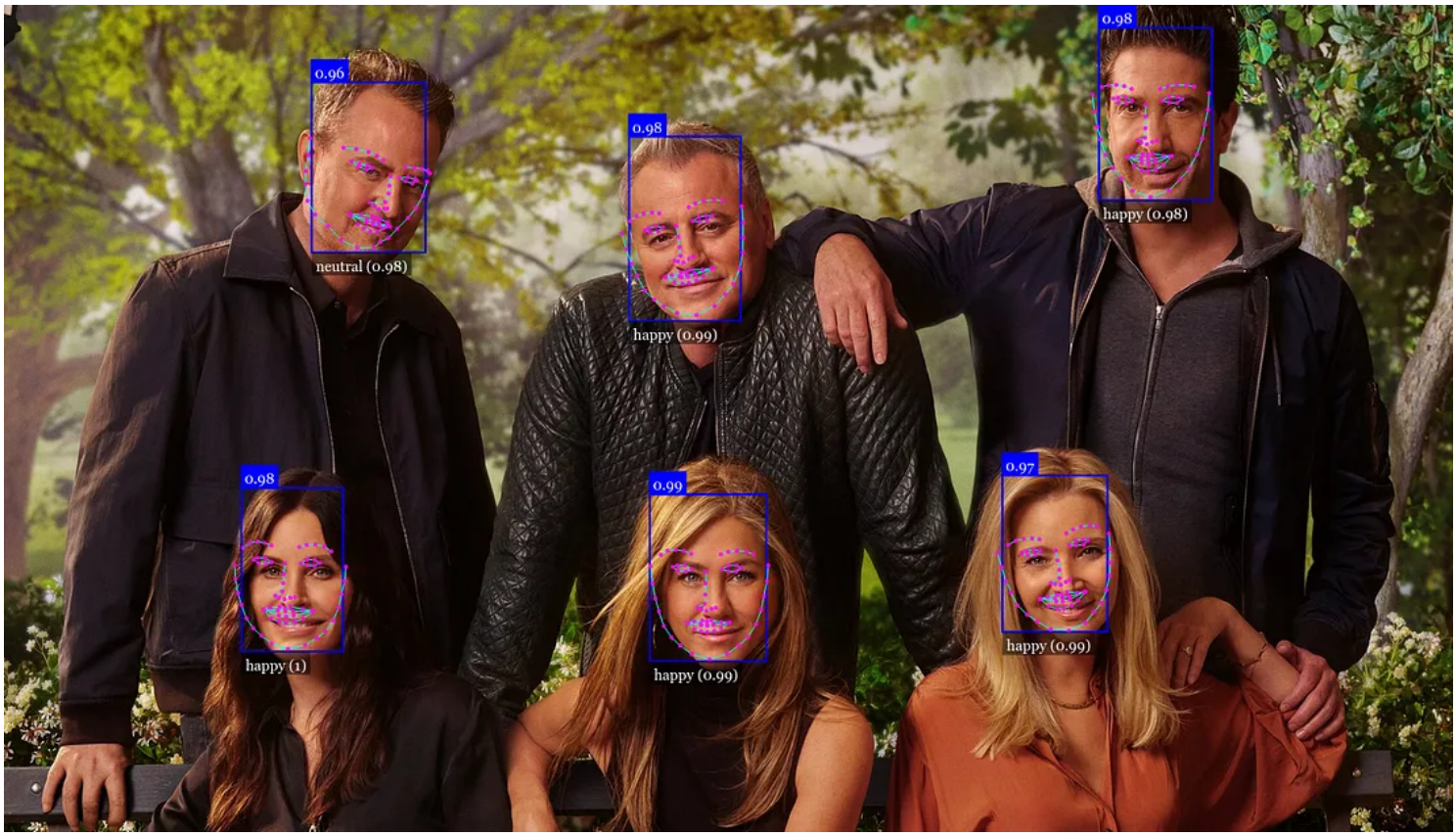
boundingBox.js hosted with ❤ by GitHub view raw

Putting all this together,

```
$(document).ready(function(){

    async function face(){

        const MODEL_URL = '/models'

        await faceapi.loadSsdMobilenetv1Model(MODEL_URL)
        await faceapi.loadFaceLandmarkModel(MODEL_URL)
        await faceapi.loadFaceRecognitionModel(MODEL_URL)
        await faceapi.loadFaceExpressionModel(MODEL_URL)

        const img= document.getElementById('originalImg')
        let faceDescriptions = await faceapi.detectAllFaces(img).withFaceLandmarks().withFaceDescr
        const canvas = $('#reflay').get(0)
        faceapi.matchDimensions(canvas, img)

        faceDescriptions = faceapi.resizeResults(faceDescriptions, img)
        faceapi.draw.drawDetections(canvas, faceDescriptions)
        faceapi.draw.drawFaceLandmarks(canvas, faceDescriptions)
        faceapi.draw.drawFaceExpressions(canvas, faceDescriptions)

    }

    face()
})
```

faceDetection.js hosted with ❤️ by GitHub view raw

> *You may get a result like this.* To align the face landmark with your image, use CSS



rResult

Face Recognition

Now let's try to recognize the face for the above image.

Create an array `label` which holds the names of the characters in the image. Name the reference image the same as the label. `fetchImage` could be used to load images from the directories.

Detect single face and compute face description for every image in and store it along with respective label with `LabeledFaceDescriptors` .

```
const labels = ['ross', 'rachel', 'chandler', 'monica', 'phoebe', 'joey']

const labeledFaceDescriptors = await Promise.all(
    labels.map(async label => {
      const imgUrl = `images/${label}.jpg`
      const img = await faceapi.fetchImage(imgUrl)

      const faceDescription = await faceapi.detectSingleFace(img).withFaceLandmarks().withFaceDesc
      if (!faceDescription) {
        throw new Error(`no faces detected for ${label}`)
      }

      const faceDescriptors = [faceDescription.descriptor]
      return new faceapi.LabeledFaceDescriptors(label, faceDescriptors)
    })
);
```

detect.js hosted with ❤ by GitHub view raw

Match the input face with reference face description with Euclidean Distance. The threshold value for the Euclidean Distance could be set. On setting threshold value, we define the reference image to be matched should have at most the specific threshold value.

`faceapi.FaceMatcher()` takes in the reference feature descriptor and the threshold to initialize the object to calculate the Euclidean distance and apply the threshold. This is done via the `faceMatcher.findBestMatch()` function by looping through all the features detected for each face in the input image.

```
const threshold = 0.6
const faceMatcher = new faceapi.FaceMatcher(labeledFaceDescriptors, threshold)

const results = faceDescriptions.map(fd => faceMatcher.findBestMatch(fd.descriptor))

//label images
results.forEach((bestMatch, i) => {
    const box = faceDescriptions[i].detection.box
    const text = bestMatch.toString()
    const drawBox = new faceapi.draw.DrawBox(box, { label: text })
    drawBox.draw(canvas)
})
```

Match.js hosted with ❤ by GitHub view raw

Summing up all the snippets,

```
$(document).ready(function(){

    async function face(){

        const MODEL_URL = '/models'

        await faceapi.loadSsdMobilenetv1Model(MODEL_URL)
        await faceapi.loadFaceLandmarkModel(MODEL_URL)
        await faceapi.loadFaceRecognitionModel(MODEL_URL)
        await faceapi.loadFaceExpressionModel(MODEL_URL)

        const img= document.getElementById('originalImg')
        let faceDescriptions = await faceapi.detectAllFaces(img).withFaceLandmarks().withFaceDescr
        const canvas = $('#reflay').get(0)
        faceapi.matchDimensions(canvas, img)

        faceDescriptions = faceapi.resizeResults(faceDescriptions, img)
        faceapi.draw.drawDetections(canvas, faceDescriptions)
        faceapi.draw.drawFaceLandmarks(canvas, faceDescriptions)
        faceapi.draw.drawFaceExpressions(canvas, faceDescriptions)


        const labels = ['ross', 'rachel', 'chandler', 'monica', 'phoebe', 'joey']

        const labeledFaceDescriptors = await Promise.all(
            labels.map(async label => {

                const imgUrl = `images/${label}.jpg`
                const img = await faceapi.fetchImage(imgUrl)

                const faceDescription = await faceapi.detectSingleFace(img).withFaceLandmarks().wi

                if (!faceDescription) {
                throw new Error(`no faces detected for ${label}`)
                }

                const faceDescriptors = [faceDescription.descriptor]
                return new faceapi.LabeledFaceDescriptors(label, faceDescriptors)
            })
        );

        const threshold = 0.6
        const faceMatcher = new faceapi.FaceMatcher(labeledFaceDescriptors, threshold)

        const results = faceDescriptions.map(fd => faceMatcher.findBestMatch(fd.descriptor))
```

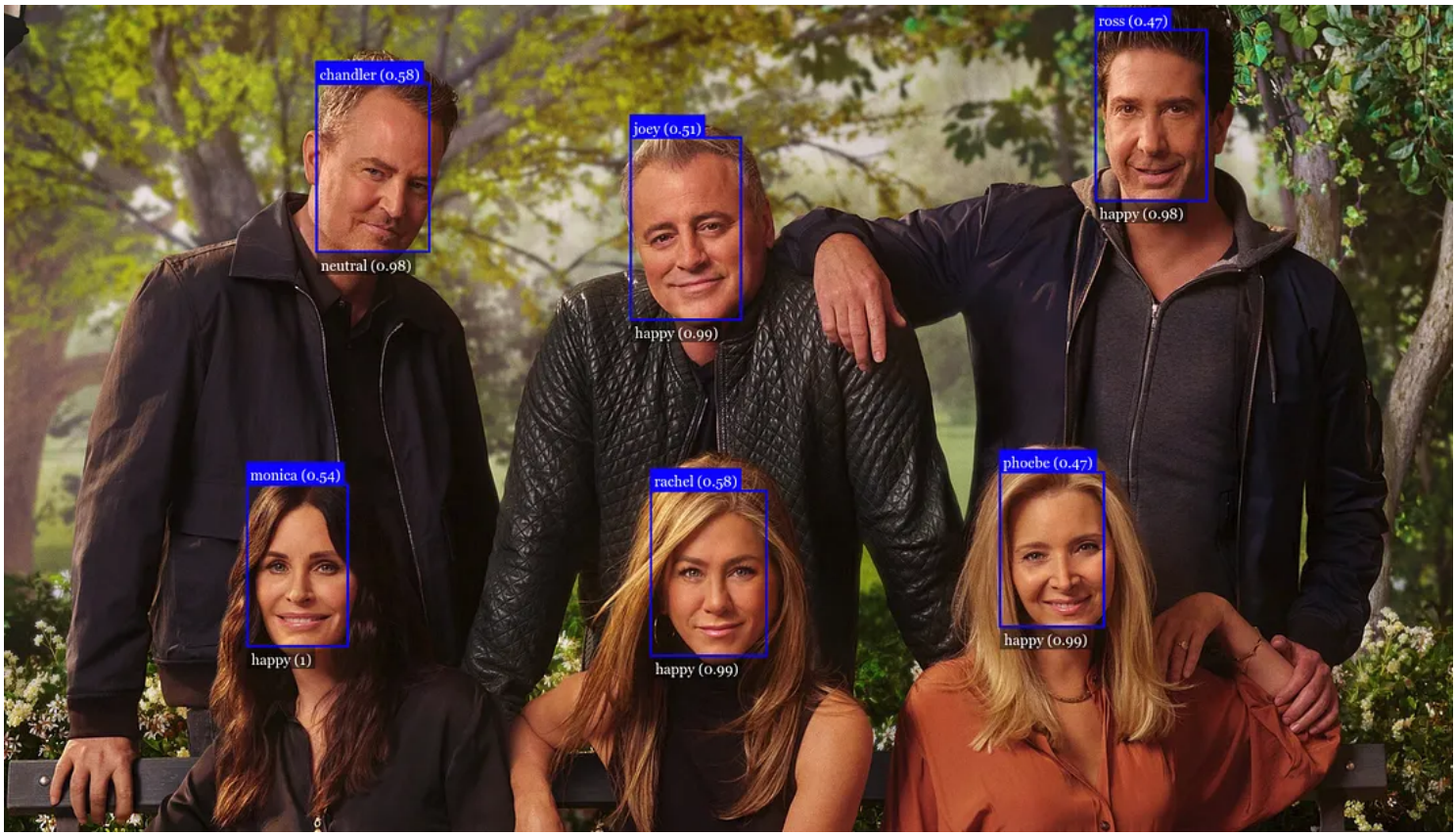
```
        const results = faceDescriptions.map(fd => faceMatcher.findBestMatch(fd.descriptor))

        results.forEach((bestMatch, i) => {
            const box = faceDescriptions[i].detection.box
            const text = bestMatch.toString()
            const drawBox = new faceapi.draw.DrawBox(box, { label: text })
            drawBox.draw(canvas)
        })

    }

    face()
})
```

faceSystem.js hosted with ❤ by GitHub view raw

Resulting image,


ross (0.39)
chandler (0.55)
happy (0.99)
phoebe (0.52)
rachel (0.5)
happy (1)
happy (0.44)
neutral (0.34)
sad (0.17)
joey (0.49)
monica (0.48)
happy (1)
happy (0.99)

Result



Result

> *You can also try this project for face detection via Webcam.*

Like to dig deeper into this?

face-api.js — JavaScript API for Face Recognition in the Browser with tensorflow.js

A JavaScript API for Face Detection, Face Recognition and Face Landmark Detection

itnext.io

Face Recognition Using JavaScript API — face-api.js

In this article, we will learn about face detection (Age/Gender/Face Positions/Mood) using face-api.js and the nearby...

towardsdatascience.com

face-api.js

Edit description

justadudewhohacks.github.io

Have any suggestions to improve this project? Share them with us in the comment section below.

Thanks for Reading!!

Don’t forget to follow The Lean Programmer Publication for more such articles, and subscribe to our newsletter tinyletter.com/TheLeanProgrammer

Face Api | Javascript Face Api | Face Recognition System | JavaScript | Artificial Intelligence

Follow

Written by Jeeva Saravanan

13 Followers · Writer for TheLeanProgrammer

Developer @Zoho Corporation | Machine Learning Enthusiast and Coder.

More from Jeeva Saravanan and TheLeanProgrammer

Jeeva Saravanan in Analytics Vidhya

How to Evaluate your Machine Learning Model.

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve...

11 min read · May 29, 2021

8 1

Mohsin Raza in TheLeanProgrammer

Roadmap for Machine Learning

Transform from Zero to Hero

· 5 min read · Sep 4, 2021

251 1

NamyaLG in TheLeanProgrammer

Connecting Firebase

Complete guide with detailed steps to connect your real-time database on Firebase to Python.

3 min read · Jun 17, 2021

105

Jeeva Saravanan in TheLeanProgrammer

Git Cheat Sheet—A simple guide to learn git commands.

This article is simple guide to few basic and necessary git commands one should know.

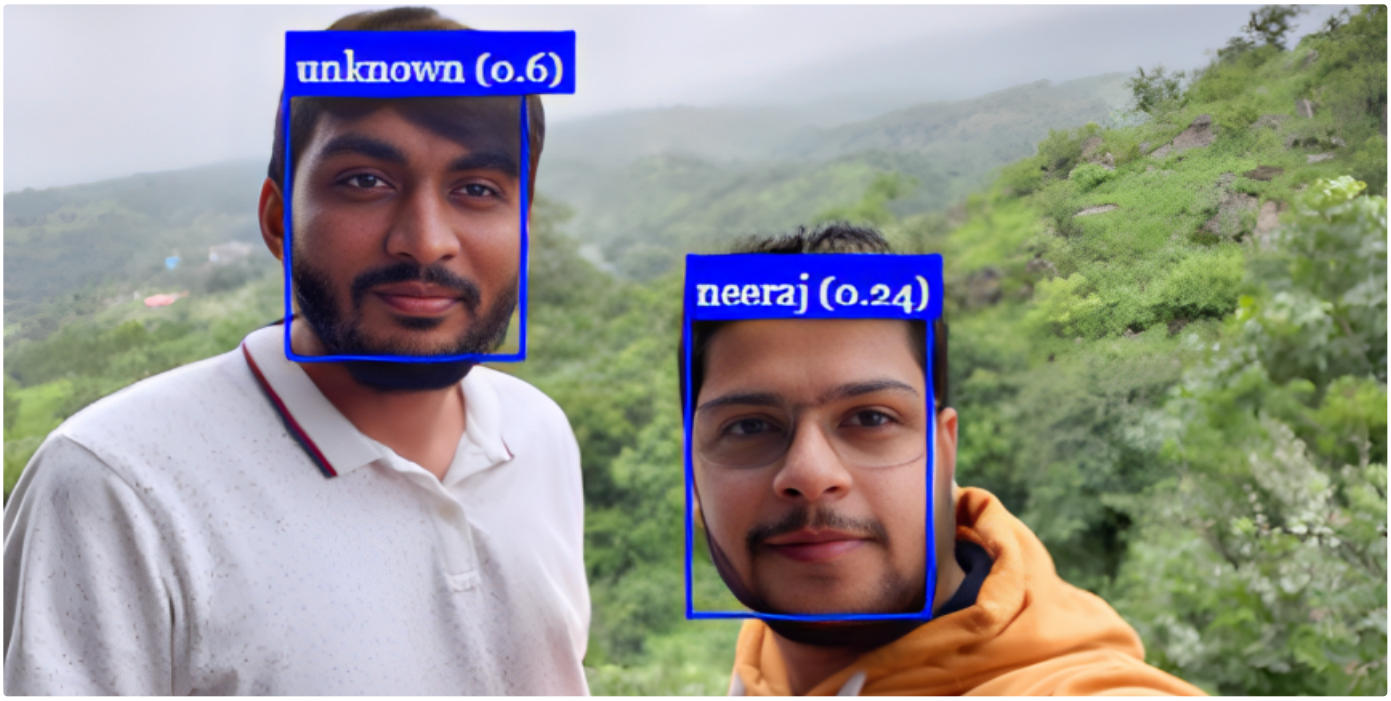
6 min read · Oct 10, 2021

2

See all from Jeeva Saravanan

See all from TheLeanProgrammer

Recommended from Medium



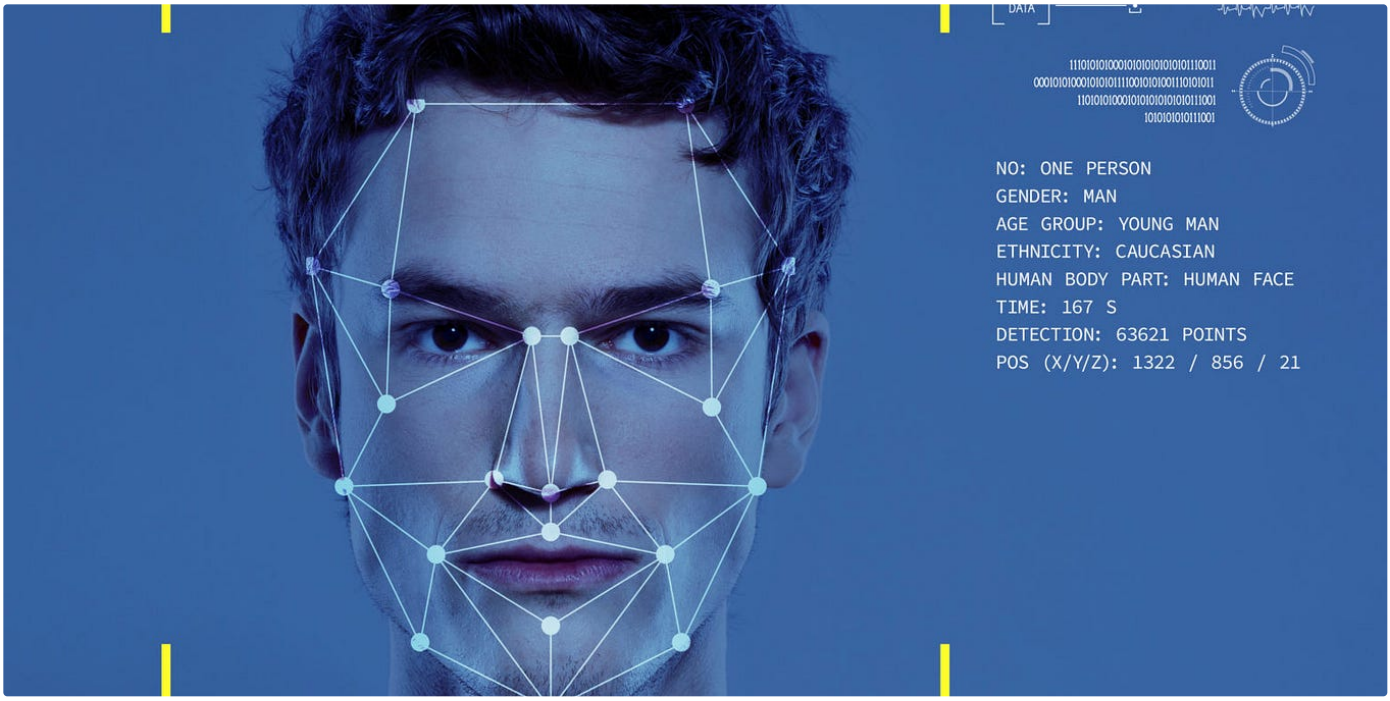
NEERAJ VAGEELE

Build your own face recognition web app using face-api.js

Face recognition is a technology that identifies or verifies individuals by analyzing and comparing their facial features with a database...

4 min read · Jul 3

10 4



Junhee Park

Bias in Facial Recognition

Junhee Park, Ji Min Sung, Bada Lee, and Ha Eun Lee

8 min read · Mar 7

50 1

Lists

AI Regulation

6 stories · 78 saves

ChatGPT

21 stories · 109 saves

ChatGPT prompts

24 stories · 252 saves

Generative AI Recommended Reading

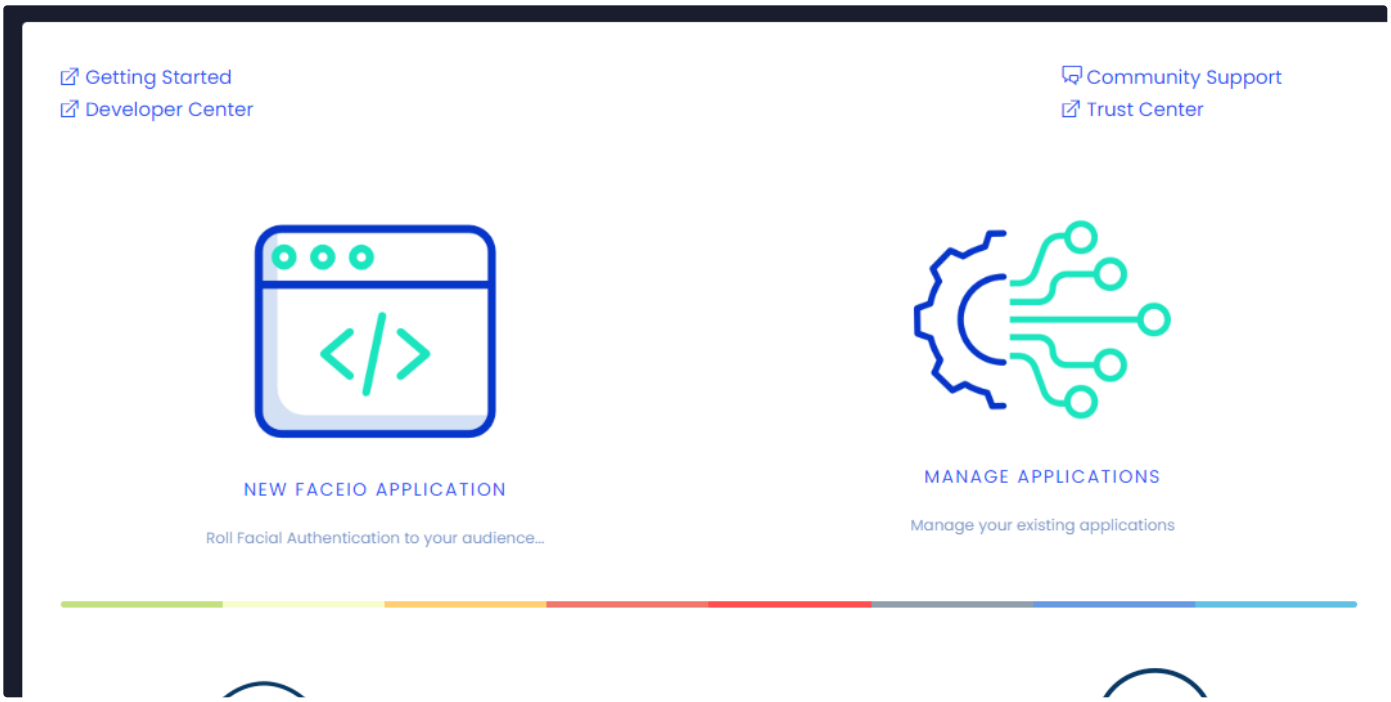
52 stories · 161 saves

Yahya Shareef

how to build your own image recognition AI with tensorflow js

first you need to know the basics of web development like html, css and JavaScript

2 min read · Apr 8



Pius Oruko in Level Up Coding

Implementing FaceIO for Facial Recognition in React: A Step-by-Step Guide

Facial recognition has become an increasingly popular technology in recent years, with numerous applications in security, authentication...

6 min read · May 4

68 1

D Monty

Face Recognition System for Surveillance and Security Applications using Python

Discover the power of a face recognition system developed using Python for surveillance and security applications. Unlock advanced facial...

4 min read · Jun 14

Anas Ghareib in CoinsBench

How to integrate a smart contract into a react-native project in 5 easy steps

First thing first, To integrate a smart contract into a React Native project, you will need to use the web3.js library.

2 min read · Mar 18

9

See more recommendations