Multiple Face Detection and Recognition System Design Applying Deep Learning in Web Browsers using JavaScript

Cristhian Gabriel Espinosa Sandoval

Follow this and additional works at: https://scholarworks.uark.edu/csceuht

Part of the Other Computer Engineering Commons

Citation

This Thesis is brought to you for free and open access by the Computer Science and Computer Engineering at ScholarWorks@UARK. It has been accepted for inclusion in Computer Science and Computer Engineering Undergraduate Honors Theses by an authorized administrator of ScholarWorks@UARK. For more information, please contact ccmiddle@uark.edu.
Multiple Face Detection and Recognition System Design
Applying Deep Learning in Web Browsers using JavaScript

An Undergraduate Honors College Thesis

in the

Department of Computer Science and Computer Engineering
College of Engineering
University of Arkansas
Fayetteville, AR
December, 2019

by

Cristhian Gabriel Espinosa Sandoval
# Table of Contents

Abstract ................................................................................................................................. 4

1. Introduction ......................................................................................................................... 4
   1.1 Objectives .................................................................................................................. 4
   1.2 Applications and Uses ............................................................................................... 5

2. Background ......................................................................................................................... 5
   2.1 Deep learning ............................................................................................................ 5
   2.2 Neural Networks ....................................................................................................... 6
       Convolutional Neural Networks (CNNs) ....................................................................... 7
       Depthwise Separable Convolutions ........................................................................... 7
       Densely Connected Convolutional Networks ........................................................... 7

3. Approach: Performing Face Detection with Deep Learning ........................................... 8
   3.1 Face Detection Models ............................................................................................. 8
       SSD MobileNet v1 ....................................................................................................... 8
       Tiny Face Detector ..................................................................................................... 8
       MTCNN (Multi-task Cascaded Convolutional Neural Networks) ............................... 8

4. System Overview ............................................................................................................... 9
   4.1 Architecture ............................................................................................................... 9
   4.2 What is node.js? ....................................................................................................... 10
   4.3 What is tensorflow.js? ............................................................................................. 10
   4.4 What is face-api.js? ................................................................................................. 11

5. Implementation using Node.js and Express .................................................................... 11
   5.1 Setting up the environment for our frontend and backend application .................. 11
   5.2 Loading detection and recognition models .............................................................. 11
   5.3 Inputs ....................................................................................................................... 12
   5.4 Face detection .......................................................................................................... 12
   5.5 Detecting Face landmark points .............................................................................. 12
   5.6 Computing face descriptors .................................................................................... 13
   5.7 Predicting age and gender recognition ..................................................................... 13
   5.9 Performing face recognition by matching descriptors ............................................. 14
   5.10 Displaying the results from detections .................................................................... 15
   5.11 Face detection model options ................................................................................ 15
       SsdMobilenetv1Options ............................................................................................ 15
TinyFaceDetectorOptions ................................................................. 16
MtcnnOptions .............................................................................. 16
Improving accuracy: Face Landmark Detection and Face Alignment ......................................................... 16
5.11 68 Point Face Landmark Detection Models ........................................................... 17
5.12 Using Face Recognition Models ........................................................................... 18
5.13 Euclidean distance ......................................................................................... 18
5.14 Frontend application (User Interface) .................................................................... 21
.............................................................................................................. 21
6. Experimental Results and Analysis ....................................................................................... 22
6.1 Multiple Face Detection (using Face-api.js) ......................................................... 22
6.2 Face expression Recognition model (using Face-api.js) ........................................ 24
6.3 Age and Gender Recognition model (using Face-api.js) ........................................ 26
6.4 Face Extraction (using Face-api.js) ........................................................................ 27
6.5 Comparing detection speed on video face tracking (using Face-api.js) .................. 28
6.6 Video conclusions ............................................................................................... 29
7. Additional improvements .............................................................................................. 30
8. Conclusion ............................................................................................................... 30
References ....................................................................................................................... 31
Appendix ............................................................................................................................. 32
Source code ....................................................................................................................... 32

Age and Gender Recognition ................................................................................................. 32
Face detection ....................................................................................................................... 36
Face Expression Recognition ................................................................................................. 41
Face Extraction ..................................................................................................................... 45
Video Face Tracking ............................................................................................................ 49
Webcam Face Detection ......................................................................................................... 53
Webcam Age and Gender Recognition .................................................................................... 57
Webcam Face Expression Recognition ...................................................................................... 62
Abstract

Deep learning has advanced progressively in the last years and now demonstrates state-of-the-art performance in various fields. In the era of big data, transformation of data into valuable knowledge has become one of the most important challenges in computing. Therefore, we will review multiple algorithms for face recognition that have been researched for a long time and are maturely developed, and analyze deep learning, presenting examples of current research.

To provide a useful and comprehensive perspective, in this paper we categorize research by deep learning architecture, including neural networks, convolutional neural networks, depthwise Separable Convolutions, densely connected convolutional networks, and present brief descriptions of each study. We consider that this research will provide valuable insights and serve as a starting point for other students to apply deep learning approaches in their computer engineering and computer science studies.

1. Introduction

With the continuous expansion of the society as well as an immediate requirement for fast and efficient automatic identity verification, biometric technology has developed rapidly in recent years. As one of the most important biometric identification technologies, face recognition attracts more and more attention due to its applications in a wide range of areas.

This research will review how machine learning, and specifically deep learning has evolved exponentially since its inception around the 1960s and trace the crucial developments leading up to the emergence of the powerful deep learning techniques that have emerged in recent years. The success of deep learning has resulted in a rapid adoption to solved real-world problems. Furthermore, we will discuss practical and theoretical issues of deep learning and suggest future research directions.

1.1 Objectives

- Implement face recognition in an optimum way in terms of runtime onto a web application and run it and optimize it for embedded systems. Various algorithms and methodologies are studied, and hardware resources planning will be done to achieve the goal.
- Compare different face recognition models based on performance and accuracy.
- Use neural networks to map the characteristics of a human face to a face descriptor, also sometimes referred as face embeddings.
- Compute the Euclidean distance between two face descriptors and decide whether two faces are similar based on a threshold value.
1.2 Applications and Uses

- In real-time embedded systems such as digital cameras.
- Neural networks have been used to recognize patterns in videos and images, such as faces, vehicles, anomalies in medical images, and interesting patterns in other type of collected data.
- Development of multiple face recognition (MFR) systems is useful for implementing applications such as bank security management systems.
- Face recognition has several applications such as in the field of security, forensic and requires more accuracy and reliability.

2. Background

This chapter will review some important concepts that are essential to study face recognition techniques, including neural networks and deep learning.

2.1 Deep learning

Deep learning involves a set of automatic learning algorithms that tries to model the highest level of abstraction using computer architectures that allow multiple and iterative nonlinear transformations of data that can be expressed in matrix form or tensor form. The idea of deep learning is based on the concept of a human brain and the interaction of neurons. The term appeared around the 1980s, but until 2012, there was not enough power to push forward this technology and it did not have enough attention [1].

In 2012, a team led by George Dahl won the Merck Molecular Activity Challenge using a multitask deep neural networks to predict the biomolecular target of a drug. This event caused a significant boom in media and more researchers and developers got interest in deep learning [2].

After different publications by scientists, scientific journals, the technology became popular. Today, it has a big number of applications and a significant part of it is in face recognition. Deep
Learning gives us the power to create a biometric software that is capable of identify or verify the identity of a person.

2.2 Neural Networks

Neural Networks are a set of algorithms, modeled loosely after the human brain, these algorithms are designed to be able to recognize patterns. The patterns that they can recognize are numerical, contained in vectors, into which all the real-world data, such as images, sound, text or time, have to be translated [1][7][8].

![Artificial Intelligence Neural Networks Nodes](image)

Figure 2. Artificial Intelligence Neural Networks Nodes

Things were not good for neural networks. The main idea was to combine a group of simple mathematical neurons to compute complicated tasks, not to use a single one. In other words, instead of using just one output layer, to send an input to many neurons called hidden layer, this is because their output works as input to another hidden layer or as an output layer of neurons [1].

Hidden layers are highly useful, in simple terms, hidden layers can find features within data and allow following layer to be able to operate on those features rather than a noisy and large raw data. For instance, a very common neural network task is finding a human face in an image, the first hidden layer could take the raw pixel values and find lines, ovals, circles, and other elements within an image. The consecutive layer would collect the position of these lines, ovals, circles and other elements within the image and use those to find the location of human faces much easier. Until recently, when machine learning techniques were commonly not applied directly to raw data inputs such as images or audio. Machine learning was done using data that already passed through feature extraction. In order to make learning easier, machine learning was done using preprocessed data which included useful features such as angles or shapes that had been already extracted [1].
Backpropagation was developed by multiple researchers in the early 60’s and it was implemented to run on computers much as it is today as early as in 1970 by Seppo Linnainmaa [4]. However, Paul Werbos was first in the US to propose that it could be used for neural nets after analyzing it in depth in his 1974 PhD Thesis. Fascinatingly, as with perceptrons, Paul Werbos, was inspired by work related to modeling human mind and thus solved the question of how multilayer neural nets could be trained [5].

Convolutional Neural Networks (CNNs)
Convolutional neural networks (CNNs) are a type of deep neural network designed to process multiple data types, but it was initially designed to analyze images [7]. The name indicates that the network employs a mathematical operation called convolution. This type of neural network is inspired in the visual cortex of the brain. The basic structure of CNNs consist of a convolutional layer, pooling layer, and fully connected layer. This design basically contains an input, an output layer, and multiple hidden layer. At this time, convolutional neural networks are one of the most popular deep learning architectures due to their remarkable capability to analyze spatial data and in a variety of computer vision tasks, including image classification, medical image analysis, natural language processing, and face recognition [1][3].

Depthwise Separable Convolutions
It comes from the idea that spatial dimensions and depth dimensions can be separated which explains the origin of its name. Depthwise separable convolution were proposed to notably accelerate the computations of convolution neural networks [8]. Depthwise separable convolutions require much less computations with a slight performance reduction. Because of its high performance, many different research fields are making use of depthwise separable convolution to implement CNN [9].

Densely Connected Convolutional Networks
The densely connected network can be used to explore new features in face recognition. They can be significantly deeper, more accurate, and really efficient to train when they have short connections between layers close the input and close to the output [10]. A common example of
a densely connected networks is how the neurons of the brain are mostly organized as this network.

3. Approach: Performing Face Detection with Deep Learning
Our goal is to perform face detection with deep learning. To keep things simple, we will make use of pretrained neural networks that we will describe below. We can accomplish that by loading one or more pictures for each subject that is going to be facially recognized, this will be our reference data. Then, it is possible to compare the input image and the reference data. It is possible to approximate similarities, gender, age, and emotions.

3.1 Face Detection Models
The system we built on Node.js will make use of the following three face detection models.

SSD MobileNet v1
One of the main face detectors that is going to be used as a model is a SSD (Single Shot Multibox Detector), which is consists of a CNN based on MobileNet V1; however, it includes additional box prediction layers that are stacked on the top of the network. The neural network can compute the locations of each face in an image and then it will return the bounding boxes and its probability for each face found. This detector is focused on obtaining high accuracy in detecting face bounding boxes instead of low inference time. SSD MobileNet is a face detection model that has been pretrained on the WIDERFACE dataset which is a publicly available.

Tiny Face Detector
The face recognition API implements an optimized face detector called Tiny Face Detector, this detector uses depthwise separable convolutions instead of regular convolutions, which makes it much faster than the other mentioned detectors. However, it is slightly less accurate face detector compared to the SSD MobileNet V1.

The Tiny Face Detector has a really good performance of detection, making it much faster, smaller and less resource consuming compared to the SSD MobileNet V1 face detector model. This model is highly optimized for mobile and web interfaces.

The model is trained on a customized dataset of ~14K labeled using bounding boxes. Moreover, the face model has been trained to predict bounding boxes with complete cover facial feature points, therefore it produces better results on face landmark detections than SSD MobileNet V1.

MTCNN (Multi-task Cascaded Convolutional Neural Networks)
Lastly, there is also another face detector implemented called MTCNN (Multi-task Cascaded Convolutional Neural Network) which is used for experimental purposes. This face model can detect a wide range of face bounding box sizes. However, it consumes a lot of resources and it is not completely optimized. MTCNN is a 3-stage cascaded CNN, which simultaneously returns 5 face landmark points with bounding boxes and scores for each face.
The MTCNN will return the bounding boxes for each face and it will also include a probability score from (0.00 to 1.0). The scores can be used to filter the bounding boxes as a tolerance value [3].

4. System Overview
The system consists of a frontend and a backend application. On this chapter, we will explain the architecture behind the platform and the libraries used to implement the face detection and recognition system.

4.1 Architecture
We will be using the MVC framework, therefore the software’s source code will be separated into three layers as shown below.

![Model-View-Controller Framework](image)

*Figure 4. Model-View-Controller Framework*

The implementation has a frontend and a backend application as shown in the next figure.

![User interacting with website (frontend) and connecting with the server (backend)](image)

*Figure 5. User interacting with website (frontend) and connecting with the server (backend)*
4.2 What is node.js?
For this research, a RESTful platform called Node.JS, Node.js is open-source, cross-platform, and execute JavaScript code outside of a browser. It is being used to perform robust face recognition and face detection using libraries to produce highly accurate results.

For face detection, we can use a deep neural net for face detection or a simple frontal face recognizer to do efficient and less robust detections.

The face recognition module uses the deep neural net to compute unique face descriptions. This face recognition module can be trained with labeled face images and it can later predict the label of any image from user input. Using deep learning algorithms and training data, we will build models and make predictions based on the best fit model. Using computing vision, we humans teach machines to recognize and detect things as we do.

![Figure 6. Phases used for face detection, face extraction, and face recognition](image)

4.3 What is tensorflow.js?
Tensorflow.js is an open-source hardware-accelerated JavaScript library to define, train and run machine learning models in the browsers, using JavaScript and high-level layers API. The name derives from the operation that deep neural networks perform on
4.4 What is face-api.js?
Face-api.js is a JavaScript module, built on top of tensorflow.js core, and it implements several Convolutional Neural Networks (CNNs) to perform face detections and face recognition, and it has been optimized to work on web and mobile devices.

5. Implementation using Node.js and Express
We will explain and provide some examples that can be used to implement the frontend and backend application. More examples and full API documentation can be found at https://justadudewhohacks.github.io/face-api.js/docs/globals.html

5.1 Setting up the environment for our frontend and backend application
1. We need to install the last version of Node.js (Latest LTS Version: 12.13.0 includes npm 6.12.0) from https://nodejs.org/en/download/
2. 
3. We will create a folder that will contain our project 
   e.g. md face_recognition (Windows) or sudo mkdir /home/user/face_recognition (Linux)
4. We will install the latest version of Expresss and face-api.js scripts via npm package using:
   ```
   npm i express
   npm i face-api.js
   ```
5. We can install @tensorflow/tfjs-node (not required) but it can speed things up significantly by compiling the native TensorFlow C++ Library. We can do this using:
   ```
   npm i face-api.js canvas @tensorflow/tfjs-node
   ```
6. It is possible now to monkey patch the environment to use polyfills.

```javascript
// import nodejs bindings to native tensorflow,
// not required, but will speed up things significantly (python required)
import '@tensorflow/tfjs-node';

// implements nodejs wrappers for HTMCanvasElement, HTMLImageElement, ImageData
import * as canv as canv from 'canvas';
import * as faceapi from 'face-api.js';

// patch nodejs environment, we need to provide an implementation of
// HTMLCanvasElement and HTMLImageElement, additionally an implementation
// of ImageData is required, in case you want to use the MTCNN
const { Canvas, Image, ImageData } = canv
faceapi.env.monkeyPatch({ Canvas, Image, ImageData });
```

5.2 Loading detection and recognition models
The global neural network instances are exported from faceapi.nets:

```javascript
console.log(faceapi.nets)
// ageGenderNet
// faceExpressionNet
// faceLandmark68Net
// faceLandmark68TinyNet
// faceRecognitionNet
// ssdMobilenetv1
```
In order to load models, it is important to copy the manifest.json and model weight files (shards) to the same public folder or route.

```javascript
// accordingly for the other models:
// await faceapi.nets.faceLandmark68Net.loadFromUri('/models')
// await faceapi.nets.faceRecognitionNet.loadFromUri('/models')
// ...
```

Another way is loading the models directly from disk

```javascript
await faceapi.nets.ssdMobilenetv1.loadFromDisk('./models')
```

We can also create an instance of the neural networks

```javascript
const net = new faceapi.SsdMobilenetv1()
await net.loadFromUri('/models')
```

### 5.3 Inputs

We can input an HTML element like an img, canvas or video using the id of the element

```html
<img id="myImg" src="images/example.png" />
<video id="myVideo" src="media/example.mp4" />
<canvas id="myCanvas" />
```

```javascript
const input = document.getElementById('myImg')
// const input = document.getElementById('myVideo')
// const input = document.getElementById('myCanvas')
// or simply:
// const input = 'myImg'
```

### 5.4 Face detection

Detecting all the faces of an image. Returns an array

```javascript
const detections = await faceapi.detectAllFaces(input)
```

Detecting the face with the highest confidence score on an image

```javascript
const detection = await faceapi.detectSingleFace(input)
```

By default, faceapi.detectAllFaces and faceapi.detectSingleFace will use the face model SSD MobileNet V1. However, we can specify the detector passing the specific parameters.

```javascript
const detections1 = await faceapi.detectAllFaces(input, new faceapi.SsdMobilenetv1Options())
const detections2 = await faceapi.detectAllFaces(input, new faceapi.TinyFaceDetectorOptions())
const detections3 = await faceapi.detectAllFaces(input, new faceapi.MtcnnOptions())
const detections3 = await faceapi.detectAllFaces(input, new faceapi.MtcnnOptions())
```
Detecting all the faces of an image and face landmark points. Returns an array

```javascript
const detectionsWithLandmarks = await faceapi.detectAllFaces(input).withFaceLandmarks()
```

Detecting the face with the highest confidence score on an image and face landmark points

```javascript
const detectionWithLandmarks = await faceapi.detectSingleFace(input).withFaceLandmarks()
```

It is also possible to use the tiny model instead of the default model

```javascript
const useTinyModel = true
const detectionsWithLandmarks = await faceapi.detectAllFaces(input).withFaceLandmarks(useTinyModel)
```

### 5.6 Computing face descriptors

Detecting all the faces of an image and face landmarks for each face. Returns an array

```javascript
const results = await faceapi.detectAllFaces(input).withFaceLandmarks().withFaceDescriptors()
```

Detecting the face with the highest confidence score on an image, face landmark points and face descriptor

```javascript
const result = await faceapi.detectSingleFace(input).withFaceLandmarks().withFaceDescriptor()
```

### 5.7 Predicting age and gender recognition

Detecting all the faces of an image, face landmarks, and approximate age and recognize gender. Returns an array

```javascript
const detectionsWithExpressions = await faceapi.detectAllFaces(input).withFaceLandmarks().withFaceExpressions()
```

Detecting the face with the highest confidence score on an image, face landmark points and approximate age and recognize gender.

```javascript
const detectionWithExpressions = await faceapi.detectSingleFace(input).withFaceLandmarks().withFaceExpressions()
```

**Note:** We can omit `.withFaceLandmarks()` and it will not compute face landmarks, but the accuracy will be slightly reduced.

### 5.8 Combining multiple tasks

```javascript
// all faces
await faceapi.detectAllFaces(input)
await faceapi.detectAllFaces(input).withFaceExpressions()
await faceapi.detectAllFaces(input).withFaceLandmarks()
await faceapi.detectAllFaces(input).withFaceLandmarks().withFaceExpressions()
await faceapi.detectAllFaces(input).withFaceLandmarks().withFaceExpressions().withFaceDescriptors()
await faceapi.detectAllFaces(input).withFaceLandmarks().withFaceExpressions().withAgeAndGender().withFaceDescriptors()
await faceapi.detectAllFaces(input).withFaceLandmarks().withFaceExpressions().withAgeAndGender().withFaceDescriptor()
await faceapi.detectAllFaces(input).withFaceLandmarks().withFaceExpressions().withAgeAndGender().withFaceDescriptor()

// single face
await faceapi.detectSingleFace(input)
await faceapi.detectSingleFace(input).withFaceExpressions()
await faceapi.detectSingleFace(input).withFaceLandmarks()
await faceapi.detectSingleFace(input).withFaceLandmarks().withFaceExpressions()
await faceapi.detectSingleFace(input).withFaceLandmarks().withFaceExpressions().withFaceDescriptor()
await faceapi.detectSingleFace(input).withFaceLandmarks().withAgeAndGender().withFaceDescriptor()
await faceapi.detectSingleFace(input).withFaceLandmarks().withAgeAndGender().withFaceDescriptor()```
5.9 Performing face recognition by matching descriptors
It is possible to use the faceapi.FaceMatcher to compare reference face descriptors to query face descriptors.

We declare the FaceMatcher using the reference data. For instance, we can use a `referenceImage` and match the descriptors of the faces that were detected to faces of successive images.

```javascript
const results = await faceapi
  .detectAllFaces(referenceImage)
  .withFaceLandmarks()
  .withFaceDescriptors()

if (!results.length) {
  return
}

// create FaceMatcher with automatically assigned labels
// from the detection results for the reference image
const faceMatcher = new faceapi.FaceMatcher(results)

Now we can just use this data to recognize a person’s face in `queryImage1`

```javascript
const singleResult = await faceapi
  .detectSingleFace(queryImage1)
  .withFaceLandmarks()
  .withFaceDescriptor()

if (singleResult) {
  const bestMatch = faceMatcher.findBestMatch(singleResult.descriptor)
  console.log(bestMatch.toString())
}
```

Or we can use this data to recognize multiple faces in `queryImage2`

```javascript
const results = await faceapi
  .detectAllFaces(queryImage2)
  .withFaceLandmarks()
  .withFaceDescriptors()

results.forEach(fd => {
  const bestMatch = faceMatcher.findBestMatch(fd.descriptor)
  console.log(bestMatch.toString())
})
```

Most importantly, we can create labels for the reference descriptors.

```javascript
const labeledDescriptors = [
  new faceapi.LabeledFaceDescriptors(
    'obama',
    [descriptorObama1, descriptorObama2]
  ),
  new faceapi.LabeledFaceDescriptors(
    'trump',
    [descriptorTrump]
  )
]

const faceMatcher = new faceapi.FaceMatcher(labeledDescriptors)
```
5.10 Displaying the results from detections

Setting the overlay canvas

```javascript
const displaySize = { width: input.width, height: input.height }
// resize the overlay canvas to the input dimensions
const canvas = document.getElementById('overlay')
faceapi.matchDimensions(canvas, displaySize)
```

Fortunately, face-api.js is predefined with some high-level drawing functions which we can use.

```javascript
/* Display detected face bounding boxes */
const detections = await faceapi.detectAllFaces(input)
// resize the detected boxes in case your displayed image has a different size than the original
const resizedDetections = faceapi.resizeResults(detections, displaySize)
// draw detections into the canvas
faceapi.draw.drawDetections(canvas, resizedDetections)

/* Display face landmarks */
const detectionsWithLandmarks = await faceapi.detectAllFaces(input)
  .withFaceLandmarks()
// resize the detected boxes and landmarks in case your displayed image has a different size than the original
const resizedResults = faceapi.resizeResults(detectionsWithLandmarks, displaySize)
// draw detections into the canvas
faceapi.draw.drawDetections(canvas, resizedResults)
// draw the landmarks into the canvas
faceapi.draw.drawFaceLandmarks(canvas, resizedResults)

/* Display face expression results */
const detectionsWithExpressions = await faceapi.detectAllFaces(input)
  .withFaceLandmarks()
  .withFaceExpressions()
// resize the detected boxes and landmarks in case your displayed image has a different size than the original
const resizedResults = faceapi.resizeResults(detectionsWithExpressions, displaySize)
// draw detections into the canvas
faceapi.draw.drawDetections(canvas, resizedResults)
// draw a textbox displaying the face expressions with minimum probability into the canvas
const minProbability = 0.05
faceapi.draw.drawFaceExpressions(canvas, resizedResults, minProbability)
```

5.11 Face detection model options

SsdMobilenetv1Options

```javascript
export interface ISsdMobilenetv1Options {
  // minimum confidence threshold
  minConfidence?: number

  // maximum number of faces to return
  maxResults?: number
}
// example
const options = new faceapi.SsdMobilenetv1Options({ minConfidence: 0.8 })
```
TinyFaceDetectorOptions

```javascript
export interface ITinyFaceDetectorOptions {
  // size at which image is processed, the smaller the faster,
  // but less precise in detecting smaller faces, must be divisible
  // by 32, common sizes are 128, 160, 224, 320, 416, 512, 608,
  // for face tracking via webcam I would recommend using smaller sizes,
  // e.g. 128, 160, for detecting smaller faces use larger sizes, e.g. 512, 608
  // default: 416
  inputSize?: number

  // minimum confidence threshold
  // default: 0.5
  scoreThreshold?: number
}
```

// example
const options = new faceapi.TinyFaceDetectorOptions({ inputSize: 320 })

MtcnnOptions

```javascript
export interface IMtcnnOptions {
  // minimum face size to expect, the higher the faster processing will be,
  // but smaller faces won’t be detected
  // default: 20
  minFaceSize?: number

  // the score threshold values used to filter the bounding
  // boxes of stage 1, 2 and 3
  // default: [0.6, 0.7, 0.7]
  scoreThresholds?: number[]

  // scale factor used to calculate the scale steps of the image
  // pyramid used in stage 1
  // default: 0.709
  scaleFactor?: number

  // number of scaled versions of the input image passed through the CNN
  // of the first stage, lower numbers will result in lower inference time,
  // but will also be less accurate
  // default: 10
  maxNumScales?: number

  // instead of specifying scaleFactor and maxNumScales you can also
  // set the scaleSteps manually
  scaleSteps?: number[]
}
```

// example
const options = new faceapi.MtcnnOptions({ minFaceSize: 100, scaleFactor: 0.8 })

It is possible to extract images just focused at the face tor each bounding box. This procedure needs to be done before passing this data to the face recognition network, as this will improve face recognition and the results will be much more accurate.

The face recognition API consists of CNN which will return 68 point face landmarks of any given face image. Using the coordinates of the landmark positions, the bonding box can be centered on each face.
5.11 68 Point Face Landmark Detection Models

The face API includes a very lightweight, fast and accurate 68 point landmark detector. The models use the idea of depthwise separable convolutions and densely connected blocks. The models are trained on a dataset of ~35k face images.

We can retrieve the face landmark point and contours and get their positions individually as shown below:

```cpp
// to get the positions of face landmark points
const landmarkPositions = landmarks.positions

// or to get the positions of individual contours, // only available for 68 point face landmarks (FaceLandmarks68)
const jawOutline = landmarks.getJawOutline()
const nose = landmarks.getNose()
const mouth = landmarks.getMouth()
const leftEye = landmarks.getLeftEye()
const rightEye = landmarks.getRightEye()
const leftEyeBrow = landmarks.getLeftEyeBrow()
const rightEyeBrow = landmarks.getRightEyeBrow()
```
5.12 Using Face Recognition Models
Now, it is possible to pass the extracted data and aligned face images to the face recognition network based on an architecture like ResNet-34 which implemented in dlib. The neural network has been trained to learn to map the different facial features on humans and translated it to a face descriptor (a vector with 128 values) which is sometimes referred to as face embeddings. The model is not limited to the dataset used for training, thus it is possible to use it to perform face recognition of any person.

In order to compare two faces, the face descriptor of each extracted image is used and then it is compared using the face descriptors of the reference data. More specifically, the Euclidean distance is computed between the two face descriptors and the algorithms will define whether two faces are similar based on a threshold value (which depends on the image resolution). For instance, a threshold value of 0.6 is ideal for a picture with resolution of 150 x 150 pixels.

5.13 Euclidean distance
This is the distance used to measure the similarity of pattern samples in the geometric pattern space. The Euclidean metric is widely used mainly because it is simple to calculate. With this distance, Euclidean space becomes a metric space [6].

Euclidean distance for a n-dimensional space is defined as:

\[ d(p, q) = \sqrt{\sum_{i=0}^{n} (p_i - q_i)^2} = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} \]

Euclidean distance for a two-dimensional space is defined as:

\[ d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \]

Figure 9. Straight-line distance between two points in Euclidean space
Euclidean distance did not consider the relationship of each dimension of the vector, and each dimension is equally important, which affect the effectiveness and scope of its use. The weighted Euclidean distance is used to ensure the accuracy of the face identification.

The following function is used to compute the Euclidean distance between two face descriptors

```javascript
// Used for computing the euclidean distance between two face descriptors
const dist = faceapi.euclideanDistance([0, 0], [0, 20])
console.log(dist) // 20
```

The following pictures illustrates the comparison of two face images using Euclidean distance to find similarity:

![Figure 10. Satisfactory results of face similarity using Euclidean distance](penny2.png) ![Figure 11. Unsatisfactory results of face similarity using Euclidean distance](penny3.png)

Distance: 0.52

Distance: 0.4

Distance: 0.81

Distance: 0.67
Figure 12. Daniel Radcliffe (Age 28 and age 11)

Distance:

0.45 (match)

Figure 13. Leonardo DiCaprio (Clean and not clean face)

Distance:

0.49 (match)

Figure 14. Cristhian Espinosa (Age 24 and Age 4)
### 5.14 Frontend application (User Interface)

**Figure 15.** Example of our frontend application running on localhost:3000. We have a menu on the left and the views are loaded on the right.
6. Experimental Results and Analysis

6.1 Multiple Face Detection (using Face-api.js)

Figure 16. Face detection and Face Bounding Boxes. Example #1

Figure 17. Face detection and Face Landmark Detection. Example #1
Figure 18. Face detection and Face Bounding Boxes. Example #2

Figure 19. Face detection and Face Bounding Boxes. Example #2
6.2 Face expression Recognition model (using Face-api.js)

The face expression recognition model is lightweight, fast, and provides realistic accuracy. The model uses depthwise separable convolutions and densely connected blocks. The model has been trained on a variety of images from public datasets and the web.

Figure 20. Face expression recognition using SSD MobileNet v1 as Neural Network and a min confidence coefficient set to 0.5. Example #1

Figure 21. Face expression recognition using SSD MobileNet v1 as Neural Network and a min confidence coefficient set to 0.5. Example #2.
Figure 22. Sad face which could possibly be angry to certain level

Figure 23. Angry Hugh Jackman

Figure 24. Disgusted face (70%) and Angry (0.26)
6.3 Age and Gender Recognition model (using Face-api.js)

The age and gender recognition model uses a multitask network, which implements a feature extraction layer, age regression layer, and gender classifier. This architecture is similar to Xception.

Figure 25. Detecting age and gender on six different faces using the face-api.js

Figure 26. Detecting age and gender on 5 different faces using the face-api.js
6.4 Face Extraction (using Face-api.js)

Figure 27. Performing face extraction using face-api.js
6.5 Comparing detection speed on video face tracking (using Face-api.js)

Figure 28. Face detection performed using Tiny Face Detector as neural network.

Figure 29. Face detection performed using SSD MobileNet v1 as neural network.
6.6 Video conclusions

<table>
<thead>
<tr>
<th>Neural Network</th>
<th>Latency</th>
<th>Estimated FPS (Frames per second)</th>
<th>Faces Detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tiny Face Detector</td>
<td>26 ms</td>
<td>37.92</td>
<td>2</td>
</tr>
<tr>
<td>SSD MobileNet v1</td>
<td>34 ms</td>
<td>29.26</td>
<td>2</td>
</tr>
<tr>
<td>MTCNN</td>
<td>977 ms</td>
<td>1.02</td>
<td>6</td>
</tr>
</tbody>
</table>

Figure 30. Face detection performed using MTCNN as neural network.
7. Additional improvements

- Implement object recognition to identify objects in images or videos.
- Add an independent feature to detect simple hand gestures from a live video. This example is not intended to solve hand gesture recognition perfectly but rather to give a hint to how to approach this task.

8. Conclusion

In this paper, we studied the use of deep neural networks on face detection and recognition. We also analyzed some of the different neural networks including convolutional neural networks, depthwise separable convolutions, and densely connected convolutional networks. We have implemented a simple face detection and recognition system using Node.js and TensorFlow.js core API making use of existing API like the face-api.js package located at https://github.com/justadudewhohacks/face-api.js/. This library is optimized to be used in the browser. Generating the data is out of the scope of this research, but there are publicly available datasets that can be used to load in the app for doing face recognition. The API allowed us to run different experiments where users can detect the emotion of a person, the gender, and the age of a person. In the last years, facial recognition has been well developed that now we can build apps that can recognize faces with great accuracy. The fascinating part about this experiment is to find more useful applications for this technology.

As we enter the top era of big data, complex systems require more and more data. The diverse training data that we will use has to be representative of the data that the neural network will come across in the real world. Along with the training data, it is difficult to understand the weights that the neural network has learned. Nonetheless, when using multiple layers, it could be really difficult to draw conclusions from the weights related to the decision that a neuron is actually handling.

This paper has covered the core concepts used in deep learning i.e. what mechanisms of backend computation result in enhanced model accuracy for face detection and face recognition. A real thanks to the researchers in this field whose discoveries and findings have contributed to the true power of neural networks.

The results demonstrate that using deep neural networks and the last techniques can improve the prediction accuracy for face detection, face recognition, and face extraction. In particular, insights that arise from experimenting, understanding, and interacting with neural networks led us to the most valuable observations. Enabling other students to freely explore this compelling, but sometimes intimidating, field is perhaps the central focus of this work. Consecutively, we need additional tools to allow to explore deep learning in terms of algorithms, architecture and design techniques.
References


doi: 10.1109/LSP.2016.2603342


doi: 10.1109/ICCIS.2013.63


Appendix
Source code
Age and Gender Recognition

<!DOCTYPE html>
<html>
<head>
<script src="face-api.js"></script>
<script src="js/commons.js"></script>
<script src="js/faceDetectionControls.js"></script>
<script src="js/imageSelectionControls.js"></script>
<link rel="stylesheet" href="styles.css">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.100.2/css/materialize.css">
<script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.100.2/js/materialize.min.js"></script>
</head>
<body>
<div id="navbar"></div>
<div class="center-content page-container">
  <div class="progress" id="loader">
    <div class="indeterminate"></div>
  </div>
  <div style="position: relative" class="margin">
    <img id="inputImg" src="" style="max-width: 800px;" />
    <canvas id="overlay" />
  </div>
  <div class="row side-by-side">
    <!-- image_selection_control -->
    <div id="selectList"></div>
    <div class="row">
      <label for="imgUrlInput">Get image from URL:</label>
      <input id="imgUrlInput" type="text" class="bold">
    </div>
    <button class="waves-effect waves-light btn"
      onclick="loadImageFromUrl()"
      >Ok</button>
    <!-- image_selection_control -->
  </div>
  <div class="row side-by-side">
    <!-- face_detector_selection_control -->
    <div id="face_detector_selection_control" class="row input-field"
      style="margin-right: 20px;">
      <select id="selectFaceDetector">
        <option value="ssd_mobilenetv1">SSD Mobilenet V1</option>
      </select>
    </div>
  </div>
</div>
</body>
</html>
<select value="tiny_face_detector">Tiny Face Detector</select> <select value="mtcnn">MTCNN</select>
<label>Select Face Detector</label>
</div>

<!-- face_detector_selection_control -->

<!-- ssd_mobilenetv1_controls -->
<span id="ssd_mobilenetv1_controls">
<div class="row side-by-side">
  <div class="row input-field" style="margin-right: 20px;">
    <select id="inputSize">
      <option value="" disabled selected>Input Size:</option>
      <option value="160">160 x 160</option>
      <option value="224">224 x 224</option>
      <option value="320">320 x 320</option>
      <option value="416">416 x 416</option>
      <option value="512">512 x 512</option>
      <option value="608">608 x 608</option>
    </select>
    <label>Input Size</label>
  </div>
  <div class="row">
    <label for="scoreThreshold">Score Threshold:</label>
    <input disabled value="0.5" id="scoreThreshold" type="text" class="bold"/>
    <button class="waves-effect waves-light btn" onclick="onDecreaseScoreThreshold()"></button>
    <button class="waves-effect waves-light btn" onclick="onIncreaseScoreThreshold()"></button>
  </div>
</div>
</span>
<!-- tiny_face_detector_controls -->

<!-- ssd_mobilenetv1_controls -->
<!-- face_detector_selection_control -->
async function updateResults() {
    if (!isFaceDetectionModelLoaded()) {
        return
    }
    const inputImgEl = $('#inputImg').get(0)
    const options = getFaceDetectorOptions()

    const results = await faceapi.detectAllFaces(inputImgEl, options)
    // compute face landmarks to align faces for better accuracy
    .withFaceLandmarks()
    .withAgeAndGender()

    const canvas = $('#overlay').get(0)
    faceapi.matchDimensions(canvas, inputImgEl)

    const resizedResults = faceapi.resizeResults(results, inputImgEl)
```javascript
faceapi.draw.drawDetections(canvas, resizedResults)

resizedResults.forEach(result => {
    const { age, gender, genderProbability } = result
    new faceapi.draw.DrawTextField(
        [`
        ${faceapi.round(age, 0)} years`,
        `${gender} (${faceapi.round(genderProbability)})`
    ],
    result.detection.box.bottomLeft
).draw(canvas)
})

async function run() {
    // load face detection and age and gender recognition models
    // and load face landmark model for face alignment
    await changeFaceDetector(SSD_MOBILENETV1)
    await faceapi.loadFaceLandmarkModel('/')
    await faceapi.nets.ageGenderNet.load('/')

    // start processing image
    updateResults()
}

$(document).ready(function() {
    renderNavBar('#navbar', 'age_and_gender_recognition')
    initImageSelectionControls('happy.jpg', true)
    initFaceDetectionControls()
    run()
})
</script>
</body>
</html>
Face detection

```html
<!DOCTYPE html>
<html>
<head>
  <script src="face-api.js"></script>
  <script src="js/commons.js"></script>
  <script src="js/faceDetectionControls.js"></script>
  <script src="js/imageSelectionControls.js"></script>
  <link rel="stylesheet" href="styles.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.100.2/css/materialize.css">
  <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.100.2/js/materialize.min.js"></script>
</head>
<body>
<div id="navbar"></div>
<div class="center-content page-container">
  <div class="progress" id="loader">
    <div class="indeterminate"></div>
  </div>
  <div style="position: relative" class="margin">
    <img id="inputImg" src="" style="max-width: 800px;" />
    <canvas id="overlay" />
  </div>
  <div class="row side-by-side">
    <!-- image_selection_control -->
    <div id="selectList"></div>
    <div class="row">
      <label for="imgUrlInput">Get image from URL:</label>
      <input id="imgUrlInput" type="text" class="bold">
    </div>
    <button class="waves-effect waves-light btn" onclick="loadImageFromUrl()">
      Ok
    </button>
    <!-- image_selection_control -->
  </div>
  <!-- Image selection -->
  <div class="row side-by-side">
    <!-- image_selection_control -->
    <div id="selectList"></div>
    <div class="row">
      <label for="imgUrlInput">Get image from File:</label>
      <input id="myFileUpload" type="file" onchange="uploadImage()"
             accept=".jpg, .jpeg, .png">
    </div>
    <button class="waves-effect waves-light btn">
      Ok
    </button>
    <!-- image_selection_control -->
  </div>
</div>
</body>
</html>
```
onclick="loadImageFromUrl()"
>
Ok
</button>
</div>

<div class="row side-by-side">

<!-- image_selection_control -->
</div>

<!-- face_detector_selection_control -->
<div id="face_detector_selection_control" class="row input-field"
style="margin-right: 20px;">
<select id="selectFaceDetector"/>
<option value="ssd_mobilenetv1">SSD Mobilenet V1</option>
<option value="tiny_face_detector">Tiny Face Detector</option>
<option value="mtcnn">MTCNN</option>
</select>
<label>Select Face Detector</label>
</div>

<!-- check boxes -->
<div class="row">
<input type="checkbox" id="withFaceLandmarksCheckbox"
onchange="onChangeWithFaceLandmarks(event)" />
<label for="withFaceLandmarksCheckbox">Detect Face Landmarks</label>
<input type="checkbox" id="hideBoundingBoxesCheckbox"
onchange="onChangeHideBoundingBoxes(event)" />
<label for="hideBoundingBoxesCheckbox">Hide Bounding Boxes</label>
</div>

<!-- ssd_mobilenetv1_controls -->
<span id="ssd_mobilenetv1_controls">
<div class="row side-by-side">

<label for="minConfidence">Min Confidence:</label>
<input disabled value="0.5" id="minConfidence" type="text"
class="bold" />
</div>
<button class="waves-effect waves-light btn"
onclick="onDecreaseMinConfidence()"
>
<i class="material-icons left">-</i>
</button>
<button class="waves-effect waves-light btn"
onclick="onIncreaseMinConfidence()"
>
<i class="material-icons left">+</i>
</button>
</span>
</div>
<!-- tiny_face_detector_controls -->
<span id="tiny_face_detector_controls">
<div class="row side-by-side">
  <div class="row input-field" style="margin-right: 20px;">
    <select id="inputSize">
      <option value="" disabled selected>Input Size:</option>
      <option value="160">160 x 160</option>
      <option value="224">224 x 224</option>
      <option value="320">320 x 320</option>
      <option value="416">416 x 416</option>
      <option value="512">512 x 512</option>
      <option value="608">608 x 608</option>
    </select>
    <label>Input Size</label>
  </div>
  <div class="row">
    <label for="scoreThreshold">Score Threshold:</label>
    <input disabled value="0.5" id="scoreThreshold" type="text" class="bold"/>
  </div>
  <button class="waves-effect waves-light btn" onclick="onDecreaseScoreThreshold()" >
    <i class="material-icons left">-</i>
  </button>
  <button class="waves-effect waves-light btn" onclick="onIncreaseScoreThreshold()" >
    <i class="material-icons left">+</i>
  </button>
</div>
</span>

<!-- tiny_face_detector_controls -->

<!-- mtcnn_controls -->
<span id="mtcnn_controls">
<div class="row side-by-side">
  <div class="row">
    <label for="minFaceSize">Minimum Face Size:</label>
    <input disabled value="20" id="minFaceSize" type="text" class="bold"/>
  </div>
  <button class="waves-effect waves-light btn" onclick="onDecreaseMinFaceSize()" >
    <i class="material-icons left">-</i>
  </button>
  <button class="waves-effect waves-light btn" onclick="onIncreaseMinFaceSize()" >
    <i class="material-icons left">+</i>
  </button>
</div>
</span>
let withFaceLandmarks = false
let withBoxes = true

function onChangeWithFaceLandmarks(e) {
  withFaceLandmarks = $(e.target).prop('checked')
  updateResults()
}

function onChangeHideBoundingBoxes(e) {
  withBoxes = !$({e.target).prop('checked')
  updateResults()
}

async function updateResults() {
  if (!isFaceDetectionModelLoaded()) {
    return
  }

  const drawBoxes = withBoxes
  const drawLandmarks = withFaceLandmarks

  const inputImgEl = $('#inputImg').get(0)
  const options = getFaceDetectorOptions()

  let task = faceapi.detectAllFaces(inputImgEl, options)
  task = withFaceLandmarks ? task.withFaceLandmarks() : task
  const results = await task

  const canvas = $('#overlay').get(0)
  faceapi.matchDimensions(canvas, inputImgEl)
  const resizedResults = faceapi.resizeResults(results, inputImgEl)

  if (withBoxes) {
    faceapi.draw.drawDetections(canvas, resizedResults)
  }

  if (withFaceLandmarks) {
    faceapi.draw.drawFaceLandmarks(canvas, resizedResults)
  }
}

async function uploadImage() {
  const imgFile = document.getElementById('myFileUpload').files[0]
  // create an HTMLImageElement from a Blob
  const img = await faceapi.bufferToImage(imgFile)
  document.getElementById('inputImg').src = img.src
  updateResults()
}

async function run() {
// load face detection
await changeFaceDetector(SSD_MOBILENETV1)
await faceapi.loadFaceLandmarkModel('/')
// start processing image
updateResults()
}

$(document).ready(function()
    renderNavBar('#navbar', 'face_detection')
    initImageSelectionControls()
    initFaceDetectionControls()
    run()
})
</script>
</body>
</html>
Face Expression Recognition

```html
<!DOCTYPE html>
<html>
<head>
  <script src="face-api.js"></script>
  <script src="js/commons.js"></script>
  <script src="js/faceDetectionControls.js"></script>
  <script src="js/imageSelectionControls.js"></script>
  <link rel="stylesheet" href="styles.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.100.2/css/materialize.css">
  <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.100.2/js/materialize.min.js"></script>
</head>
<body>
<div id="navbar"></div>
<div class="center-content page-container">
  <div style="position: relative" class="margin">
    <img id="inputImg" src="" style="max-width: 800px;" />
    <canvas id="overlay" />
  </div>
</div>
</body>
</html>
```
<select>
    <option value="tiny_face_detector">Tiny Face Detector</option>
    <option value="mtcnn">MTCNN</option>
</select>

<label>Select Face Detector</label>

<div>
    <div class="row side-by-side">-
        <div class="row">
            <div class="row input-field" style="margin-right: 20px;">-
                <select id="inputSize">
                    <option value="" disabled selected>Input Size:</option>
                    <option value="160">160 x 160</option>
                    <option value="224">224 x 224</option>
                    <option value="320">320 x 320</option>
                    <option value="416">416 x 416</option>
                    <option value="512">512 x 512</option>
                    <option value="608">608 x 608</option>
                </select>
                <label>Input Size</label>
            </div>
            <div class="row">
                <div class="row">
                    <input disabled value="0.5" id="minConfidence" type="text" class="bold"/>
                    <button class="waves-effect waves-light btn" onclick="onDecreaseMinConfidence()"
                            >-
                        <i class="material-icons left">-</i>
                    </button>
                    <button class="waves-effect waves-light btn" onclick="onIncreaseMinConfidence()"
                            >-
                        <i class="material-icons left">+</i>
                    </button>
                </div>
            </div>
        </div>
    </div>
</div>

<!-- ssd_mobilenetv1_controls -->
<span id="ssd_mobilenetv1_controls">
    <div class="row side-by-side">
        <div class="row">
            <input disabled value="0.5" id="minConfidence" type="text" class="bold"/>
            <button class="waves-effect waves-light btn" onclick="onDecreaseMinConfidence()"
                    >-
                <i class="material-icons left">-</i>
            </button>
            <button class="waves-effect waves-light btn" onclick="onIncreaseMinConfidence()"
                    >-
                <i class="material-icons left">+</i>
            </button>
        </div>
    </div>
</span>

<!-- tiny_face_detector_controls -->
<span id="tiny_face_detector_controls">
    <div class="row side-by-side">
        <div class="row">
            <input disabled value="0.5" id="scoreThreshold" type="text" class="bold"/>
            <button class="waves-effect waves-light btn" onclick="onDecreaseScoreThreshold()"
                    >-
        </div>
    </div>
</span>
<i class="material-icons left">-</i>
</button>
<button class="waves-effect waves-light btn" onclick="onIncreaseScoreThreshold()"
><i class="material-icons left">+</i>
</button>
</div>
</span>
<!-- tiny_face_detector_controls -->
<!-- mtcnn_controls -->
<span id="mtcnn_controls"
><div class="row side-by-side"
><div class="row"
><label for="minFaceSize">Minimum Face Size:</label>
<input disabled value="20" id="minFaceSize" type="text" class="bold">
</div>
<button class="waves-effect waves-light btn" onclick="onDecreaseMinFaceSize()"
><i class="material-icons left">-</i>
</button>
<button class="waves-effect waves-light btn" onclick="onIncreaseMinFaceSize()"
><i class="material-icons left">+</i>
</button>
</div>
</span>
<!-- mtcnn_controls -->
</span>
</div>
</span>
</body>
<script>
let withFaceLandmarks = false
let withBoxes = true

function onChangeWithFaceLandmarks(e) {
  withFaceLandmarks = $(e.target).prop('checked')
  updateResults()
}

function onChangeHideBoundingBoxes(e) {
  withBoxes = !$($(e.target).prop('checked'))
  updateResults()
}

let thresh = 0.1

async function updateResults() {
  if (!isFaceDetectionModelLoaded()) {
    return
  }

  const inputImgEl = $('#inputImg').get(0)
  const options = getFaceDetectorOptions()
  let task = faceapi.detectAllFaces(inputImgEl, options)

  withFaceLandmarks = task
  withBoxes = task
}

</script>
```javascript
const results = await task
// compute face landmarks to align faces for better accuracy
   .withFaceLandmarks()
   .withFaceExpressions()

const canvas = $('#overlay').get(0)
faceapi.matchDimensions(canvas, inputImgEl)

const resizedResults = faceapi.resizeResults(results, inputImgEl)
const minConfidence = 0.05

faceapi.draw.drawDetections(canvas, resizedResults)
faceapi.draw.drawFaceExpressions(canvas, resizedResults, minConfidence)

if (withBoxes) {
    faceapi.draw.drawDetections(canvas, resizedResults)
}
if (withFaceLandmarks) {
    faceapi.draw.drawFaceLandmarks(canvas, resizedResults)
}

async function uploadImage() {
    const imgFile = document.getElementById('myFileUpload').files[0]
    // create an HTMLImageElement from a Blob
    const img = await faceapi.bufferToImage(imgFile)
document.getElementById('inputImg').src = img.src
    updateResults()
}

async function run() {
    // load face detection and face expression recognition models
    // and load face landmark model for face alignment
    await changeFaceDetector(SSD_MOBILENETV1)
    await faceapi.loadFaceLandmarkModel('/
    await faceapi.loadFaceExpressionModel('/
    // start processing image
    updateResults()
}
```

$(document).ready(function() {  
    renderNavBar('#navbar', 'face_expression_recognition')  
    initImageSelectionControls('happy.jpg', true)  
    initFaceDetectionControls()  
    run()  
})
Face Extraction

```html
<!DOCTYPE html>
<html>
<head>
    <script src="face-api.js"></script>
    <script src="js/commons.js"></script>
    <script src="js/faceDetectionControls.js"></script>
    <script src="js/imageSelectionControls.js"></script>
    <link rel="stylesheet" href="styles.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.100.2/css/materialize.css">
    <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.100.2/js/materialize.min.js"></script>
</head>
<body>
    <div id="navbar"></div>
    <div class="center-content page-container">
        <div id="loader">
            <div class="indeterminate"></div>
        </div>
        <div style="position: relative" class="margin">
            <img id="inputImg" src="" style="max-width: 800px;" />
            <canvas id="overlay" />
        </div>
        <div id="facesContainer"></div>
    </div>
    <div class="row side-by-side">
        <!-- face_detector_selection_control -->
        <div id="face_detector_selection_control" class="row input-field" style="margin-right: 20px;">
            <select id="selectFaceDetector">
                <option value="ssd_mobilenetv1">SSD Mobilenet V1</option>
                <option value="tiny_face_detector">Tiny Face Detector</option>
                <option value="mtcnn">MTCNN</option>
            </select>
            <label>Select Face Detector</label>
        </div>
        <!-- image_selection_control -->
        <div id="selectList"></div>
        <div class="row">
            <label for="imgUrlInput">Get image from URL:</label>
            <input id="imgUrlInput" type="text" class="bold">
        </div>
        <button class="waves-effect waves-light btn" onclick="loadImageFromUrl()">
            Ok
        </button>
    </div>
</body>
</html>
```
<label>Input Size</label>
</div>
<div class="row">
<label for="scoreThreshold">Score Threshold:</label>
<input disabled value="0.5" id="scoreThreshold" type="text" class="bold">
</div>
<button class="waves-effect waves-light btn" onclick="onDecreaseScoreThreshold()">
<i class="material-icons left">-</i>
</button>
<button class="waves-effect waves-light btn" onclick="onIncreaseScoreThreshold()">
<i class="material-icons left">+</i>
</button>

<button class="waves-effect waves-light btn" onclick="onDecreaseMinFaceSize()">
<i class="material-icons left">-</i>
</button>
<button class="waves-effect waves-light btn" onclick="onIncreaseMinFaceSize()">
<i class="material-icons left">+</i>
</button>
</span>
</div>
</script>

async function updateResults() {
  if (!isFaceDetectionModelLoaded()) {
    return
  }

  const inputImgEl = $('inputImg').get(0)
const options = getFaceDetectorOptions()

const detections = await faceapi.detectAllFaces(inputImgEl, options)
const faceImages = await faceapi.extractFaces(inputImgEl, detections)

displayExtractedFaces(faceImages)
}

async function uploadImage() {
  const imgFile = document.getElementById('myFileUpload').files[0]
  // create an HTMLImageElement from a Blob
  const img = await faceapi.bufferToImage(imgFile)
document.getElementById('inputImg').src = img.src
updateResults()
}

function displayExtractedFaces(faceImages) {
  const canvas = $('#overlay').get(0)
  faceapi.matchDimensions(canvas, $('#inputImg').get(0))

  $('#facesContainer').empty()
  faceImages.forEach(canvas => $('#facesContainer').append(canvas))
}

async function run() {
  // load face detection model
  await changeFaceDetector(selectedFaceDetector)

  // start processing image
  updateResults()
}

$(document).ready(function() {
  renderNavBar('#navbar', 'face_extraction')
  initImageSelectionControls()
  initFaceDetectionControls()
  run()
})
</script>
</body>
</html>
Video Face Tracking

```html
<!DOCTYPE html>
<html>
<head>
  <script src="face-api.js"></script>
  <script src="js/commons.js"></script>
  <script src="js/faceDetectionControls.js"></script>
  <link rel="stylesheet" href="styles.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.100.2/css/materialize.css">
  <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.100.2/js/materialize.min.js"></script>
</head>
<body>
  <div id="navbar"></div>
  <div class="center-content page-container">
    <div id="loader" class="progress indeterminate"></div>
    <video src="bbt.mp4" id="inputVideo" autoplay muted loop playsinline></video>
    <canvas id="overlay" /></div>
    <div class="row side-by-side">
      <!-- face_detector_selection_control -->
      <div id="face_detector_selection_control" class="row input-field" style="margin-right: 20px;">
        <select id="selectFaceDetector">
          <option value="ssd_mobilenetv1">SSD Mobilenet V1</option>
          <option value="tiny_face_detector">Tiny Face Detector</option>
          <option value="mtcnn">MTCNN</option>
        </select>
        <label>Select Face Detector</label>
      </div>
      <!-- check boxes -->
      <div class="row" style="width: 220px;">
        <input type="checkbox" id="withFaceLandmarksCheckbox" onchange="onChangeWithFaceLandmarks(event)" />
        <label for="withFaceLandmarksCheckbox">Detect Face Landmarks</label>
        <input type="checkbox" id="hideBoundingBoxesCheckbox" onchange="onChangeHideBoundingBoxes(event)" />
        <label for="hideBoundingBoxesCheckbox">Hide Bounding Boxes</label>
      </div>
      <!-- fps_meter -->
      <div id="fps_meter" class="row side-by-side">
        <label for="time">Time:</label>
        <input disabled value="" id="time" type="text" class="bold">
        <label for="fps">Estimated Fps:</label>
        <input disabled value="" id="fps" type="text" class="bold">
      </div>
    </div>
  </div>
</body>
</html>
```
<div class="row side-by-side">
  <div class="row">
    <label for="minFaceSize">Minimum Face Size:</label>
    <input disabled value="20" id="minFaceSize" type="text" class="bold">
  </div>
  <button class="waves-effect waves-light btn" onclick="onDecreaseMinFaceSize()">
    <i class="material-icons left">-</i>
  </button>
  <button class="waves-effect waves-light btn" onclick="onIncreaseMinFaceSize()">
    <i class="material-icons left">+</i>
  </button>
</div>

<script>
let forwardTimes = []
let withFaceLandmarks = false
let withBoxes = true

function onChangeWithFaceLandmarks(e) {
  withFaceLandmarks = $(e.target).prop('checked')
}

function onChangeHideBoundingBoxes(e) {
  withBoxes = !$(e.target).prop('checked')
}

function updateTimeStats(timeInMs) {
  forwardTimes = [timeInMs].concat(forwardTimes).slice(0, 30)
  const avgTimeInMs = forwardTimes.reduce((total, t) => total + t) / forwardTimes.length
  $('#time').val(`${Math.round(avgTimeInMs)} ms`)
  $('#fps').val(`${faceapi.round(1000 / avgTimeInMs)}`)
}

async function onPlay(videoEl) {
  if(!videoEl.currentTime || videoEl.paused || videoEl.ended || !isFaceDetectionModelLoaded())
    return setTimeout(() => onPlay(videoEl))

  const options = getFaceDetectorOptions()
  const ts = Date.now()
  const drawBoxes = withBoxes
  const drawLandmarks = withFaceLandmarks

  let task = faceapi.detectAllFaces(videoEl, options)
  task = withFaceLandmarks ? task.withFaceLandmarks() : task
  const results = await task

  updateTimeStats(Date.now() - ts)
</script>
const canvas = $('#overlay').get(0)
const dims = faceapi.matchDimensions(canvas, videoEl, true)
const resizedResults = faceapi.resizeResults(results, dims)
if (drawBoxes) {
    faceapi.draw.drawDetections(canvas, resizedResults)
}
if (drawLandmarks) {
    faceapi.draw.drawFaceLandmarks(canvas, resizedResults)
}
setTimeout(() => onPlay(videoEl))
}

async function run() {
    // load face detection and face landmark models
    await changeFaceDetector(TINY_FACE_DETECTOR)
    await faceapi.loadFaceLandmarkModel('/
    changeInputSize(416)
    // start processing frames
    onPlay($('inputVideo').get(0))
}

function updateResults() {}

$(document).ready(function() {
    renderNavBar('#navbar', 'video_face_tracking')
    initFaceDetectionControls()
    run()
});
</script>
</body>
</html>
Webcam Face Detection

```html
<!DOCTYPE html>
<html>
<head>
  <script src="face-api.js"></script>
  <script src="js/commons.js"></script>
  <script src="js/faceDetectionControls.js"></script>
  <link rel="stylesheet" href="styles.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.100.2/css/materialize.css">
  <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.100.2/js/materialize.min.js"></script>
</head>
<body>
  <div id="navbar"></div>
  <div class="center-content page-container">
    <div class="progress" id="loader">
      <div class="indeterminate"></div>
    </div>
    <div style="position: relative" class="margin">
      <video onloadedmetadata="onPlay(this)" id="inputVideo" autoplay muted playsinline>
      </video>
      <canvas id="overlay" />
    </div>
    <div class="row side-by-side">
      <!-- face_detector_selection_control -->
      <div id="face_detector_selection_control" class="row input-field" style="margin-right: 20px;">
        <select id="selectFaceDetector">
          <option value="ssd_mobilenetv1">SSD Mobilenet V1</option>
          <option value="tiny_face_detector">Tiny Face Detector</option>
          <option value="mtcnn">MTCNN</option>
        </select>
        <label>Select Face Detector</label>
      </div>
      <!-- fps_meter -->
      <div id="fps_meter" class="row side-by-side">
        <div>
          <label for="time">Time:</label>
          <input disabled value="-" id="time" type="text" class="bold">
          <label for="fps">Estimated Fps:</label>
          <input disabled value="-" id="fps" type="text" class="bold">
        </div>
      </div>
    </div>
  </div>
</body>
</html>
```
<!-- ssd_mobilenetv1_controls -->
<span id="ssd_mobilenetv1_controls">
  <div class="row side-by-side">
    <div class="row">
      <label for="minConfidence">Min Confidence:</label>
      <input disabled value="0.5" id="minConfidence" type="text" class="bold">
    </div>
    <button class="waves-effect waves-light btn" onclick="onDecreaseMinConfidence()">
      <i class="material-icons left">-</i>
    </button>
    <button class="waves-effect waves-light btn" onclick="onIncreaseMinConfidence()">
      <i class="material-icons left">+</i>
    </button>
  </div>
</span>

<!-- tiny_face_detector_controls -->
<span id="tiny_face_detector_controls">
  <div class="row side-by-side">
    <div class="row input-field" style="margin-right: 20px;">
      <select id="inputSize">
        <option value="" disabled selected>Input Size:</option>
        <option value="128">128 x 128</option>
        <option value="160">160 x 160</option>
        <option value="224">224 x 224</option>
        <option value="320">320 x 320</option>
        <option value="416">416 x 416</option>
        <option value="512">512 x 512</option>
        <option value="608">608 x 608</option>
      </select>
      <label>Input Size</label>
    </div>
    <div class="row">
      <label for="scoreThreshold">Score Threshold:</label>
      <input disabled value="0.5" id="scoreThreshold" type="text" class="bold">
      <button class="waves-effect waves-light btn" onclick="onDecreaseScoreThreshold()">
        <i class="material-icons left">-</i>
      </button>
      <button class="waves-effect waves-light btn" onclick="onIncreaseScoreThreshold()">
        <i class="material-icons left">+</i>
      </button>
    </div>
  </div>
</span>
<i class="material-icons left">+</i></button>
</div>
</span>  
</div>
</span>  
</body>
<script>
let forwardTimes = []

function updateTimeStats(timeInMs) {
    forwardTimes = [timeInMs].concat(forwardTimes).slice(0, 30)
    const avgTimeInMs = forwardTimes.reduce((total, t) => total + t) / forwardTimes.length
    $('#time').val(`Math.round(avgTimeInMs) ms`)
    $('#fps').val(`faceapi.round(1000 / avgTimeInMs)`)  
}

async function onPlay() {
    const videoEl = $('inputVideo').get(0)

    if(videoEl.paused || videoEl.ended || !isFaceDetectionModelLoaded())
        return setTimeout(() => onPlay())

    const options = getFaceDetectorOptions()
    const ts = Date.now()
    const result = await faceapi.detectSingleFace(videoEl, options)
    updateTimeStats(Date.now() - ts)
```javascript
if (result) {
    const canvas = $('#overlay').get(0)
    const dims = faceapi.matchDimensions(canvas, videoEl, true)
    faceapi.draw.drawDetections(canvas, faceapi.resizeResults(result, dims))
}

setTimeout(() => onPlay())

async function run() {
    // load face detection model
    await changeFaceDetector(TINY_FACE_DETECTOR)
    changeInputSize(128)

    // try to access users webcam and stream the images
    // to the video element
    const stream = await navigator.mediaDevices.getUserMedia({ video: {} })
    const videoEl = $('#inputVideo').get(0)
    videoEl.srcObject = stream
}

function updateResults() {
}

$(document).ready(function() {
    renderNavBar('#navbar', 'webcam_face_detection')
    initFaceDetectionControls()
    run()
})
```

```html
</script>
</body>
</html>```
Webcam Age and Gender Recognition

```html
<!DOCTYPE html>
<html>
<head>
  <script src="face-api.js"></script>
  <script src="js/commons.js"></script>
  <script src="js/faceDetectionControls.js"></script>
  <link rel="stylesheet" href="styles.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.100.2/css/materialize.css">
  <script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.100.2/js/materialize.min.js"></script>
</head>
<body>
  <div id="navbar"></div>
  <div class="center-content page-container">
    <div class="progress" id="loader">
      <div class="indeterminate"></div>
    </div>
    <div style="position: relative" class="margin">
      <video onloadedmetadata="onPlay(this)" id="inputVideo" autoplay muted playsinline></video>
      <canvas id="overlay" />
    </div>
    <div class="row side-by-side">
      <!-- face_detector_selection_control -->
      <div id="face_detector_selection_control" class="row input-field">
        <select id="selectFaceDetector" style="margin-right: 20px;">
          <option value="ssd_mobilenetv1">SSD Mobilenet V1</option>
          <option value="tiny_face_detector">Tiny Face Detector</option>
          <option value="mtcnn">MTCNN</option>
        </select>
        <label>Select Face Detector</label>
      </div>
      <!-- check boxes -->
      <div class="row" style="width: 220px;">
        <input type="checkbox" id="hideBounding BoxesCheckbox" onchange="onChangeHideBoundingBoxes(event)" />
        <label for="hideBounding BoxesCheckbox">Hide Bounding Boxes</label>
      </div>
      <!-- fps_meter -->
      <div id="fps_meter" class="row side-by-side">
        <label for="time">Time:</label>
      </div>
    </div>
  </div>
</body>
</html>
```
<input disabled value="-" id="time" type="text" class="bold">
<label for="fps">Estimated Fps:</label>
<input disabled value="-" id="fps" type="text" class="bold">
</div>
</div>
<!-- fps_meter -->

<!-- ssd_mobilenetv1_controls -->
<span id="ssd_mobilenetv1_controls">
<div class="row side-by-side">
<div class="row">
<label for="minConfidence">Min Confidence:</label>
<input disabled value="0.5" id="minConfidence" type="text" class="bold">
</div>
</div>
<button class="waves-effect waves-light btn" onclick="onDecreaseMinConfidence()">
<i class="material-icons left">-</i>
</button>
<button class="waves-effect waves-light btn" onclick="onIncreaseMinConfidence()">
<i class="material-icons left">+</i>
</button>
</span>
<!-- ssd_mobilenetv1_controls -->

<!-- tiny_face_detector_controls -->
<span id="tiny_face_detector_controls">
<div class="row side-by-side">
<div class="row input-field" style="margin-right: 20px;">
<select id="inputSize">
<option value="" disabled selected>Input Size:</option>
<option value="128">128 x 128</option>
<option value="160">160 x 160</option>
<option value="224">224 x 224</option>
<option value="320">320 x 320</option>
<option value="416">416 x 416</option>
<option value="512">512 x 512</option>
<option value="608">608 x 608</option>
</select>
<label>Input Size</label>
</div>
<div class="row">
<label for="scoreThreshold">Score Threshold:</label>
<input disabled value="0.5" id="scoreThreshold" type="text" class="bold">
</div>
<button class="waves-effect waves-light btn"
<div class="row side-by-side">
  <div class="row">
    <label for="minFaceSize">Minimum Face Size:</label>
    <input disabled value="20" id="minFaceSize" type="text" class="bold">
  </div>
</div>

<button class="waves-effect waves-light btn" onclick="onDecreaseMinFaceSize()"
>
  <i class="material-icons left">-</i>
</button>

<button class="waves-effect waves-light btn" onclick="onIncreaseMinFaceSize()"
>
  <i class="material-icons left">+</i>
</button>

<script>
  let forwardTimes = []
  let predictedAges = []
  let withBoxes = true

  function onChangeHideBoundingBoxes(e) {
    withBoxes = !$(e.target).prop('checked')
  }

  function updateTimeStats(timeInMs) {
    forwardTimes = [timeInMs].concat(forwardTimes).slice(0, 30)
    const avgTimeInMs = forwardTimes.reduce((total, t) => total + t) / forwardTimes.length
    $('#time').val(`${Math.round(avgTimeInMs)} ms`)
    $('#fps').val(`${faceapi.round(1000 / avgTimeInMs)}`)
  }
</script>
```javascript
function interpolateAgePredictions(age) {
    const predictedAges = [age].concat(predictedAges).slice(0, 30)
    const avgPredictedAge = predictedAges.reduce((total, a) => total + a) / predictedAges.length
    return avgPredictedAge
}

async function onPlay() {
    const videoEl = $('#inputVideo').get(0)
    if (videoEl.paused || videoEl.ended || !isFaceDetectionModelLoaded())
        return setTimeout(() => onPlay())

    const options = getFaceDetectorOptions()
    const ts = Date.now()

    const result = await faceapi.detectSingleFace(videoEl, options)
        .withAgeAndGender()
    updateTimeStats(Date.now() - ts)
    if (result) {
        const canvas = $('#overlay').get(0)
        const dims = faceapi.matchDimensions(canvas, videoEl, true)

        const resizedResult = faceapi.resizeResults(result, dims)
        if (withBoxes) {
            faceapi.draw.drawDetections(canvas, resizedResult)
        }
        const { age, gender, genderProbability } = resizedResult
        // interpolate gender predictions over last 30 frames
        // to make the displayed age more stable
        new faceapi.draw.DrawTextField({
            'facetapi.round(interpolatedAge, 0)'} years',
            'facetapi.round(genderProbability))
        },
        result.detection.box.bottomLeft
        ).draw(canvas)
    }

    setTimeout(() => onPlay())
}

async function run() {
    // load face detection and face expression recognition models
    await changeFaceDetector(TINY_FACE_DETECTOR)
    await faceapi.nets.ageGenderNet.load('/')
    changeInputSize(224)

    // try to access users webcam and stream the images
    // to the video element
    const stream = await navigator.mediaDevices.getUserMedia({ video: {} })
```
```javascript
const videoEl = $('#inputVideo').get(0)
videoEl.srcObject = stream

function updateResults() {}

$(document).ready(function() {
  renderNavbar('#navbar', 'webcam_age_and_gender_recognition')
  initFaceDetectionControls()
  run()
})
</script>
</body>
</html>
```
Webcam Face Expression Recognition

```html
<!DOCTYPE html>
<html>
<head>

<script src="face-api.js"></script>
<script src="js/commons.js"></script>
<script src="js/faceDetectionControls.js"></script>
<link rel="stylesheet" href="styles.css">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.100.2/css/materialize.css">
<script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/0.100.2/js/materialize.min.js"></script>

</head>

<body>

<div id="navbar"></div>

<div class="center-content page-container">

<div class="progress" id="loader">
  <div class="indeterminate"></div>
</div>

<div style="position: relative" class="margin">
  <video onloadedmetadata="onPlay(this)" id="inputVideo" autoplay muted playsinline></video>
  <canvas id="overlay" />
</div>

<div class="row side-by-side">

  <!-- face_detector_selection_control -->
  <div id="face_detector_selection_control" class="row input-field" style="margin-right: 20px;">
    <select id="selectFaceDetector">
      <option value="ssd_mobilenetv1">SSD Mobilenet V1</option>
      <option value="tiny_face_detector">Tiny Face Detector</option>
      <option value="mtcnn">MTCNN</option>
    </select>
    <label>Select Face Detector</label>
  </div>

  <!-- check boxes -->
  <div class="row" style="width: 220px;">
    <input type="checkbox" id="hideBoundingBoxesCheckbox" onchange="onChangeHideBoundingBoxes(event)" />
    <label for="hideBoundingBoxesCheckbox">Hide Bounding Boxes</label>
  </div>

  <!-- fps_meter -->
  <div id="fps_meter" class="row side-by-side">
    <div>
      <label for="time">Time: </label>
    </div>
  </div>

</div>

</body>
</html>
```
<input disabled value="-" id="time" type="text" class="bold">
<label for="fps">Estimated Fps:</label>
<input disabled value="-" id="fps" type="text" class="bold">
</div>
</div>
<!-- fps_meter -->
</div>
<!-- ssd_mobilenetv1_controls -->
<span id="ssd_mobilenetv1_controls">
<div class="row side-by-side">
<div class="row">
<label for="minConfidence">Min Confidence:</label>
<input disabled value="0.5" id="minConfidence" type="text" class="bold">
</div>

<button class="waves-effect waves-light btn">
<i class="material-icons left">-</i>
</button>
<button class="waves-effect waves-light btn">
<i class="material-icons left">+</i>
</button>
</div>
</span>
<!-- ssd_mobilenetv1_controls -->

<!-- tiny_face_detector_controls -->
<span id="tiny_face_detector_controls">
<div class="row side-by-side">
<div class="row input-field" style="margin-right: 20px;">
<select id="inputSize">
<option value="" disabled selected>Input Size:</option>
<option value="128">128 x 128</option>
<option value="160">160 x 160</option>
<option value="224">224 x 224</option>
<option value="320">320 x 320</option>
<option value="416">416 x 416</option>
<option value="512">512 x 512</option>
<option value="608">608 x 608</option>
</select>
<label>Input Size</label>
</div>
<div class="row">
<label for="scoreThreshold">Score Threshold:</label>
<input disabled value="0.5" id="scoreThreshold" type="text" class="bold">
</div>
<button class="waves-effect waves-light btn"
onclick="onDecreaseScoreThreshold()"
>  
<i class="material-icons left">-</i>
</button>
<button  
class="waves-effect waves-light btn"
onclick="onIncreaseScoreThreshold()"
>  
<i class="material-icons left">+</i>
</button>
</div>
</span>
<!-- tiny_face_detector_controls -->

<!-- mtcnn_controls -->
<span id="mtcnn_controls">
<div class="row side-by-side">
<div class="row">
<label for="minFaceSize">Minimum Face Size:</label>
<input disabled value="20" id="minFaceSize" type="text" class="bold">
</div>
<button  
class="waves-effect waves-light btn"
onclick="onDecreaseMinFaceSize()"
>  
<i class="material-icons left">-</i>
</button>
<button  
class="waves-effect waves-light btn"
onclick="onIncreaseMinFaceSize()"
>  
<i class="material-icons left">+</i>
</button>
</div>
</span>
<!-- mtcnn_controls -->

</span>
</div>
</span>
<!-- mtcnn_controls -->

<script>
let forwardTimes = []
let withBoxes = true

function onChangeHideBoundingBoxes(e) {
  withBoxes = !$(e.target).prop('checked')
}

function updateTimeStats(timeInMs) {
  forwardTimes = [timeInMs].concat(forwardTimes).slice(0, 30)
  const avgTimeInMs = forwardTimes.reduce((total, t) => total + t) / forwardTimes.length
  $('#time').val(`Math.round(avgTimeInMs) ms`)  
  $('#fps').val(`faceapi.round(1000 / avgTimeInMs)`)  
}

async function onPlay() {


const videoEl = $('#inputVideo').get(0)

if(videoEl.paused || videoEl.ended || !isFaceDetectionModelLoaded())
    return setTimeout(() => onPlay())

const options = getFaceDetectorOptions()

const ts = Date.now()

const result = await faceapi.detectSingleFace(videoEl, options).withFaceExpressions()

updateTimeStats(Date.now() - ts)

if (result) {
    const canvas = $('#overlay').get(0)
    const dims = faceapi.matchDimensions(canvas, videoEl, true)

    const resizedResult = faceapi.resizeResults(result, dims)
    const minConfidence = 0.05
    if (withBoxes) {
        faceapi.draw.drawDetections(canvas, resizedResult)
    }
    faceapi.draw.drawFaceExpressions(canvas, resizedResult, minConfidence)
}

setTimeout(() => onPlay())
}

async function run() {
    // load face detection and face expression recognition models
    await changeFaceDetector(TINY_FACE_DETECTOR)
    await faceapi.loadFaceExpressionModel('')
    changeInputSize(224)

    // try to access users webcam and stream the images
    // to the video element
    const stream = await navigator.mediaDevices.getUserMedia({ video: [] })
    const videoEl = $('#inputVideo').get(0)
    videoEl.srcObject = stream
}

function updateResults() {}

$(document).ready(function() {
    renderNavBar('#navbar', 'webcam_face_expression_recognition')
    initFaceDetectionControls()
    run()
})
</script>
</body>
</html>