

智慧学习与智慧教育 | 基于face-api.js

# 情绪识别系统

Real-Time Facial Emotion Recognition System



## 小组成员

## 分工

张柏兰

界面与其他功能实现、  
图片情绪识别、ppt、文档

周一卓

人脸识别+情绪实时识别、文档

赵 穗

资料搜索、文档

余佳浩

文档、文献搜索、数据集预处理



# 主要內容

01

## 情緒识别介绍

Introduction to Emotion Recognition

02

## 相关工作

Related work on Emotion Recognition

03

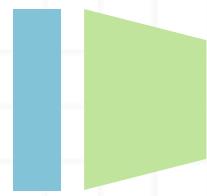
## 模型与系统介绍

Introduction to the model and system

04

## 总结与反思

Summary and reflection

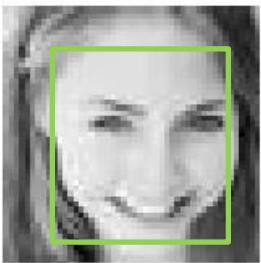


# 人类的七种情绪

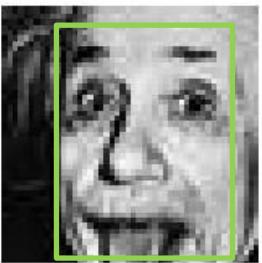
The Seven Emotions of Humanity



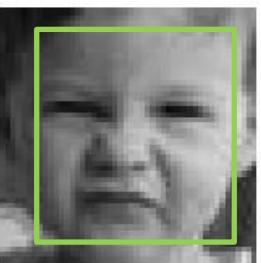
开心 (happy)



Happy



Surprise



Disgust



生气 (angry)



Anger



Fear



Neutral



中立 (neutral)



厌恶 (disgusted)



伤心 (sad)

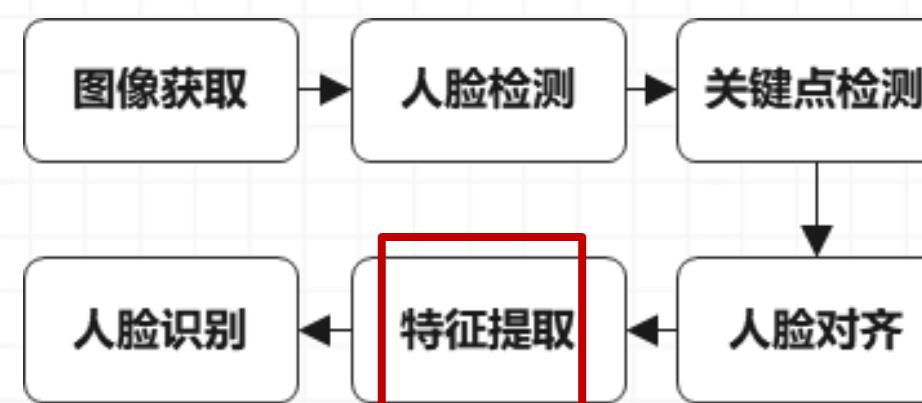


惊讶 (surprised)



恐惧 (fearful)

人脸识别 → 情绪识别



## ✓ 任务描述

开发一个基于面部表情的情绪分析系统，系统需要能够**实时捕捉**人脸并分析情绪，**输出**分析结果。

## ✓ 实现与拓展

借助**HTML、CSS、JavaScript**和**MySQL**搭建**Emotion-Recognition**系统，实现登录、注册功能。

登录后可以根据按钮选择**上传照片**进行情绪分析或打开摄像头进行**实时分析**；同时输出识别后对应的人名和情绪。

# 设计流程

Design process

## 基于人脸识别的面部情绪识别

### 人脸识别与定位

人脸识别

### 人脸特征提取

### 面部情绪识别

对待识别图像进行扫描，进行人脸检测，辨别其中是否含有人脸；  
当检测到图像含有人脸时，确定人脸的位置、大小；

对步骤（1）检测到的人脸图像进行特征提取和人脸图像分割分析，  
确定人脸的基本轮廓和面部器官的位置、轮廓；

提取面部器官特征，并分解出特征形状，  
根据面部器官的轮廓形状设定情感判断条件，判断人物情感情义。

# 设计流程

Design process

文献调研与资料查询

Facenet/Facenet512

VGGFace

DeepFace

OpenFace

DeepID

Dlib

模型选择与性能对比

分析并总结各种人脸识别模型的优缺点。  
精度和速度之间达到最佳平衡的模型作为我们的最终选择。

ArcFace

SFace

GhostFaceNet

模型转换与优化

将选定模型的权重和架构转换为适合前端使用的格式。

TensorFlow.js格式

前端界面设计与开发

设计并实现用户界面，确保界面友好、直观且响应迅速。

HTML

CSS

JavaScript

模型集成与系统实现

将转换后的情绪识别模型集成到前端界面中，实现实时性和高效性，优化代码提高性能。

测试与验证

在实际应用场景中进行测试，根据测试反馈进行优化和改进，提升系统的用户体验和可靠性。

# 数据集

Datasets

## FER2013



由**35886**张人脸不同表情图片组成，其中训练集**28708**张，验证集和测试集各**3589**张。

每张图片的大小是48\*48px，表情分为7种：

0 anger 生气； 1 disgust 厌恶； 2 fear 恐惧； 3 happy 开心； 4 normal 中立； 5 sad 伤心； 6 surprised 惊讶



# 情绪识别-人脸识别模型

Emotion recognition – Face Recognition



## Facenet/Facenet512

**FaceNet** 是一个由 **Google** 研究团队开发的人脸识别系统，它基于深度学习技术，可以实现高精度的人脸识别、验证和聚类任务。FaceNet 通过学习直接从图像像素到人脸嵌入的映射，使得它在各种人脸识别任务中表现出色。

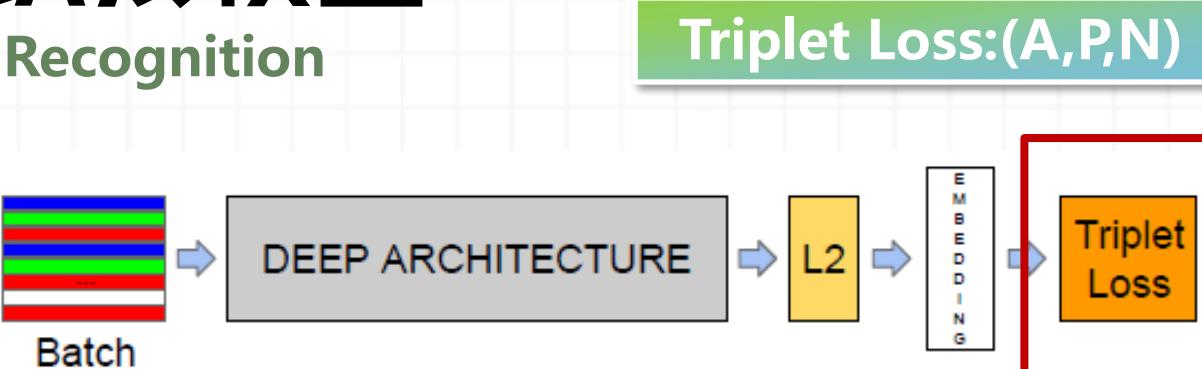


Figure 2. **Model structure.** Our network consists of a batch input layer and a deep CNN followed by  $L_2$  normalization, which results in the face embedding. This is followed by the triplet loss during training.

### 三元组损失函数的目标

使得同一身份的嵌入向量之间的距离比不同身份的嵌入向量之间的距离小于一个固定的边界值 (margin)。

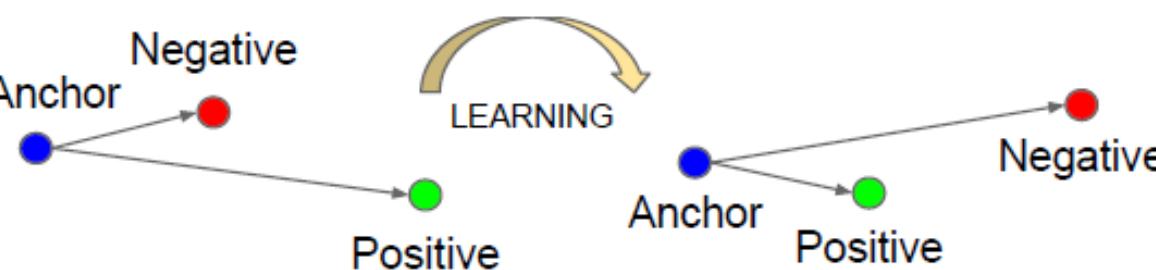


Figure 3. The **Triplet Loss** minimizes the distance between an *anchor* and a *positive*, both of which have the same identity, and maximizes the distance between the *anchor* and a *negative* of a different identity.

**Anchor (A)** : 基准人脸图像。

layer	size-in	size-out	kernel	param	FLPS
conv1	$220 \times 220 \times 3$	$110 \times 110 \times 64$	$7 \times 7 \times 3, 2$	9K	115M
pool1	$110 \times 110 \times 64$	$55 \times 55 \times 64$	$3 \times 3 \times 64, 2$	0	0
rnorm1	$55 \times 55 \times 64$	$55 \times 55 \times 64$			
conv2a	$55 \times 55 \times 64$	$55 \times 55 \times 64$	$1 \times 1 \times 64, 1$	4K	13M
conv2	$55 \times 55 \times 64$	$55 \times 55 \times 192$	$3 \times 3 \times 64, 1$	111K	335M
rnorm2	$55 \times 55 \times 192$	$55 \times 55 \times 192$		0	
pool2	$55 \times 55 \times 192$	$28 \times 28 \times 192$	$3 \times 3 \times 192, 2$	0	
conv3a	$28 \times 28 \times 192$	$28 \times 28 \times 192$	$1 \times 1 \times 192, 1$	37K	29M
conv3	$28 \times 28 \times 192$	$28 \times 28 \times 384$	$3 \times 3 \times 192, 1$	664K	521M
pool3	$28 \times 28 \times 384$	$14 \times 14 \times 384$	$3 \times 3 \times 384, 2$	0	
conv4a	$14 \times 14 \times 384$	$14 \times 14 \times 384$	$1 \times 1 \times 384, 1$	148K	29M
conv4	$14 \times 14 \times 384$	$14 \times 14 \times 256$	$3 \times 3 \times 384, 1$	885K	173M
conv5a	$14 \times 14 \times 256$	$14 \times 14 \times 256$	$1 \times 1 \times 256, 1$	66K	13M
conv5	$14 \times 14 \times 256$	$14 \times 14 \times 256$	$3 \times 3 \times 256, 1$	590K	116M
conv6a	$14 \times 14 \times 256$	$14 \times 14 \times 256$	$1 \times 1 \times 256, 1$	66K	13M
conv6	$14 \times 14 \times 256$	$14 \times 14 \times 256$	$3 \times 3 \times 256, 1$	590K	116M
pool4	$14 \times 14 \times 256$	$7 \times 7 \times 256$	$3 \times 3 \times 256, 2$	0	
concat	$7 \times 7 \times 256$	$7 \times 7 \times 256$		0	
fc1	$7 \times 7 \times 256$	$1 \times 32 \times 128$	maxout p=2	103M	103M
fc2	$1 \times 32 \times 128$	$1 \times 32 \times 128$	maxout p=2	34M	34M
fc7128	$1 \times 32 \times 128$	$1 \times 1 \times 128$		524K	0.5M
L2	$1 \times 1 \times 128$	$1 \times 1 \times 128$		0	
total				140M	1.6B

Table 1. **NN1.** This table show the structure of our Zeiler&Fergus [22] based model with  $1 \times 1$  convolutions inspired by [9]. The input and output sizes are described in  $rows \times cols \times \#filters$ . The kernel is specified as  $rows \times cols, stride$  and the maxout [6] pooling size as  $p = 2$ .

# 情绪识别-人脸识别模型

Emotion recognition – Face Recognition



## | VGGFace

**VGGFace** 是基于 **VGGNet** 训练得到的人脸识别模型。

先使用 **Softmax** 在 **VGGFace2\_dataset (论文)** 上预训练，最终输出维度是 2622 维，代表 2622 个人（每个人有固定张数的人脸图像），然后再使用 **Triplet Loss** 训练出 1024 维的人脸特征向量。

度量学习，学习独特性和紧凑性

**VGGFace** 用少量数据训练性能就不错的原因

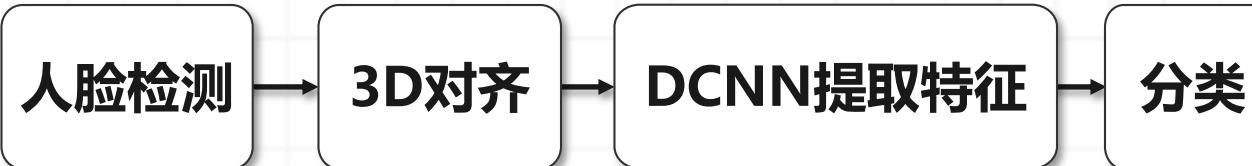
- (1) VGGFace 使用的 **backbone** 分类 (**ConvNet**) 性能较好。
- (2) 使用了先分类后度量学习的 **fine-tuning** 策略。
- (3) 使用的数据集更加结构化。

Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as “conv<receptive field size>-<number of channels>”. The ReLU activation function is not shown for brevity.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
conv3-64	conv3-64 LRN	conv3-64 <b>conv3-64</b>	conv3-64 conv3-64	conv3-64	conv3-64 conv3-64
		maxpool			
conv3-128	conv3-128	conv3-128 <b>conv3-128</b>	conv3-128 conv3-128	conv3-128	conv3-128 conv3-128
		maxpool			
conv3-256	conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 <b>conv1-256</b>	conv3-256	conv3-256 conv3-256 conv3-256 <b>conv3-256</b>
		maxpool			
conv3-512	conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 <b>conv1-512</b>	conv3-512	conv3-512 conv3-512 <b>conv3-512</b>
		maxpool			
conv3-512	conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 <b>conv1-512</b>	conv3-512	conv3-512 conv3-512 <b>conv3-512</b>
		maxpool			
		FC-4096			
		FC-4096			
		FC-1000			
		soft-max			

# 情绪识别-人脸识别模型

Emotion recognition – Face Recognition



**DeepFace** 是 **FaceBook** 提出来的，通过深度卷积神经网络（**DCNN**）实现高精度的人脸识别。

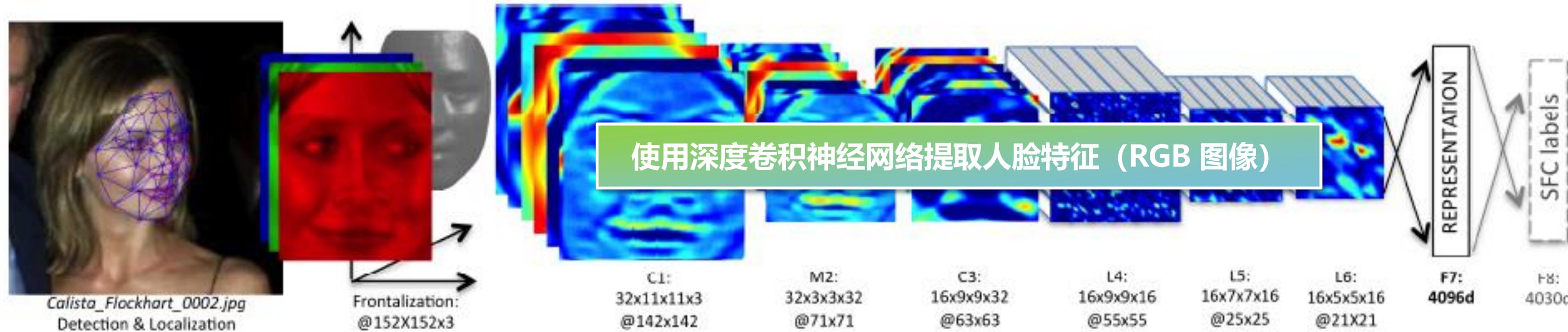


Figure 2. Outline of 3D仿射变换实现人脸对齐

Figure 2 shows the outline of 3D affine transformation for face alignment. The diagram illustrates the DeepFace architecture, starting with a detected face image (*Calista\_Flockhart\_0002.jpg*) and progressing through detection and localization, frontalization (@152X152x3), and feature extraction using a DCNN. The DCNN consists of a single convolution-pooling-convolution layer followed by three locally-connected layers and two fully-connected layers. The feature maps produced at each layer are color-coded. The network includes over 120 million parameters, with the majority coming from the local and fully connected layers.

## 交叉熵损失 (Cross-Entropy Loss)

最小化真实类别和预测类别之间的差异来优化网络参数。

DeepFace采用了基于检测点的人脸检测方法

- (1) 先选择6个基准点: (*fiducial Point Detector*)  
**2只眼睛中心、1个鼻子点、3个嘴上的点。**
- (2) 通过LBP特征用SVR来学习得到基准点。

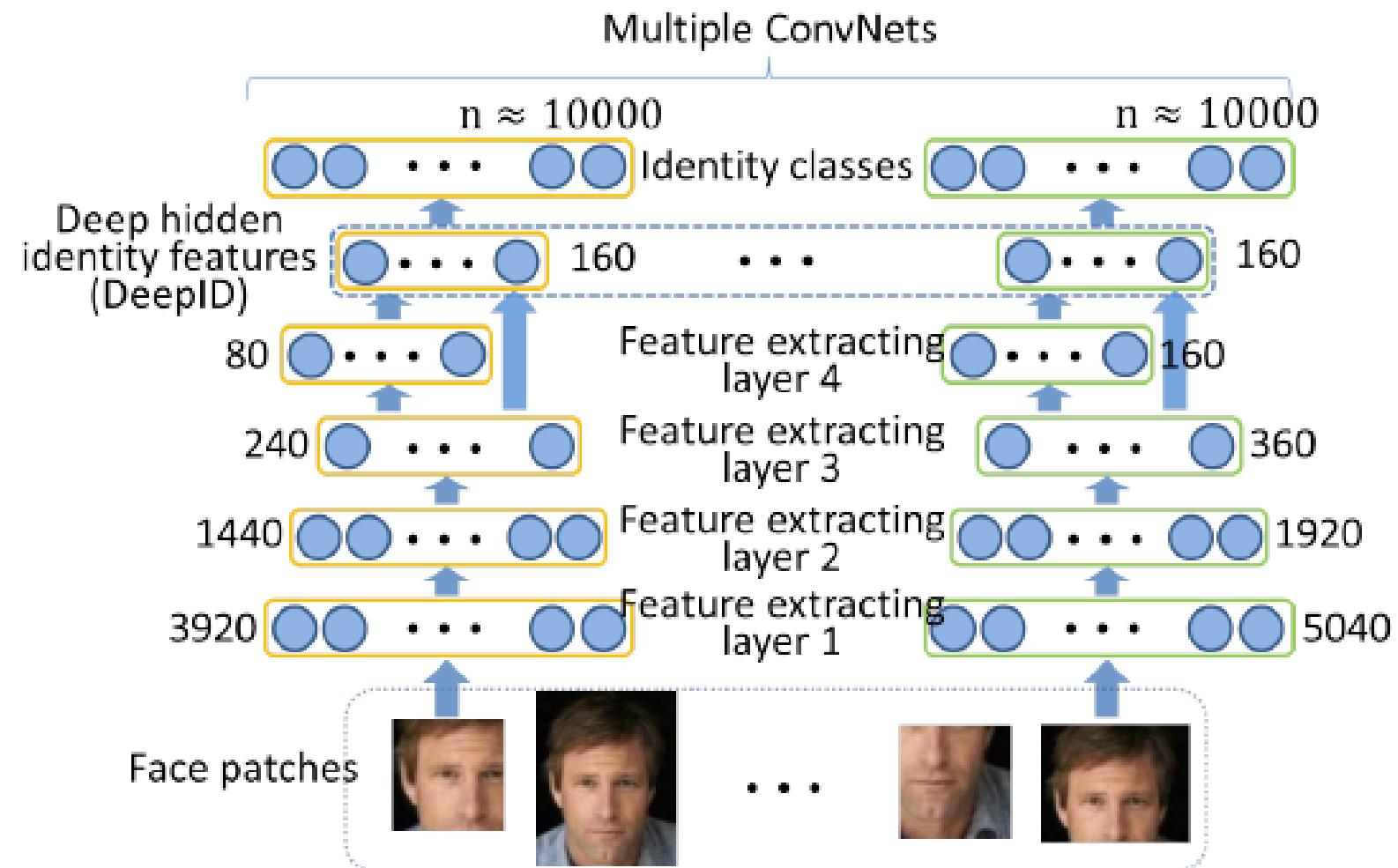
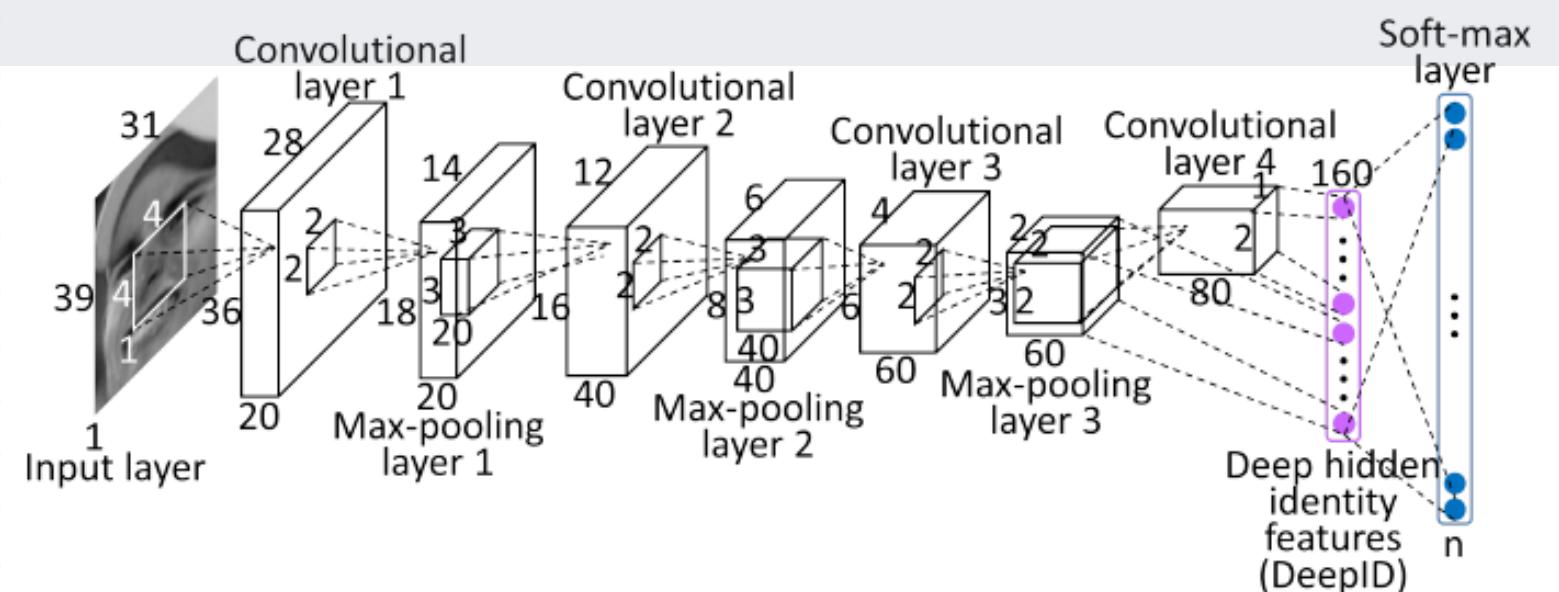
特征表示和匹配：  
使用余弦相似度或欧氏  
距离计算不同人脸特征  
向量之间的相似度。

# 情绪识别-人脸识别模型

Emotion recognition – Face Recognition

## ★ | DeepID

**DeepID (Deep Learning Face Representation by Joint Identification-Verification)** 是一种特征提取的算法，本文提出一种使用深度学习提取人脸**深层次特征**的方法，将身份验证和身份确认联合起来训练深度神经网络，旨在提高人脸识别的精度。



## DeepID 的网络结构

- (1) 包括多个卷积层和全连接层，用于提取人脸特征，生成一个特征向量 (DeepID)，用于后续的身份验证和确认任务。
- (2) 同时进行身份验证和身份确认的联合训练：  
**身份验证 (Identification)**：将模型作为一个多分类问题进行训练，每个人脸对应一个类别。使用交叉熵损失函数来优化模型。  
**身份确认 (Verification)**：使用对比损失 (Contrastive Loss)来优化模型，使同一人的人脸特征向量距离较近，不同人的人脸特征向量距离较远。

# 情绪识别-人脸识别模型

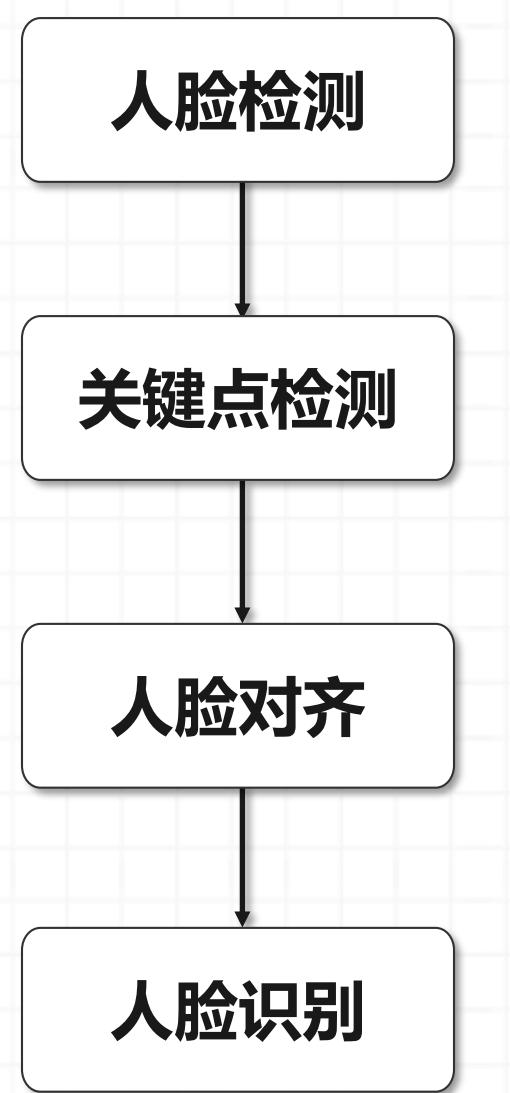
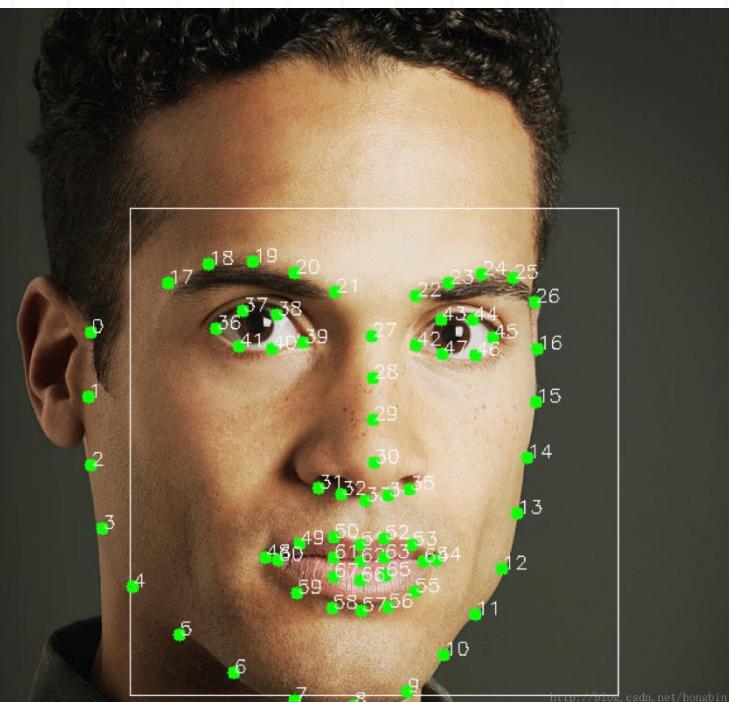
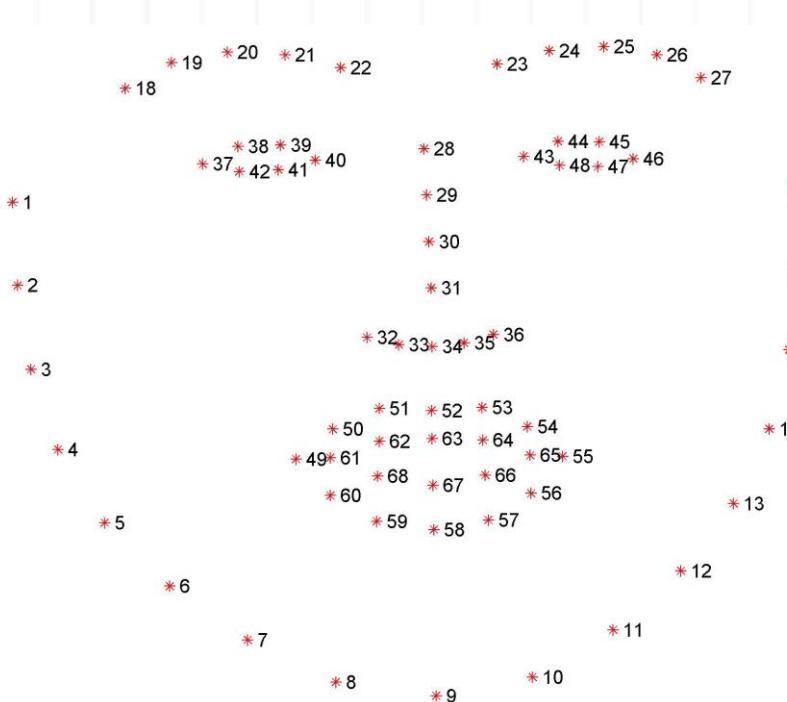
Emotion recognition – Face Recognition



| Dlib

Dlib C++ Library

**Dlib** 是一个包含机器学习算法的C++开源工具包。其中人脸识别部分使用经典的直方图定向梯度(**HOG**)特征，结合线性分类器，图像金字塔和滑动窗口检测方案。



HOG 特征描述了图像的局部梯度方向分布，通过线性 SVM 分类器检测图像中的人脸区域。

提供 5 点和 68 点的面部关键点检测器，用于检测如眼睛、鼻子嘴巴等。

将检测到的人脸关键点对齐到标准模板，以消除姿态、尺度和旋转的影响。

基于深度残差网络（ResNet），将人脸转换为特征向量（embeddings），通过比较特征向量进行人脸识别。

# 情绪识别-人脸识别模型

Emotion recognition – Face Recognition



集成了较为先进的面部行为分析算法

**OpenFace**是一个基于深度学习的开源面部识别和表情分析库，由*D Carnegie Mellon University*的研究人员创建并维护。它为开发者和研究人员提供了一整套工具，用于实时地捕捉、跟踪和分析面部运动，是AI和计算机视觉领域的一个强大工具。

OpenFace的核心在于它的深度神经网络模型，该模型经过大量面部图像数据训练，能够精确地识别出人脸的关键点，如眼睛、鼻子、嘴巴的位置，并且可以追踪面部肌肉的细微变化以识别人类的情绪和表情。



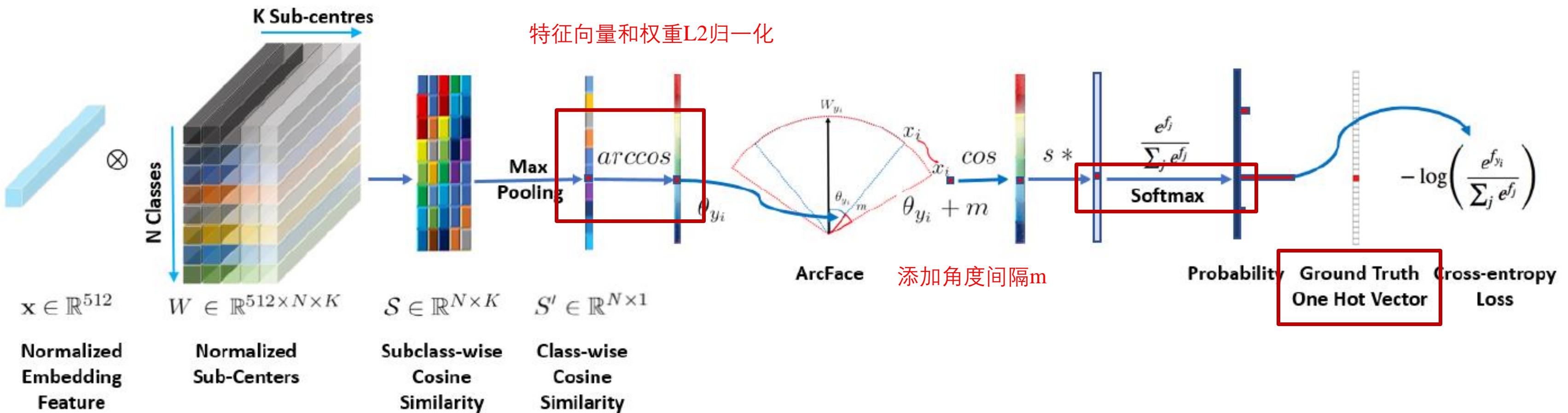
Fig. 1: OpenFace 2.0 is a framework that implements modern facial behavior analysis algorithms including: facial landmark detection, head pose tracking, eye gaze and facial action unit recognition.

# 情绪识别-人脸识别模型

Emotion recognition – Face Recognition

## ★ | ArcFace

**ArcFace** 提出一种新的用于人脸识别的损失函数：***additive angular margin loss***。



# 情绪识别-人脸识别模型

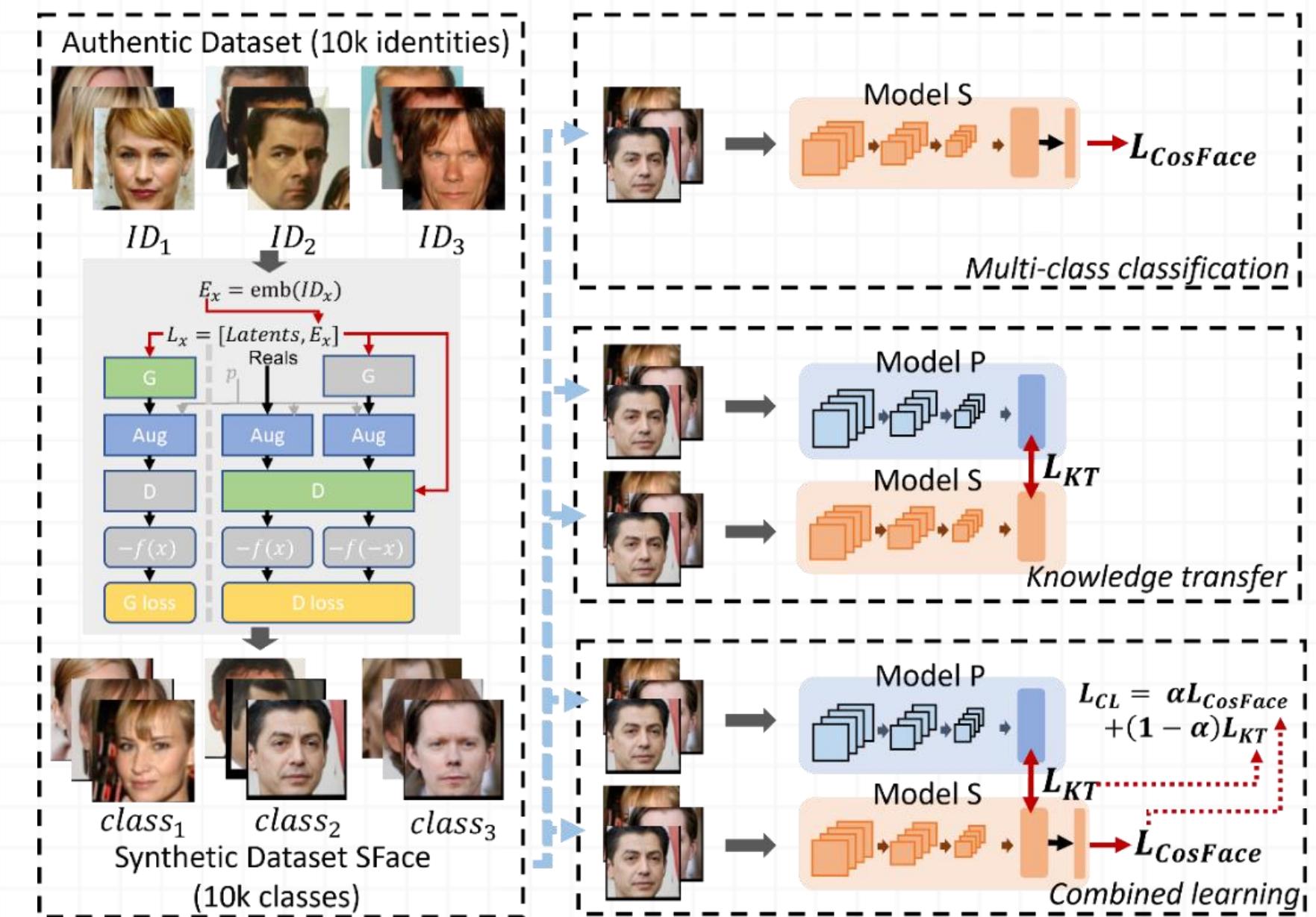
Emotion recognition – Face Recognition

## ★ | SFace

**Sface** 结合 **anchor-based** 方法和 **anchor-free** 的方法，来缓解人脸检测问题中尺度变化较大（Large Scale Variations）的问题。

Anchor-based方法: 在输入图像上预定义一组固定大小和纵横比的锚框（anchor boxes），这些锚框用于预测目标的位置和类别。这些锚框覆盖了不同尺度和形状的潜在目标区域。检测器会根据这些锚框调整预测框的位置和大小，并对目标进行分类。

Anchor-free方法: 不依赖预定义的锚框，而是直接在特征图上预测目标的中心点、尺寸和类别。这种方法通常通过检测目标的关键点或中心点，并回归目标的边界框来实现目标检测。

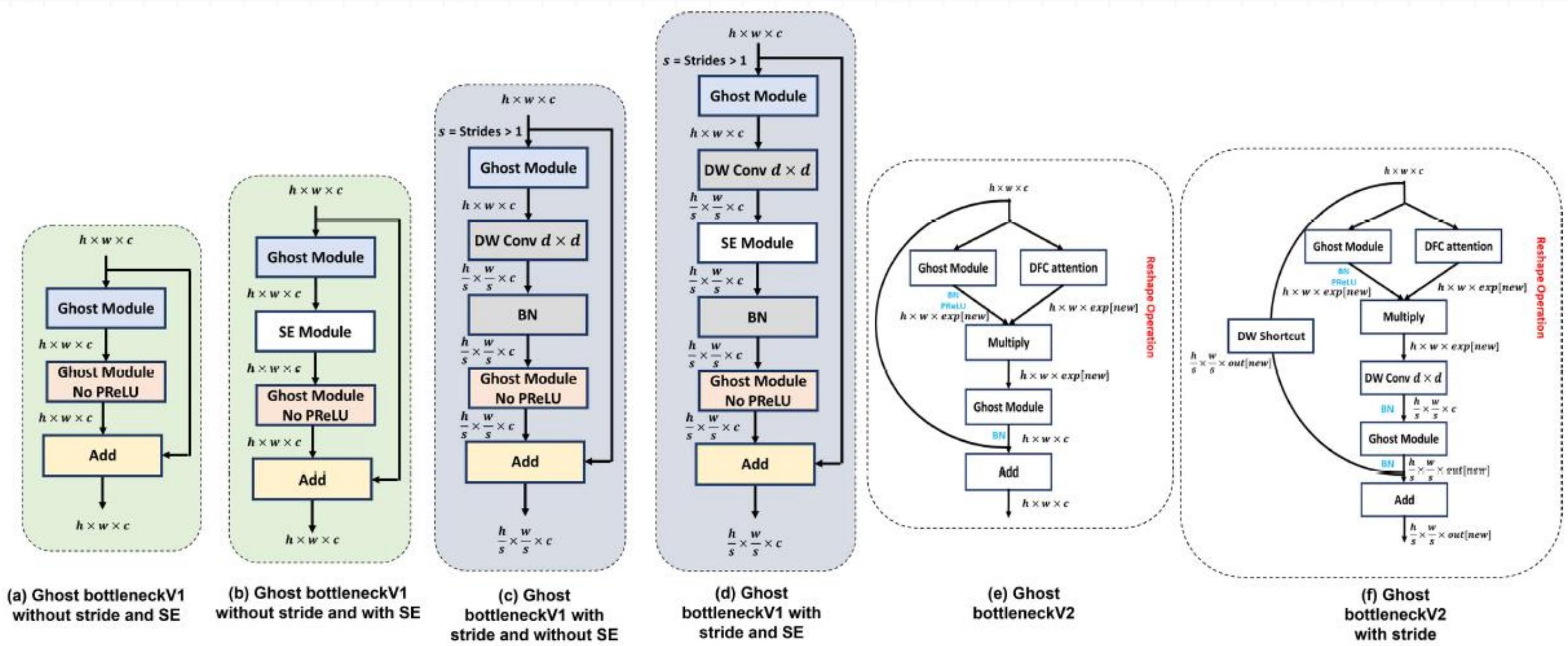


# 情绪识别-人脸识别模型

Emotion recognition – Face Recognition

## ★ | GhostFaceNet

***GhostFaceNet*** 的主要思想是卷积过程中，各个通道的卷积特征图高度相似，因此提出对**部分通道进行卷积**，将卷积得到的特征图进行线性变换得到剩余通道的特征图，减少特征映射冗余，减少计算量。



# 模型性能对比

## Model performance comparison

**Steps:** 选择两张包含相似人脸的不同图片，使用不同模型，进行100次训练，取均值，分别作为模型在单张图片上的耗时和模型对人脸特征欧式距离计算的准确度

### 耗时对比

model\_name和detector\_backend分别决定了用于人脸特征提取的模型和用于人脸检测的方法。

detector_backend model_name	VGG-Face	Facenet	Facenet512	OpenFace	DeepFace	DeepID	Dlib	ArcFace	SFace	GhostFaceNet
<b>opencv</b>	0.764	0.828	0.948	0.523	3.317	0.351	0.801	0.472	0.353	0.859
<b>retinaface</b>	3.206	3.279	3.353	2.975	5.749	2.813	3.202	1.224	2.815	3.501
<b>mtcnn</b>	2.4	2.249	2.254	1.93	4.672	1.78	2.164	2.146	1.788	2.696
<b>ssd</b>	0.719	0.85	0.779	0.478	3.28	0.307	0.518	0.692	0.303	0.95
<b>dlib</b>	1.348	1.464	1.4	1.086	4.57	0.99	1.247	1.291	0.912	1.447
<b>mediapipe</b>	0.492	0.541	0.539	0.235	2.986	0.06	0.406	0.437	0.067	0.68
<b>yolov8</b>	1.344	1.414	1.422	1.097	4.504	0.931	1.278	1.308	0.975	1.47
<b>centerface</b>	1.077	1.126	1.135	0.836	3.58	0.66	1.017	1.041	0.67	1.2
<b>skip</b>	<b>0.456</b>	<b>0.514</b>	<b>0.52</b>	<b>0.216</b>	<b>2.968</b>	<b>0.042</b>	<b>0.387</b>	<b>0.416</b>	<b>0.045</b>	<b>0.563</b>

# 模型性能对比

## Model performance comparison

**Steps:**选择两张包含相似人脸的不同图片，使用不同模型，跑100次取均值分别作为模型在单张图片上的耗时和模型对人脸特征欧式距离计算的准确度。

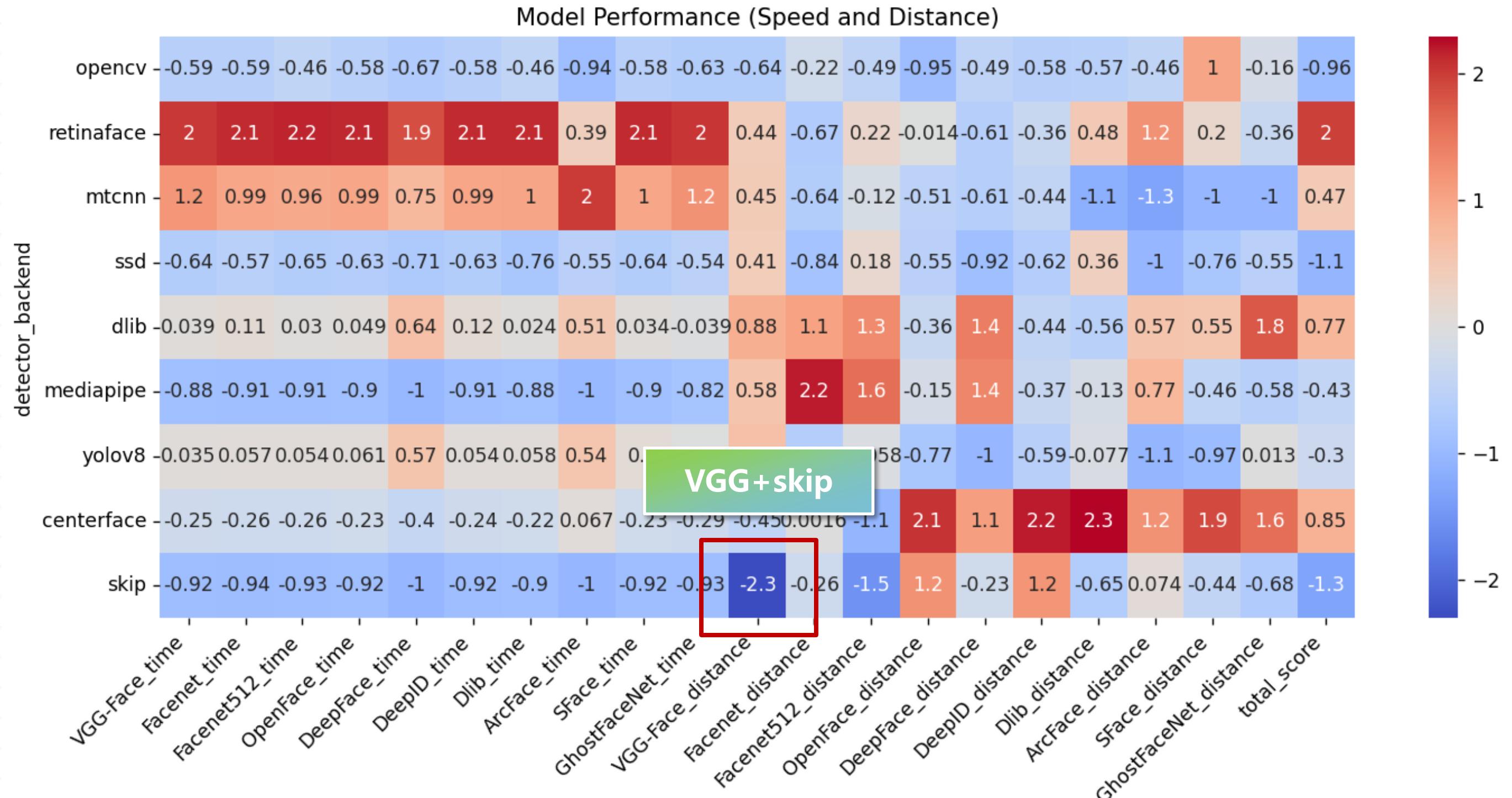
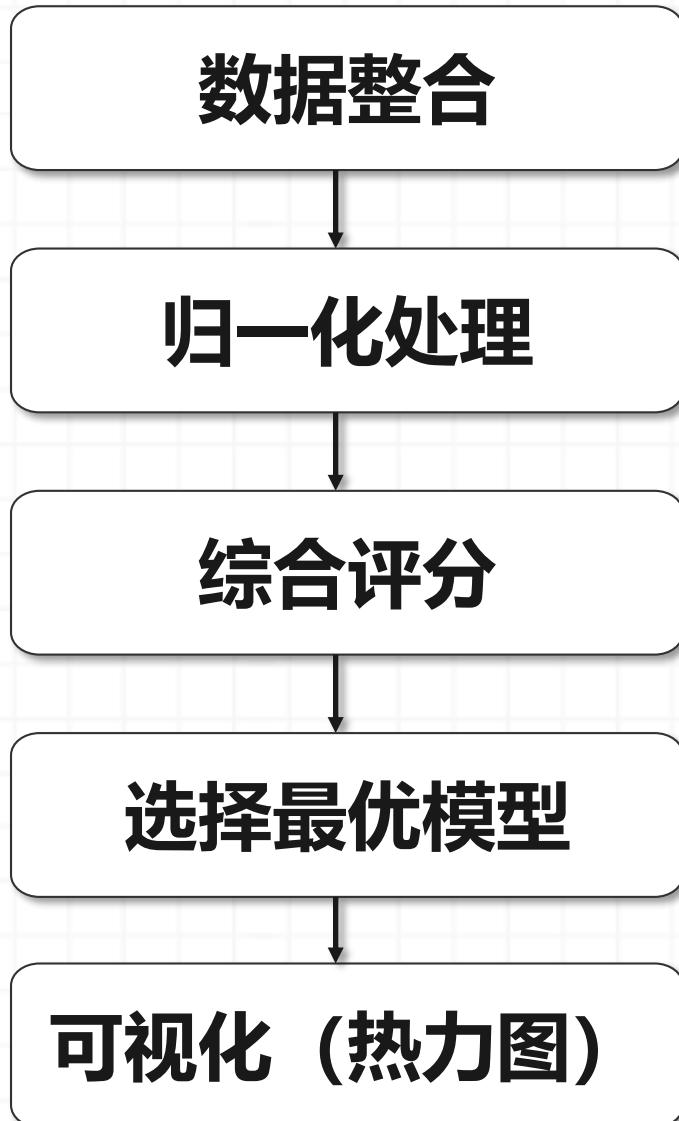
### 欧式距离计算对比

model\_name和detector\_backend分别决定了用于人脸特征提取的模型和用于人脸检测的方法。

model_name detector_backend	VGG-Face	Facenet	Facenet512	OpenFace	DeepFace	DeepID	Dlib	ArcFace	SFace	GhostFaceNet
opencv	0.530548501	0.28928425	0.497800398	0.08695239	0.19976334	0.0032641	0.02997704	0.472	0.493046528	0.609681693
retinaface	0.595583354	0.216934544	0.589549064	0.23544326	0.18234094	0.0083936	0.048041588	0.779864931	0.421562223	0.587752961
mtcnn	0.59637549	0.222431461	0.544791043	0.15690752	0.18174523	0.0065688	0.020306644	0.317205264	0.312503377	0.510747019
ssd	0.593963333	0.191180288	0.584222644	0.15100345	0.13685657	0.0024291	0.045896722	0.366076035	0.336122704	0.565851494
dlib	0.622220924	0.493154546	0.727784965	0.18084235	0.48171743	0.0064778	0.030179224	0.660581518	0.452508678	0.829912433
mediapipe	0.603722691	0.676149577	0.764532704	0.21419191	0.46985753	0.0080482	0.03751549	0.696160923	0.363204776	0.562543862
yolov8	0.607281813	0.225613191	0.553248635	0.11585069	0.12011066	0.0030753	0.038467413	0.362082374	0.317913476	0.629811349
centerface	0.541882457	0.325057692	0.423777992	0.56733494	0.43041584	0.0682482	0.079094878	0.779864931	0.573449554	0.80832582
skip	0.430994896	0.283478838	0.361540216	0.43075707	0.23773311	0.0447385	0.028647983	0.569562759	0.364811238	0.550621891

# 模型性能对比

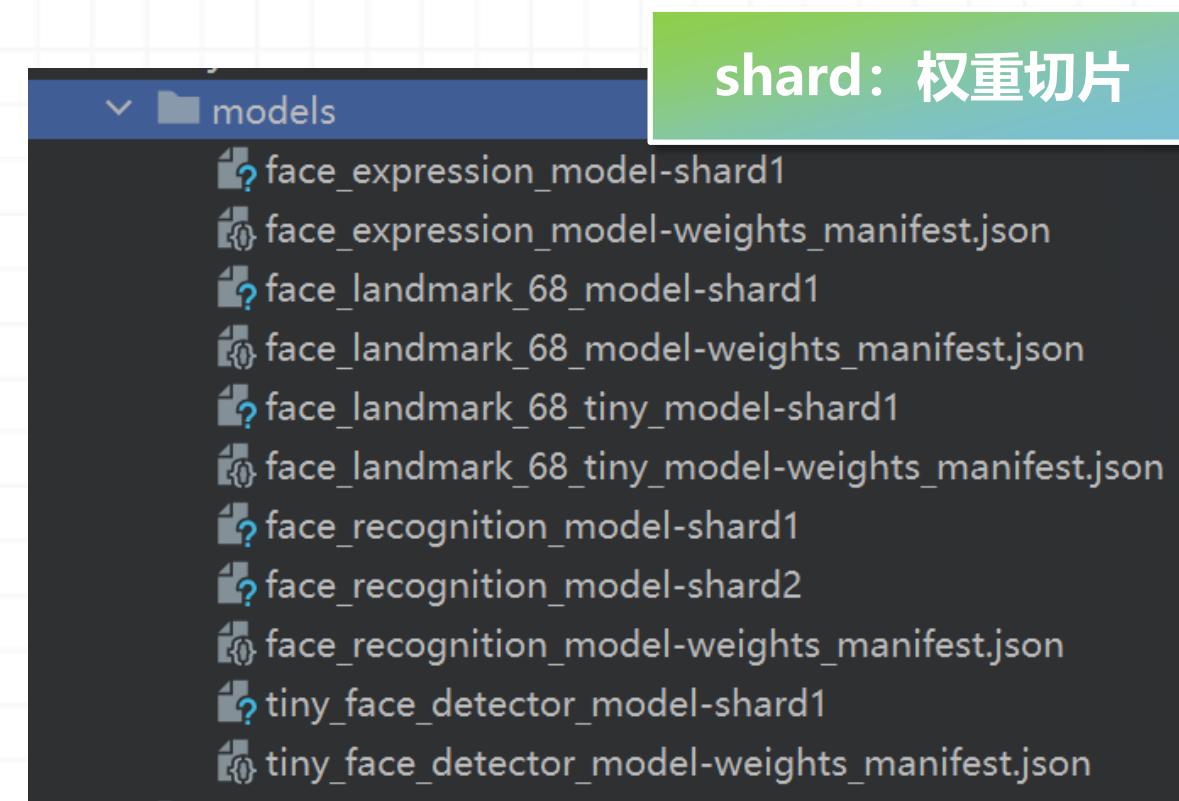
Model performance comparison



# 模型转化与优化

Model transformation and optimization

✓ 使用TensorFlow.js加载并运行情绪识别模型，实现实时视频流的情绪分析。



✓ 显示情绪分析结果，包括情绪类别和置信度，支持图表和文字描述。

## TensorFlow.js 是一个用于使用 JavaScript 进行机器学习开发的库

使用 JavaScript 开发机器学习模型，并直接在浏览器或 Node.js 中使用机器学习模型。

[查看教程](#)

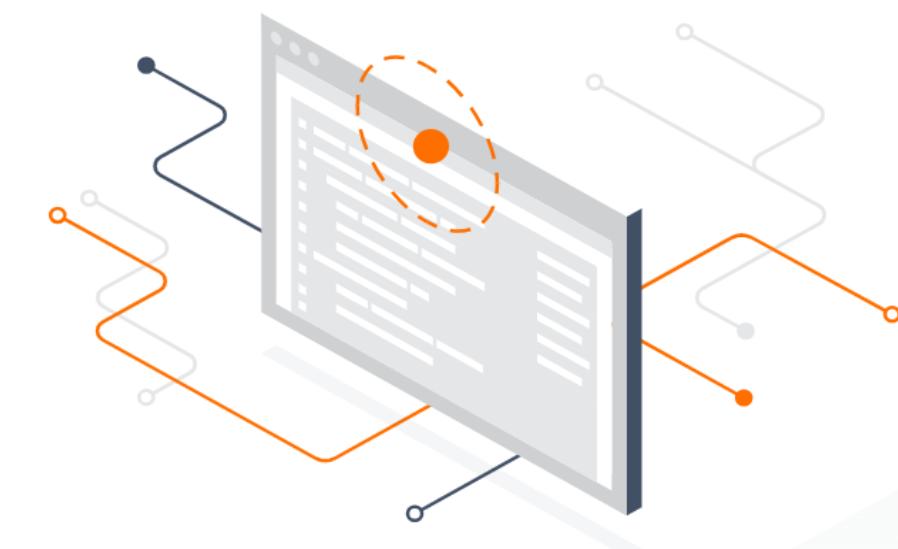
[查看模型](#)

[查看演示](#)

教程将通过完整的端到端示例向您展示如何使用 TensorFlow.js。

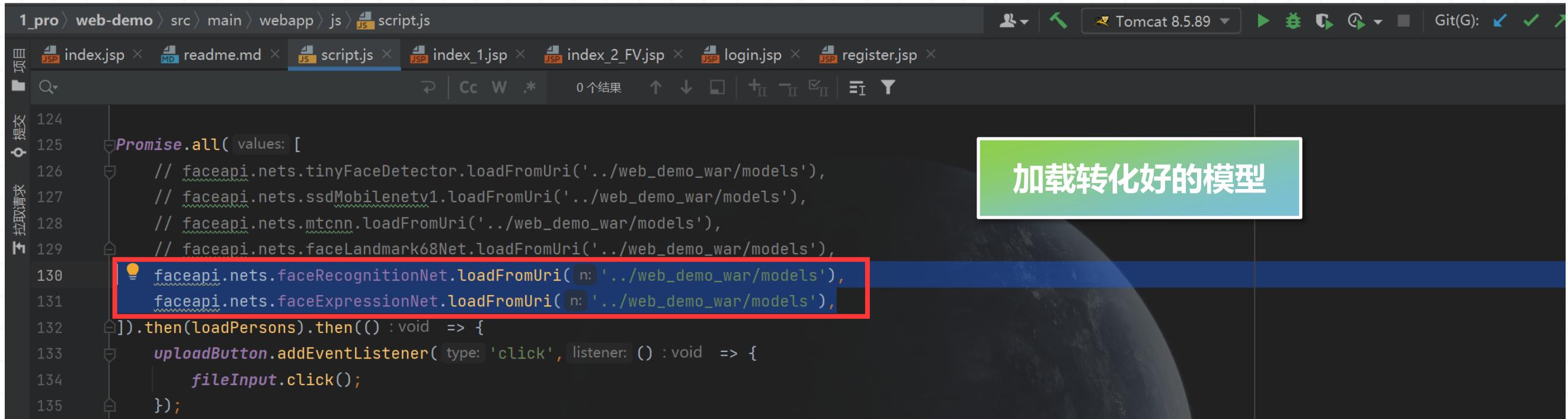
经过预先训练的开箱即用模型，适用于常见用例。

使用 TensorFlow.js 在浏览器中运行的在线演示和示例。



# 相关函数

## Introduction of Function

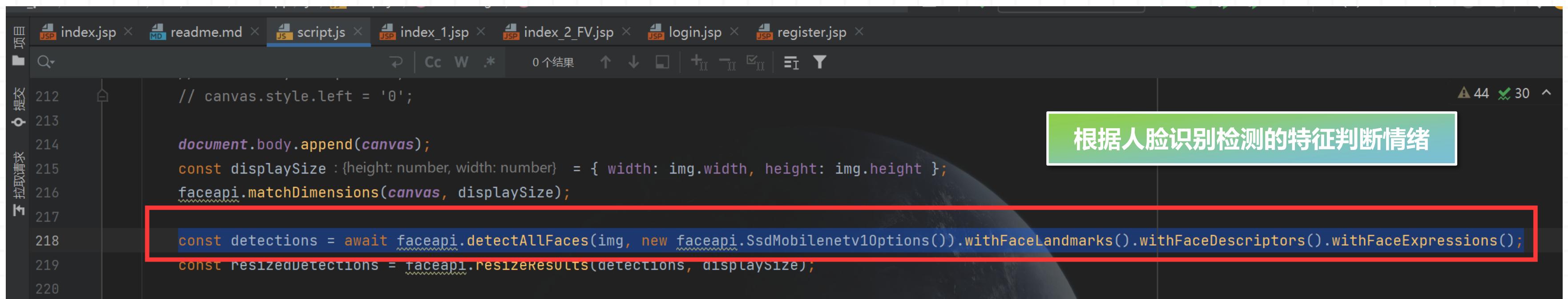


1\_pro > web-demo > src > main > webapp > js > script.js

```
124
125     Promise.all( values: [
126         // faceapi.nets.tinyFaceDetector.loadFromUri('../web_demo_war/models'),
127         // faceapi.nets.ssdMobilenetv1.loadFromUri('../web_demo_war/models'),
128         // faceapi.nets.mtcnn.loadFromUri('../web_demo_war/models'),
129         // faceapi.nets.faceLandmark68Net.loadFromUri('../web_demo_war/models'),
130         faceapi.nets.faceRecognitionNet.loadFromUri( n: '../web_demo_war/models'),
131         faceapi.nets.faceExpressionNet.loadFromUri( n: '../web_demo_war/models'),
132     ]).then(loadPersons).then(() : void => {
133         uploadButton.addEventListener( type: 'click', listener: () : void => {
134             inputFile.click();
135         });
136     });
137 
```

加载转化好的模型

## Script.js



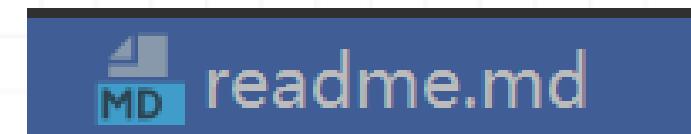
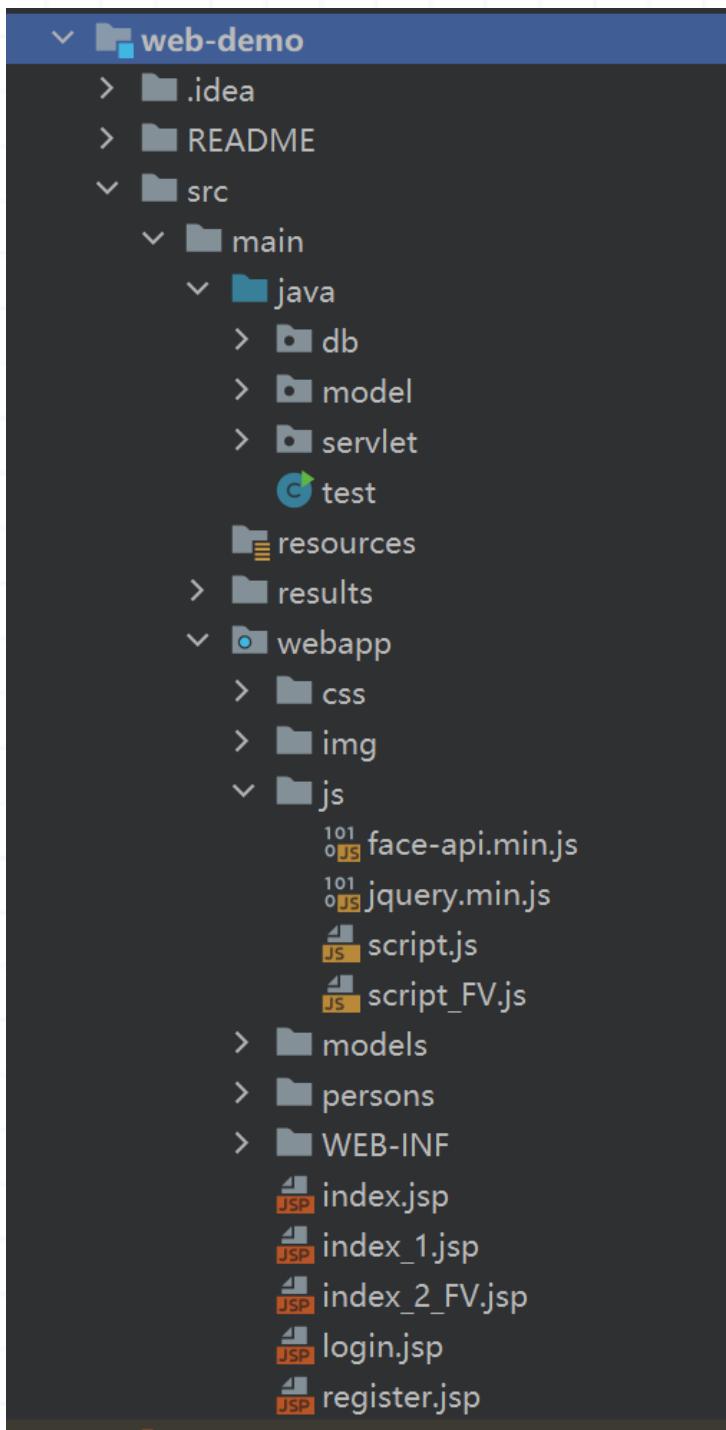
1\_pro > web-demo > src > main > webapp > js > script.js

```
212     // canvas.style.left = '0';
213
214     document.body.append(canvas);
215     const displaySize :{height: number, width: number} = { width: img.width, height: img.height };
216     faceapi.matchDimensions(canvas, displaySize);
217
218     const detections = await faceapi.detectAllFaces(img, new faceapi.SsdMobilenetv1Options()).withFaceLandmarks().withFaceDescriptors().withFaceExpressions();
219     const resizedDetections = faceapi.resizeResults(detections, displaySize);
220 
```

根据人脸识别检测的特征判断情绪

# 项目结构

Introduction of



代码已上传到github上：[https://github.com/Altyabbi/Emotion\\_Recognition.git](https://github.com/Altyabbi/Emotion_Recognition.git)

# 😊 Emotion Recognition System

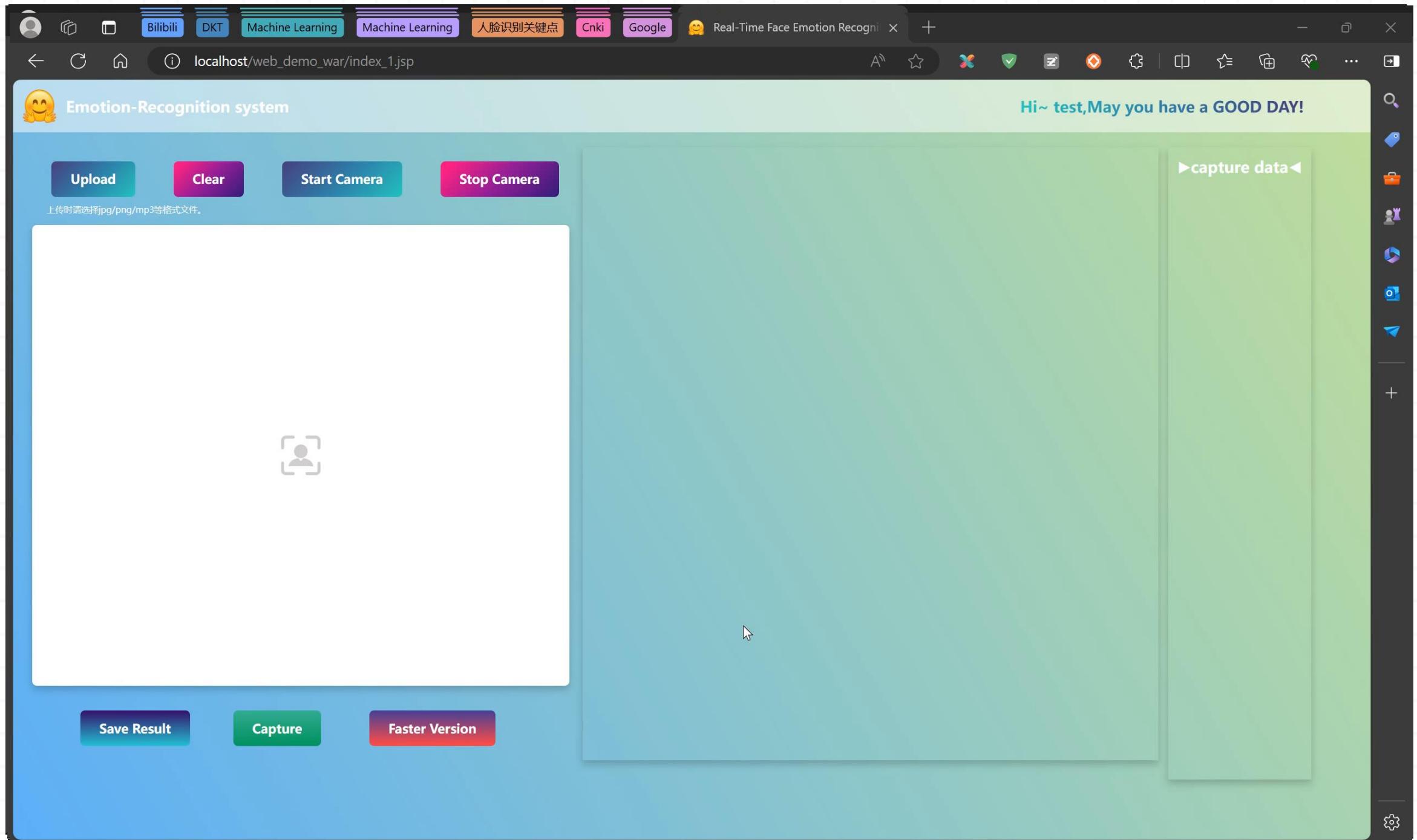
人员情绪分析系统 | 张柏兰 | 周一卓 | 余佳浩 | 赵穗

## ★界面介绍

- `index.jsp` 首页（选择登录、注册）
- `register.jsp` 注册
- `login.jsp` 登录页面——【用户登录JS校验】获取两次输入的密码，判断密码是否一致
- `index_1.jsp` 人脸识别界面
- `index_2_FV.jsp` 人脸识别极速版

# 系统界面展示

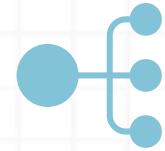
Introduction of



- 注册用户 ✓
- 用户登录 ✓
- 加载模型动画 ✓
- 上传照片进行识别 ✓
- 打开摄像头实时识别 ✓
- Save截屏 ✓
- Capture输出时间+人名  
+情绪 ✓
- 极速版 (更流畅) ✓

# 总结与反思

Summary and Reflection



## 后续改进方向

- 登录时可截取视频流中某一帧（仅含用户本身人脸信息）将人脸信息保存入数据库，方便后续人脸识别使用。
- 目前还是有些卡顿，对界面、响应做进一步的优化（更改为基于Vue框架）。
- 模型性能仍需提高（光线较暗时识别情况较差）。
- 尝试融入在线学习。

### 用户层面

- 面部表情变化数据实时可视化，并存入数据库中（便于分析）。
- 系统目前只能上传图片或实时检测，对于上传视频如何处理有待开发。
- 为保证用户安全性，在注册登录界面可加入人脸识别验证环节。

智慧学习与智慧教育 | 基于face-api.js

# 情绪识别系统

Real-Time Facial Emotion Recognition System



## 小组成员

## 分工

张柏兰

界面与其他功能实现、  
图片情绪识别、ppt、文档

周一卓

人脸识别+情绪实时识别、文档

赵 穗

文档、文献搜索

余佳浩

文档、文献搜索、数据集查找