باسمه تعالی

دانشکده مهندسي مکانیک

پروژهٔ پایانی- تشخیص احساسات صورت

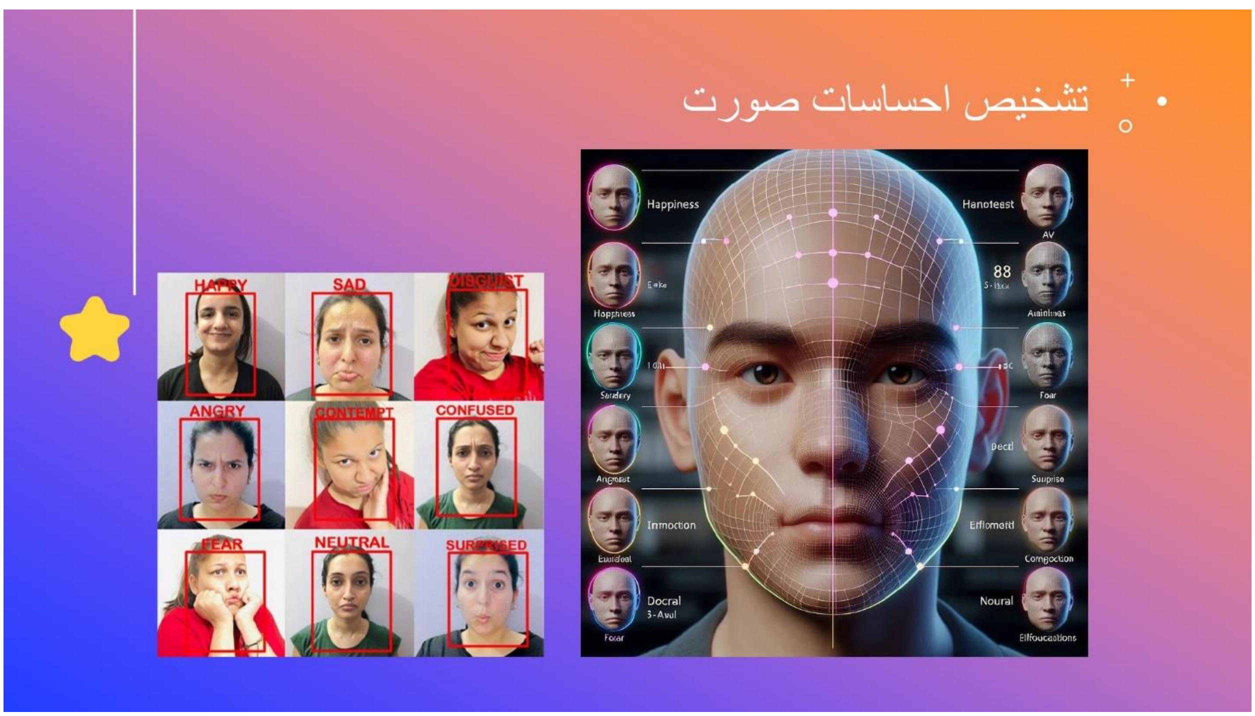
اعضای گروه:
محمّدمهدی انصاری
محمّدمهدی بزاز
محمّدجواد قاضی خانی

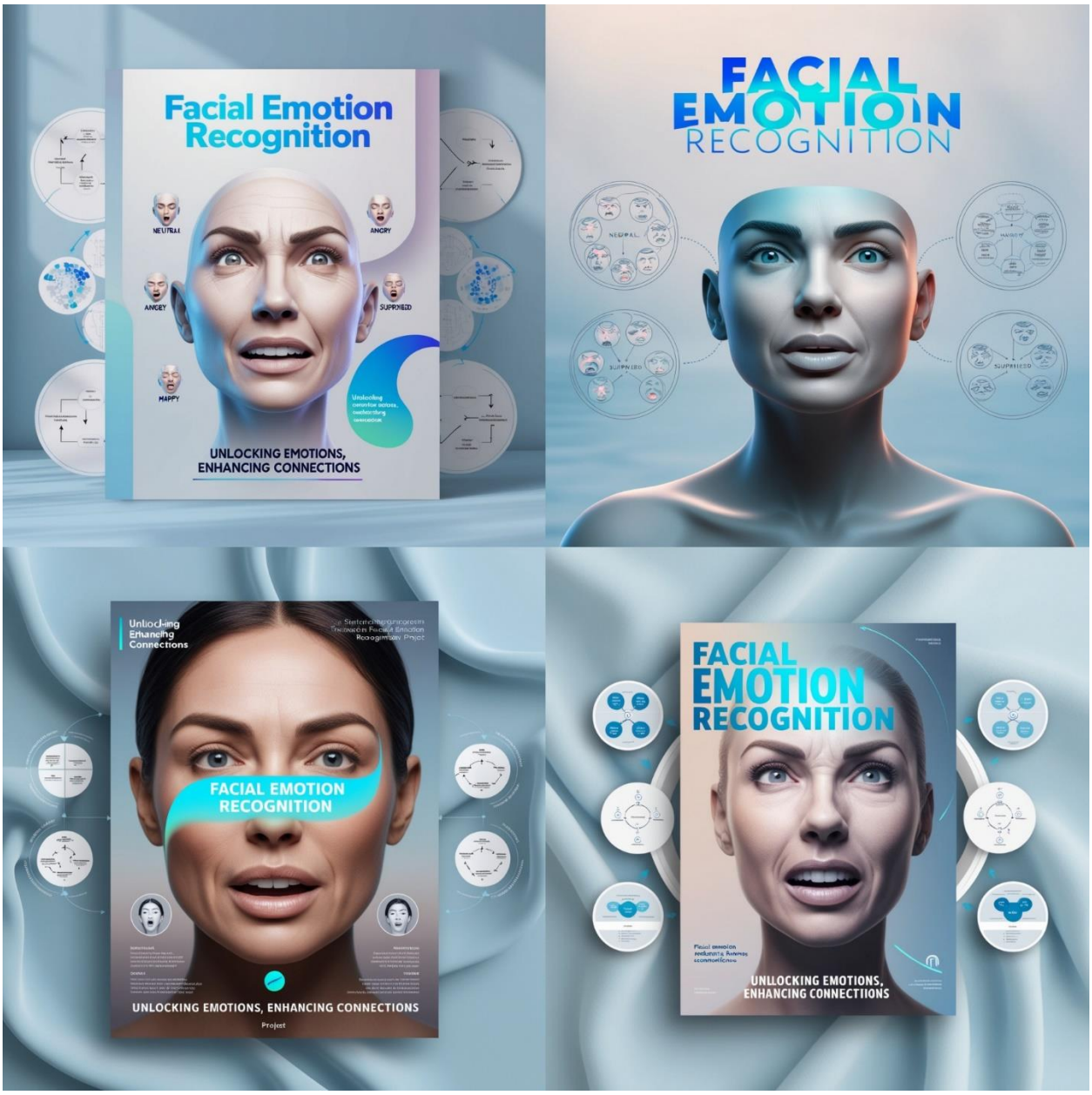
درس:
هوش مصنوعی

استاد:
دکتر سیّد حسن ذبیحی‌فر

پاییز 1403


12
تشخیص احساسات صورت


تشخیص احساسات صورت
HAPPY
SAD
DISGUST
ANGRY
CONTEMPT
CONFUSED
FEAR
NEUTRAL
SURPRISED
Happiness



Generated by Leonardo Ai
Prompt: "Facial Emotion Recognition" Project Poster

فهرست مطالب

مارکت FER

بازار تشخیص احساسات چهره (Facial Emotion Recognition - FER) تحلیل روندها و چشم‌اندازهای منطقه‌ای

بازار جهانی FER در سال 2022 به ارزش 2.9 میلیارد دلار بود و پیش‌بینی می‌شود تا سال 2030 به 7.7 میلیارد دلار برسد (نرخ رشد سالانه مرکب 14.5% از 2024 تا 2030). این بازار تحت تأثیر روندهای منطقه‌ای و تفاوت‌های فرهنگی به‌سرعت در حال رشد است.

تحلیل منطقه‌ای

- **آمریکای شمالی:**
این منطقه، با تمرکز بر ایالات متحده و کانادا، بیشترین سهم بازار جهانی FER (بیش از 40%) را دارد. شرکت‌های بزرگی نظیر مایکروسافت و IBM در صنایع مختلف از جمله بهداشت، خرده‌فروشی و خودروسازی روی توسعه این فناوری سرمایه‌گذاری می‌کنند.
- **اروپا:**
قوانین سخت‌گیرانه حریم خصوصی، از جملهGDPR ، موجب کاهش سرعت پذیرش FER در اروپا شده است. با این حال، پیشرفت در یادگیری ماشین و همکاری‌های شرکتی به رشد بازار در کشورهایی مانند انگلستان، آلمان و فرانسه کمک کرده است.
- **آسیا-اقیانوسیه:**
رشد سریع این بازار به دلیل سرمایه‌گذاری در فناوری‌های هوشمند و تقاضای روزافزون برای اتوماسیون است. چین و هند پیشگام این رشد هستند، اما تفاوت‌های قوانین ملی مانعی برای توسعه یکنواخت در منطقه است.
- **خاورمیانه و آفریقا:**
این بازار در مراحل اولیه خود قرار دارد، اما کشورهایی مانند امارات و آفریقای جنوبی در حال افزایش سرمایه‌گذاری در این زمینه برای بهبود تجربه مشتری هستند.
- **آمریکای لاتین:**
کشورهایی مانند برزیل و آرژانتین به‌آرامی در حال استفاده از FER در خرده‌فروشی و امنیت هستند. چالش‌هایی نظیر بی‌ثباتی اقتصادی می‌تواند روند رشد را کند کند.

تقسیم‌بندی بازار

- **بر اساس نوع:**
  - **آنلاین**
  - **آفلاین**
- **بر اساس کاربرد:**
  - دولت
  - خرده‌فروشی
  - بهداشت
  - سرگرمی
  -

شرکت‌های کلیدی فعال در بازارFER

شامل Microsoft، IBM، Affectiva، Py-Feat، NEC Global، MorphCast و Cameralyze.

روندهای کلیدی و فرصت‌ها

1. **رشد فناوری:**
   - پیشرفت در الگوریتم‌های یادگیری عمیق و بینایی کامپیوتر.
   - ترکیب FER با واقعیت افزوده و مجازی برای ایجاد تجربه‌های شخصی‌سازی‌شده.
2. **کاربردهای صنعتی:**
   - نظارت بیمار و تشخیص سلامت روان در حوزه پزشکی.
   - تحلیل رفتار مشتری و بهبود استراتژی‌های بازاریابی در خرده‌فروشی.
   - تحلیل مخاطبان و ارائه تجربیات تعاملی در صنعت سرگرمی.
3. **چالش‌ها:**
   - نگرانی‌های مربوط به حریم خصوصی و رضایت کاربران.
   - مسائل اخلاقی در استفاده از داده‌های تشخیص چهره.

بازار FER با تقاضای فزاینده در صنایع مختلف و پیشرفت‌های تکنولوژیک، فرصت‌های قابل‌توجهی برای سرمایه‌گذاری و توسعه ارائه می‌دهد.

انواع دیتاست‌ها

29 dataset results for Facial Expression Recognition (FER) ×

AffectNet

AffectNet is a large facial expression dataset with around 0.4 million images manually labeled for the presence of eight (neutral, happy, angry, sad, fear, surprise, disgust, contempt) facial ex...

304 PAPERS • 3 BENCHMARKS

CK+ (Extended Cohn-Kanade dataset)

The Extended Cohn-Kanade (CK+) dataset contains 593 video sequences from a total of 123 different subjects, ranging from 18 to 50 years of age with a variety of genders and heritage....

232 PAPERS • 2 BENCHMARKS

RAF-DB (Real-world Affective Faces)

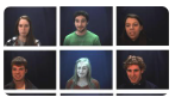
The Real-world Affective Faces Database (RAF-DB) is a dataset for facial expression. It contains 29672 facial images tagged with basic or compound expressions by 40 independent taggers....

160 PAPERS • 3 BENCHMARKS

FER2013 (Facial Expression Recognition 2013 Dataset)

Fer2013 contains approximately 30,000 facial RGB images of different expressions with size restricted to 48×48, and the main labels of it can be divided into 7 types: 0=Angry, 1=Disgust,...

153 PAPERS • 5 BENCHMARKS



DISFA (Denver Intensity of Spontaneous Facial Action)

The Denver Intensity of Spontaneous Facial Action (DISFA) dataset consists of 27 videos of 4844 frames each, with 130,788 images in total. Action unit annotations are on different levels of in-...

145 PAPERS • 3 BENCHMARKS

بالغ بر 30 دیتاست آزاد برای آموزش مدل‌های هوش‌مصنوعی تشخیص احساسات چهره وجود دارد. تفاوت دیتاست‌ها در تعداد کلاس، تعداد عکس در هر کلاس و در مجموع، رنگی بودن یا نبودن، ابعاد پیکسل‌ها، متفاوت بودن الگوی تصاویر ثبت شده و ... می‌باشد.

TABLE I: Dataset comparison

Characteristics	FER2013	RAF-DB	AffectNet-7	ExpW
No. of channels	1	3	3	3
Image size	48 × 48	100 × 100	224 × 224	224 × 224
Total Samples	35,887	15,339	287,401	91,793
No. of classes	7	7	7	7

Dataset	Num. of Classes	Happy	Sad	Angry	Surprise	Neutral	Fear	Disgust	Contempt	Accuracy
Fer2013	7								-	
RAF-DB	7								-	
AffectnNet	8									

TABLE II: Class wise data distribution across four datasets: FER2013, RAF-DB, AffectNet-7 and ExpW

Class	**FER2013**		**RAF-DB**		**AffectNet-7**		**ExpW**	
	Train	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
Angry	3995	491	705	162	24882	500	2569	364
Disgust	436	416	717	160	3803	500	2796	396
Fear	4097	626	281	74	6378	500	761	108
Happy	7215	594	4772	1185	134415	500	21375	3024
Neutral	4965	528	2524	680	74874	500	24418	3454
Sad	4830	879	1982	478	25459	500	7391	1046
Surprise	3171	55	1290	329	14090	500	4943	699
Total	**28709**	**3589**	**12271**	**3068**	**283901**	**3500**	**64253**	**9091**

TABLE 3
Number of Annotated Images in Each Category

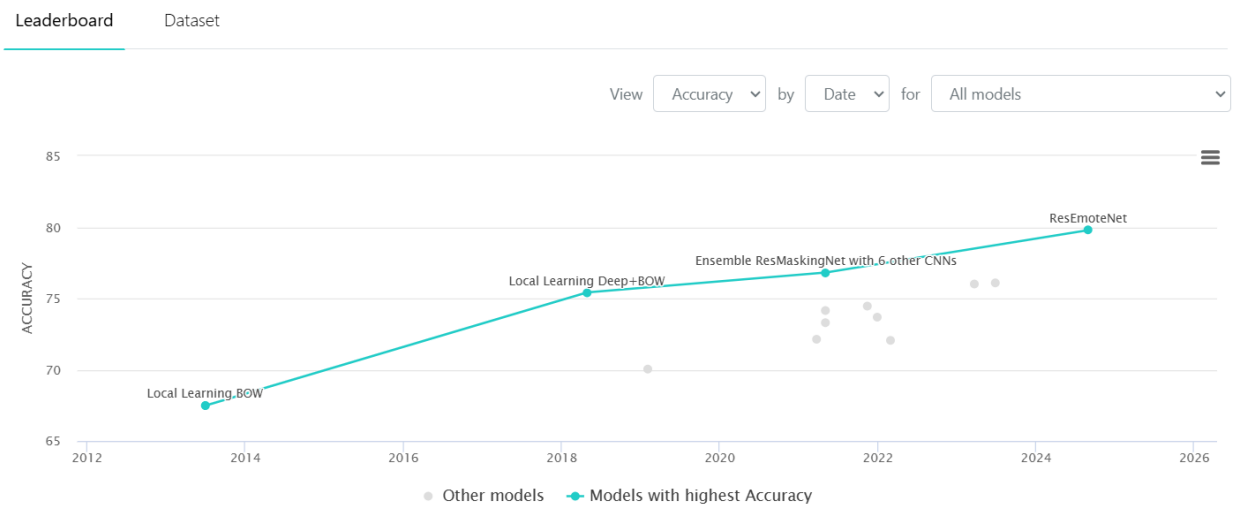
Expression	**Number**
Neutral	80,276
Happy	146,198
Sad	29,487
Surprise	16,288
Fear	8,191
Disgust	5,264
Anger	28,130
Contempt	5,135
None	35,322
Uncertain	13,163
Non-Face	88,895

AffectNet

TABLE III: Test Accuracy (%) comparison of ResEmoteNet with existing state-of-the-art methods across four datasets: FER2013, RAF-DB, AffectNet-7 and ExpW.

Method	Accuracy in %			
	FER2013	**RAF-DB**	**AffectNet-7**	**ExpW**
Seg. VGG-19 [29]	75.97	-	-	-
EmoNeXt [30]	76.12	-	-	-
En. ResMaskingNet [13]	76.82	-	-	-
SEResNet [31]	-	83.37	56.54	-
Arm [32]	-	90.42	62.5	-
APVit [28]	-	91.98	66.91	73.48
ARBEx [33]	-	92.47	-	-
S2D [12]	-	92.57	67.62	-
C MT EmoAffectNet [14]	-	-	69.4	-
AGLRLS [34]	-	-	-	73
SchiNet [35]	-	-	-	73.10
Proposed ResEmoteNet	**79.79**	**94.76**	**72.93**	**75.67**

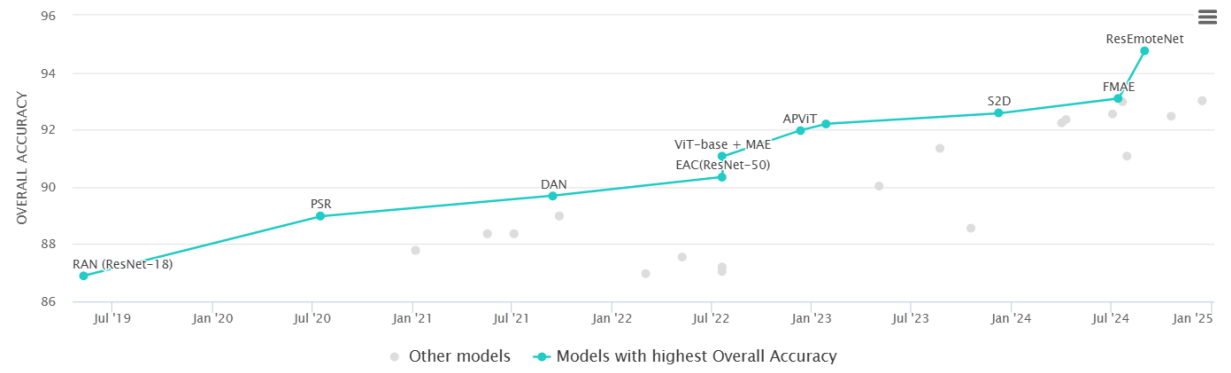
Facial Expression Recognition (FER) on FER2013



Facial Expression Recognition (FER) on RAF-DB

Leaderboard Dataset

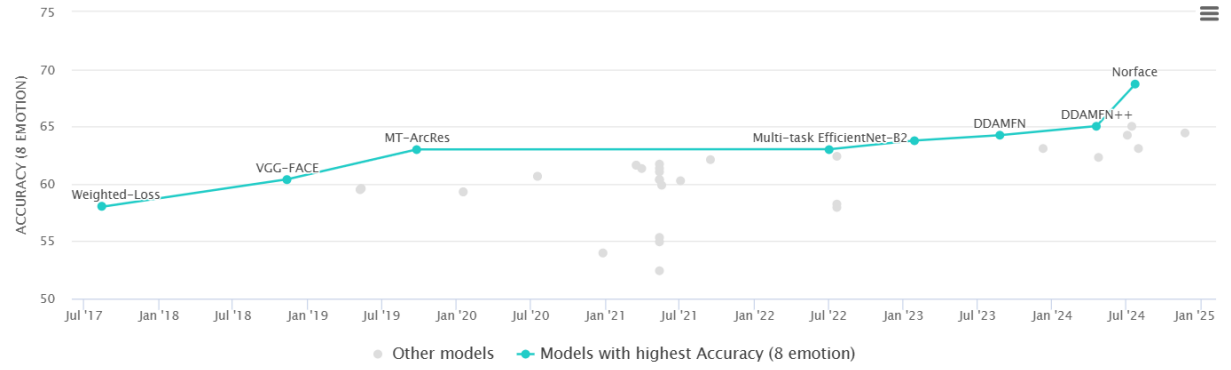
View Overall Accuracy by Date for All models



Facial Expression Recognition (FER) on AffectNet

Leaderboard Dataset

View Accuracy (8 emotion) by Date for All models



کتابخانهٔ دیپ‌فیس

DeepFace یک کتابخانه منبع‌باز در پایتون است که برای پردازش و تحلیل چهره طراحی شده است. این کتابخانه از الگوریتم‌های یادگیری عمیق برای انجام وظایف مختلف مرتبط با چهره استفاده می‌کند. DeepFace یک رابط کاربری ساده فراهم می‌کند که امکان استفاده از مدل‌های از پیش آموزش‌دیده را برای تشخیص و تحلیل چهره فراهم می‌سازد.

ویژگی‌ها و قابلیت‌ها:

1. **تشخیص هویت چهره (Face Recognition)**
مقایسه دو تصویر چهره برای بررسی شباهت و تأیید هویت.
2. **تشخیص احساسات (Emotion Analysis)**
شناسایی حالات احساسی نظیر شادی، عصبانیت، غم و ترس.
3. **تشخیص سن و جنسیت (Age & Gender Detection)**
تخمین سن و شناسایی جنسیت افراد.
4. **تشخیص نژاد (Race Detection)**
تخمین قومیت یا نژاد افراد با استفاده از تصویر چهره. نژادهای شناسایی‌شده توسط این کتابخانه شامل:
    - آسیایی(Asian)
    - اروپایی یا سفیدپوست(White)
    - آفریقایی یا سیاه‌پوست(Black)
    - هندی(Indian)
    - خاورمیانه‌ای یا لاتین(Middle Eastern/Latino)

    این قابلیت می‌تواند در تحلیل داده‌های جمعیتی، بازاریابی هدفمند، یا تحقیق‌های علمی مورد استفاده قرار گیرد، هرچند باید با دقت به مسائل اخلاقی و حفظ حریم خصوصی انجام شود.

5. **پشتیبانی از چندین مدل یادگیری عمیق:**
این کتابخانه از مدل‌های معروف مانندVGG-Face ، Google FaceNet، OpenFace، DeepID و Dlib پشتیبانی می‌کند.
6. **استفاده آسان:**
تنها با چند خط کد می‌توان وظایف پیچیده مرتبط با تحلیل چهره را انجام داد.
7. **انعطاف‌پذیری:**
امکان استفاده از مدل‌های پیش‌فرض یا افزودن مدل‌های دلخواه برای بهبود عملکرد وجود دارد.

مزایا:

- ساده بودن استفاده حتی برای کاربران مبتدی.
- عدم نیاز به GPU برای اجرای وظایف پایه.
- قابل استفاده در پروژه‌های امنیتی، تحلیل ویدئو، روان‌شناسی و بازاریابی.

DeepFace ابزاری قدرتمند برای پروژه‌های مرتبط با تحلیل چهره است که با ارائه ویژگی‌های متنوع و انعطاف‌پذیری بالا، امکان کاربرد در حوزه‌های مختلف را فراهم می‌کند.

شبکه‌های اجرا شده روی دیتاست‌ها

Dataset	Model	Acc.	Val. Acc.	epoch	epoch time	Total time
FER2013	SimpleDeepCNN	72.09	65.43	45	350s	4.5h
RAF-DB	SimpleCNN	79	73	50	75s	1h
RAF-DB	SimpleCNN	84.38	76.04	100	50s	1.5h
RAF-DB	**ResEmoteNet**	**88.25**	**78.56**	**45**	**200s**	**2.5h**
AffectNet	ResEmoteNet	66.05	65.48	45	550s	7h
AffectNet	ResNet18	-	68.77	10	150s	0.5h

تست‌ها با مدل 88 درصدی روی دیتاست RAF-DB انجام می‌شوند.

نمایش معماری شبکه های عصبی

FER2013	SimpleDeepCNN	72.09	65.43	45	350s	4.5h

```
model= tf.keras.models.Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), padding='same', activation='relu',
input_shape=(48, 48,1)))
model.add(Conv2D(64,(3,3), padding='same', activation='relu' ))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128,(5,5), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(512,(3,3), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(512,(3,3), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(256,activation = 'relu'))
model.add(BatchNormalization())
model.add(Dropout(0.25))

model.add(Dense(512,activation = 'relu'))
model.add(BatchNormalization())
model.add(Dropout(0.25))

model.add(Dense(7, activation='softmax'))
```

RAF-DB	SimpleCNN	79	73	50	75s	1h

```
model=Sequential()
model.add( Conv2D(64,(3,3),input_shape=X.shape[1:],activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add( Conv2D(64,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add( Conv2D(32,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))


model.add(Flatten())

model.add(Dense(7,activation='softmax'))
```

RAF-DB	SimpleCNN	84.38	76.04	100	50s	1.5h

```
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(70, 70, 1)),
    MaxPooling2D((2, 2)),
    Dropout(0.2),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Dropout(0.3),
    Conv2D(128, (3, 3), activation='relu'),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.4),
    Dense(num_classes, activation='softmax')
])
```

RAF-DB	ResEmoteNet	88.25	78.56	45	200s	2.5h
AffectNet	ResEmoteNet	66.05	65.48	45	550s	7h

```
# SEBlock Implementation
def se_block(input_tensor, reduction=16):
    channels = input_tensor.shape[-1]
    x = GlobalAveragePooling2D()(input_tensor)
    x = Dense(channels // reduction, activation='relu')(x)
    x = Dense(channels, activation='sigmoid')(x)
    x = Reshape((1, 1, channels))(x)
    return multiply([input_tensor, x])

# Residual Block Implementation
def residual_block(input_tensor, filters, stride=1):
    shortcut = input_tensor

    x = Conv2D(filters, (3, 3), strides=stride, padding='same')(input_tensor)
    x = BatchNormalization()(x)
    x = ReLU()(x)
    x = Conv2D(filters, (3, 3), strides=1, padding='same')(x)
    x = BatchNormalization()(x)

    if stride != 1 or input_tensor.shape[-1] != filters:
        shortcut = Conv2D(filters, (1, 1), strides=stride,
padding='same')(input_tensor)
        shortcut = BatchNormalization()(shortcut)

    x = Add()([x, shortcut])
    x = ReLU()(x)
    return x

# Modified Model
def build_model(input_shape=(48, 48, 1), num_classes=7):
    inputs = Input(shape=input_shape)

    # Initial Conv Blocks
    x = Conv2D(32, (3, 3), padding='same', activation='relu')(inputs)
    x = Conv2D(64, (3, 3), padding='same', activation='relu')(x)
    x = BatchNormalization()(x)
    x = MaxPooling2D(pool_size=(2, 2))(x)
    x = Dropout(0.25)(x)

    x = Conv2D(128, (5, 5), padding='same', activation='relu')(x)
    x = BatchNormalization()(x)
```

```
    x = MaxPooling2D(pool_size=(2, 2))(x)
    x = Dropout(0.25)(x)

    x = Conv2D(512, (3, 3), padding='same', activation='relu')(x)
    x = BatchNormalization()(x)
    x = MaxPooling2D(pool_size=(2, 2))(x)
    x = Dropout(0.25)(x)

    # Add SEBlock
    x = se_block(x)

    # Residual Blocks
    x = residual_block(x, 512, stride=1)
    x = residual_block(x, 512, stride=2)

    # Flatten and Fully Connected Layers
    x = Flatten()(x)
    x = Dense(256, activation='relu')(x)
    x = BatchNormalization()(x)
    x = Dropout(0.25)(x)

    x = Dense(512, activation='relu')(x)
    x = BatchNormalization()(x)
    x = Dropout(0.25)(x)

    outputs = Dense(num_classes, activation='softmax')(x)

    model = Model(inputs, outputs)
    return model
```

AffectNet	ResNet18	-	68.77	10	150s	0.5h

```python
class Model(nn.Module):
    def __init__(self, num_classes: int):
        super().__init__()
        self.resnet = models.resnet18(weights=ResNet18_Weights.DEFAULT)
        self.resnet.fc = nn.Linear(self.resnet.fc.in_features, num_classes)
        self.softmax = nn.Softmax(dim=-1)

    def forward(self, x):
        return self.softmax(self.resnet(x))
```

مقایسه بهترین دقّت مدل ها با مقالات

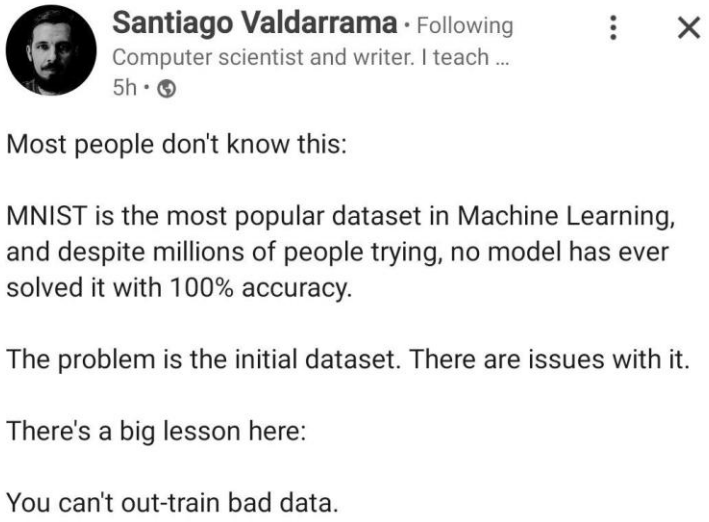
Dataset	Model	Acc.	Val. Acc.	epoch	epoch time	Total time
FER2013	SimpleDeepCNN	72.09	65.43	45	350s	4.5h
RAF-DB	**ResEmoteNet**	**88.25**	**78.56**	**45**	**200s**	**2.5h**
AffectNet	ResNet18	-	68.77	10	150s	0.5h

با مقایسهٔ مقادیر بدست آورده شده توسّط گروه ما برای دقّت روی دیتاست‌های مختلف و دقّت‌های بدست‌آمده در مقالات، می‌توان نتیجه گرفت، مدل دیتاست افکت‌نت در سطح بهترین مقالات است؛ همچنین مدل دیتاست رف-دی‌بی با درنظرگرفتن اینکه مقالات از روش‌های ترکیبی و پیچیده استفاده کرده‌اند، مدل ما از دقّت مورد قبولی برخوردار است؛ مدل fer2013 نیز نسبت به درصدهای مقالات مورد قبول است.

مشکلات و راه‌های افزایش دقّت

مشکلاتی که شامل بحث FER در حوزهٔ دیتاست‌ها می‌شود، شامل این موارد می‌شود: در بعضی موارد دیتاست کوچک و دقت بالایی را نتیجه می‌دهد، امّا تعمیم‌پذیری پایینی دارد؛ در بعضی موارد دیتاست بزرگ بوده و موجب کاهش دقّت می‌شود. در دیتاست‌ها بعضاً یک یا چندکلاس از تعداد عکس یا دیتای کافی برخوردار نیستند و این موضوع باعث آموزش ضعیف برای حدس آن کلاس می‌شود؛ و یا افزایش تعداد کلاس‌ها که از دقّت می‌کاهد.

مورد دیگری که شامل هر دیتاستی در هر موضوعی می‌شود، نوع تصاویر و دیتاست است که گاها در هر دیتاست شامل الگویی خاص بوده و مانع تعمیم‌پذیری برای حالت‌های واقعی می‌شود. نظر شما را به توییت زیر جلب می‌کنم:



همین بحث در مورد دیتاست‌های FER نیز صادق است؛ نمونهٔ بارز آن دیتاست affectnet می‌باشد.

راهکارها: کاهش تعداد کلاس‌ها، انجام تغییر در تعداد داده‌های دیتاست، استفاده از GANها برای تولید دیتاست برای کلاس‌ها با دیتای کم.

نمونه کد

از مدل 88درصدی رف-دی‌بی برای توضیح نحوه کدنویسی استفاده می‌شود؛ مابقی با اندکی تغییر، یکسان‌اند. (به غیر از مدل 68 درصدی افکت‌نت که روی سرور با GPU ی بالا، پردازش شده است؛ پایتورچ)

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
import keras
from keras.preprocessing import image
from keras.models import Sequential
from keras.layers import Conv2D, MaxPool2D,
Flatten,Dense,Dropout,BatchNormalization
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import cv2
from tensorflow.keras.applications import VGG16, InceptionResNetV2
from keras import regularizers
from tensorflow.keras.optimizers import Adam,RMSprop,SGD,Adamax
```

فراخوانی کتابخانه‌ها

```
train_dir = r"D:\University\0 دانشگاه علم و صنعت ایران\7 نیمسال های تحصیلی\6 ترم هفت هوش
مصنوعی\Project\Datasets\4 RAF-DB\archive\DATASET\train"
test_dir = r"D:\University\0 دانشگاه علم و صنعت ایران\7 نیمسال های تحصیلی\6 ترم هفت هوش
مصنوعی\Project\Datasets\4 RAF-DB\archive\DATASET\test"
```

مسیر فایل‌های تست و ترین

```
import tensorflow as tf
from tensorflow.keras.layers import (Conv2D, MaxPooling2D, Flatten, Dense,
Dropout,
                                     BatchNormalization, GlobalAveragePooling2D,
Reshape, multiply, ReLU, Add, Input)
from tensorflow.keras.models import Sequential, Model

# SEBlock Implementation
def se_block(input_tensor, reduction=16):
    channels = input_tensor.shape[-1]
    x = GlobalAveragePooling2D()(input_tensor)
    x = Dense(channels // reduction, activation='relu')(x)
    x = Dense(channels, activation='sigmoid')(x)
    x = Reshape((1, 1, channels))(x)
```

```
    return multiply([input_tensor, x])

# Residual Block Implementation
def residual_block(input_tensor, filters, stride=1):
    shortcut = input_tensor

    x = Conv2D(filters, (3, 3), strides=stride, padding='same')(input_tensor)
    x = BatchNormalization()(x)
    x = ReLU()(x)
    x = Conv2D(filters, (3, 3), strides=1, padding='same')(x)
    x = BatchNormalization()(x)

    if stride != 1 or input_tensor.shape[-1] != filters:
        shortcut = Conv2D(filters, (1, 1), strides=stride,
padding='same')(input_tensor)
        shortcut = BatchNormalization()(shortcut)

    x = Add()([x, shortcut])
    x = ReLU()(x)
    return x

# Modified Model
def build_model(input_shape=(48, 48, 1), num_classes=7):
    inputs = Input(shape=input_shape)

    # Initial Conv Blocks
    x = Conv2D(32, (3, 3), padding='same', activation='relu')(inputs)
    x = Conv2D(64, (3, 3), padding='same', activation='relu')(x)
    x = BatchNormalization()(x)
    x = MaxPooling2D(pool_size=(2, 2))(x)
    x = Dropout(0.25)(x)

    x = Conv2D(128, (5, 5), padding='same', activation='relu')(x)
    x = BatchNormalization()(x)
    x = MaxPooling2D(pool_size=(2, 2))(x)
    x = Dropout(0.25)(x)

    x = Conv2D(512, (3, 3), padding='same', activation='relu')(x)
    x = BatchNormalization()(x)
    x = MaxPooling2D(pool_size=(2, 2))(x)
    x = Dropout(0.25)(x)

    # Add SEBlock
    x = se_block(x)
```

```
    # Residual Blocks
    x = residual_block(x, 512, stride=1)
    x = residual_block(x, 512, stride=2)

    # Flatten and Fully Connected Layers
    x = Flatten()(x)
    x = Dense(256, activation='relu')(x)
    x = BatchNormalization()(x)
    x = Dropout(0.25)(x)

    x = Dense(512, activation='relu')(x)
    x = BatchNormalization()(x)
    x = Dropout(0.25)(x)

    outputs = Dense(num_classes, activation='softmax')(x)

    model = Model(inputs, outputs)
    return model

# Build and Compile Model
model = build_model()
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Model Summary
model.summary()
```

تعریف مدل

```
img_size=48
```

```
train_datagen = ImageDataGenerator(width_shift_range = 0.1,
                                   height_shift_range = 0.1,
                                   horizontal_flip = True,
                                   rescale = 1./255,
                                   validation_split = 0.2
                                  )
validation_datagen = ImageDataGenerator(rescale = 1./255,
                                   validation_split = 0.2)
```

در نظر گرفتن شیفت خوردن به جهات متفاوت برای دیتا و نرمالایز کردن اعداد پیکسل‌ها

```
train_generator = train_datagen.flow_from_directory(directory = train_dir,
                                                    target_size =
(img_size,img_size),
                                                    batch_size = 64,
                                                    color_mode = "grayscale",
                                                    class_mode = "categorical",
                                                    subset = "training"
                                                   )
validation_generator = validation_datagen.flow_from_directory( directory =
test_dir,
                                                               target_size =
(img_size,img_size),
                                                               batch_size = 64,
                                                               color_mode =
"grayscale",
                                                               class_mode =
"categorical",
                                                               subset =
"validation"
                                                              )
```

تعیین ابعاد و رنگی بودن یا نبودن و سایرموارد

```
model.compile(
    optimizer = Adam(learning_rate=0.001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
  )
```

کامپایل کردن مدل

```
epochs = 45
batch_size = 64
```

تعیین مقدار ایپاک و بچ سایز

```
history = model.fit(x = train_generator,epochs = epochs,validation_data =
validation_generator)
```

فیت کردن مدل

```
Epoch 1/45
154/154 ━━━━━━━━━━━━━━━━━━━━ 232s 1s/step - accuracy: 0.2608 - loss: 2.2288 - val_accuracy: 0.3879 - val_loss: 1.6919
Epoch 2/45
154/154 ━━━━━━━━━━━━━━━━━━━━ 205s 1s/step - accuracy: 0.3979 - loss: 1.7144 - val_accuracy: 0.4190 - val_loss: 1.5887
Epoch 3/45
154/154 ━━━━━━━━━━━━━━━━━━━━ 203s 1s/step - accuracy: 0.4752 - loss: 1.4746 - val_accuracy: 0.4812 - val_loss: 1.4680
Epoch 4/45
154/154 ━━━━━━━━━━━━━━━━━━━━ 203s 1s/step - accuracy: 0.5338 - loss: 1.3145 - val_accuracy: 0.5221 - val_loss: 1.4247
Epoch 5/45
154/154 ━━━━━━━━━━━━━━━━━━━━ 202s 1s/step - accuracy: 0.5953 - loss: 1.1533 - val_accuracy: 0.6121 - val_loss: 1.0739
Epoch 6/45
154/154 ━━━━━━━━━━━━━━━━━━━━ 202s 1s/step - accuracy: 0.6266 - loss: 1.0398 - val_accuracy: 0.6498 - val_loss: 1.0110
Epoch 7/45
154/154 ━━━━━━━━━━━━━━━━━━━━ 202s 1s/step - accuracy: 0.6526 - loss: 0.9532 - val_accuracy: 0.6792 - val_loss: 0.9526
Epoch 8/45
154/154 ━━━━━━━━━━━━━━━━━━━━ 203s 1s/step - accuracy: 0.6767 - loss: 0.8857 - val_accuracy: 0.5319 - val_loss: 1.2689
Epoch 9/45
154/154 ━━━━━━━━━━━━━━━━━━━━ 203s 1s/step - accuracy: 0.6889 - loss: 0.8623 - val_accuracy: 0.7218 - val_loss: 0.8273
Epoch 10/45
154/154 ━━━━━━━━━━━━━━━━━━━━ 204s 1s/step - accuracy: 0.7028 - loss: 0.8301 - val_accuracy: 0.6678 - val_loss: 0.9253
Epoch 11/45
154/154 ━━━━━━━━━━━━━━━━━━━━ 205s 1s/step - accuracy: 0.7206 - loss: 0.7783 - val_accuracy: 0.7545 - val_loss: 0.7810
Epoch 12/45
154/154 ━━━━━━━━━━━━━━━━━━━━ 204s 1s/step - accuracy: 0.7253 - loss: 0.7752 - val_accuracy: 0.6678 - val_loss: 0.9068
Epoch 13/45
...
Epoch 44/45
154/154 ━━━━━━━━━━━━━━━━━━━━ 203s 1s/step - accuracy: 0.8779 - loss: 0.3474 - val_accuracy: 0.7643 - val_loss: 0.8706
Epoch 45/45
154/154 ━━━━━━━━━━━━━━━━━━━━ 204s 1s/step - accuracy: 0.8825 - loss: 0.3303 - val_accuracy: 0.7856 - val_loss: 0.7312
```

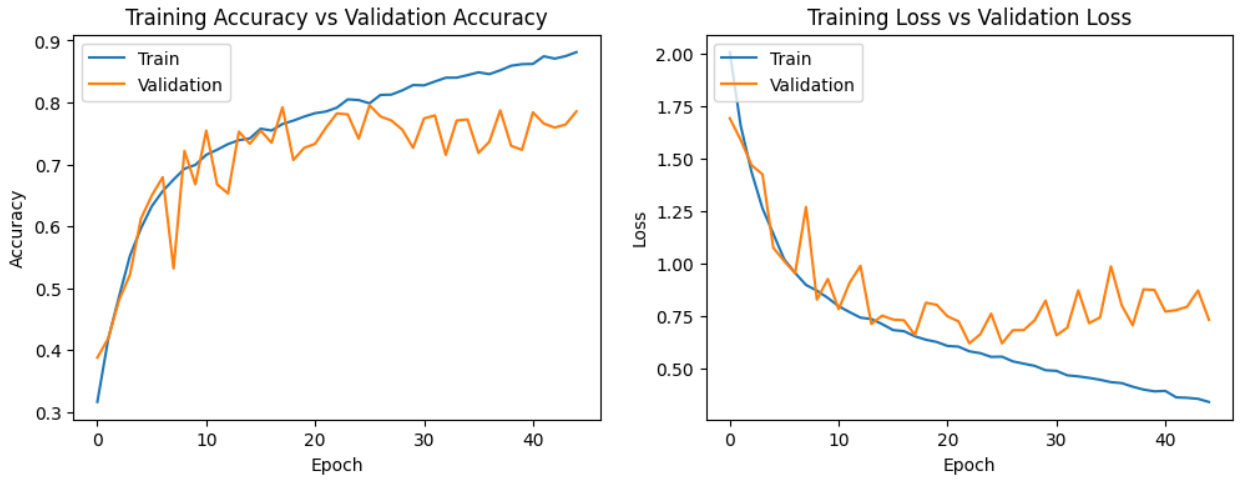
```python
fig , ax = plt.subplots(1,2)
train_acc = history.history['accuracy']
train_loss = history.history['loss']
fig.set_size_inches(12,4)

ax[0].plot(history.history['accuracy'])
ax[0].plot(history.history['val_accuracy'])
ax[0].set_title('Training Accuracy vs Validation Accuracy')
ax[0].set_ylabel('Accuracy')
ax[0].set_xlabel('Epoch')
ax[0].legend(['Train', 'Validation'], loc='upper left')

ax[1].plot(history.history['loss'])
ax[1].plot(history.history['val_loss'])
ax[1].set_title('Training Loss vs Validation Loss')
ax[1].set_ylabel('Loss')
ax[1].set_xlabel('Epoch')
ax[1].legend(['Train', 'Validation'], loc='upper left')

plt.show()
```

رسم نمودارهای دقت و ارور



تست های گرفته شده

برای انجام تست‌ها از مدل با دقّت بالاتر (مدل 88 درصدی ResEmoteNet آموزش دیده شده روی دیتاست RAF-DB) استفاده می‌کنیم؛ بدین منظور فایلی مجزا برای قراردادن نتایج تست منظور شده است که در کنار فایل گزارش و ارائه، موجود می‌باشد. برای انجام تست از عکس دکتر ذبیحی‌فر، عکس‌های ساخته شده توسط هوش مصنوعی لئوناردو و کوپایلت، و عکس‌های دیتاست افکت‌نت که مدل روی آن‌ها آموزش داده نشده است، استفاده می‌کنیم.

روش‌های تست

برای انجام تست‌ها می‌توان عکس ورودی داد، از فریم‌های ویدئو استفاده کرد، از وبکم خروجی گرفت و یا از برنامهٔ IP WebCam و اتّصال به دوربین تلفن‌همراه؛ در ادامه کدهایی که این روش‌ها را پیاده سازی می‌کنند، آورده می‌شوند.

عکس جدید

```
import numpy as np
import cv2
from keras.models import load_model
```

فراخوانی کتابخانه‌ها

```
# Load the saved model
model = load_model('model_rafdb.h5')
```

فراخواندن مدل

```
# Emotion labels for FER 2013 dataset
emotion_labels = ['Surprise', 'Fear', 'Disgust', 'Happy', 'Sad', 'Angry',
'Neutral']
```

کلاس‌ها

```
face_detection = cv2.CascadeClassifier('haar_cascade_face_detection.xml')

settings = {
    'scaleFactor': 1.3,
    'minNeighbors': 5,
    'minSize': (50, 50)
}
```

موارد لازم برای دیتکت شدن صورت توسط opencv

```
def preprocess_image(image_path, face_detection, settings):
    # Load image in color
    img = cv2.imread(image_path)
    if img is None:
        raise ValueError("Image not found or unable to load.")

    # Convert image to grayscale
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Detect faces in the image
    detected_faces = face_detection.detectMultiScale(gray, **settings)

    if len(detected_faces) == 0:
        raise ValueError("No faces detected in the image.")

    # Process only the first detected face (optional: handle multiple faces)
    x, y, w, h = detected_faces[0]

    # Crop the detected face
    face = gray[y + 5:y + h - 5, x + 20:x + w - 20]
```

```
    # Resize face to 48x48
    face = cv2.resize(face, (48, 48))
    # Normalize pixel values
    face = face / 255.0
    # Expand dimensions to match model input shape (1, 48, 48, 1)
    face = np.expand_dims(face, axis=-1)  # Add channel dimension
    face = np.expand_dims(face, axis=0)  # Add batch dimension

    return face
```

تابع پری پراسس تصویر ورودی و تشخیص صورت و برش آن

```
def predict_emotion(image_path):
    try:
        # Preprocess the image
        processed_img = preprocess_image(image_path, face_detection, settings)

        # Validate processed image shape
        if processed_img is None or len(processed_img.shape) != 4:
            raise ValueError(f"Invalid processed image shape:
{processed_img.shape if processed_img is not None else 'None'}")

        # Debugging shapes
        print(f"Processed image shape: {processed_img.shape}")
        print(f"Expected model input shape: {model.input_shape}")

        # Get prediction
        prediction = model.predict(processed_img)

        # Debugging prediction output
        print(f"Prediction output: {prediction}")

        # Get the index of the highest probability
        emotion_idx = np.argmax(prediction)
        # Get the corresponding label
        emotion = emotion_labels[emotion_idx]

        return emotion
    except Exception as e:
        print(f"Error during emotion prediction: {e}")
        return None
```

تابع تشخیص حالت صورت

```
def predict_emotion_with_probabilities(image_path):
    try:
        # Preprocess the image
        processed_img = preprocess_image(image_path, face_detection, settings)

        if processed_img is None or len(processed_img.shape) != 4:
            raise ValueError(f"Invalid input shape: {processed_img.shape if
processed_img is not None else 'None'}")

        # Print input shape for debugging
        print(f"Processed image shape: {processed_img.shape}")
        print(f"Model expected input shape: {model.input_shape}")

        # Get prediction probabilities for all classes
        predictions = model.predict(processed_img)[0]  # [0] to flatten the batch
dimension

        # Get the index of the highest probability
        emotion_idx = np.argmax(predictions)
        # Get the corresponding label
        emotion = emotion_labels[emotion_idx]
        # Display probabilities for each class
        probabilities = {emotion_labels[i]: predictions[i] for i in
range(len(emotion_labels))}
        return emotion, probabilities
    except Exception as e:
        print(f"Error during prediction: {e}")
        return None, None
```

تابع تشخیص حالت صورت به علاوه درصد حالات متفاوت

```
image_path = 'test.jpg'
predicted_emotion, class_probabilities =
predict_emotion_with_probabilities(image_path)

if predicted_emotion:
    print(f"The predicted emotion is: {predicted_emotion}")
    print("Class probabilities:")
    for emotion, prob in class_probabilities.items():
        print(f"{emotion}: {prob:.4f}")
else:
    print("Prediction failed.")
```

آدرس دهی تصویر جدید و استفاده از تابع "تشخیص حالت صورت به علاوه درصد حالات متفاوت"

```
# Example usage
image_path = 'fearful (4).jfif'
predicted_emotion = predict_emotion(image_path)

if predicted_emotion:
    print(f"The predicted emotion is: {predicted_emotion}")
else:
    print("Failed to predict emotion.")
```

آدرس دهی تصویر جدید و استفاده از تابع "تشخیص حالت صورت"

ویدئو

```
cap = cv2.VideoCapture(r"C:\Users\A\Downloads\ope4vtu0ln.mp4")
```

آدرس دهی ویدئو

```
emotion_dict = ['Surprise', 'Fear', 'Disgust', 'Happy', 'Sad', 'Angry',
'Neutral']

while True:
    # Find haar cascade to draw bounding box around face
    ret, frame = cap.read()
    frame = cv2.resize(frame, (1280, 720))
    if not ret:
        break
    face_detector = cv2.CascadeClassifier('haar_cascade_face_detection.xml')
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # detect faces available on camera
    num_faces = face_detector.detectMultiScale(gray_frame, scaleFactor=1.3,
minNeighbors=5)

    # take each face available on the camera and Preprocess it
    for (x, y, w, h) in num_faces:
        cv2.rectangle(frame, (x, y-50), (x+w, y+h+10), (0, 255, 0), 4)
        roi_gray_frame = gray_frame[y:y + h, x:x + w]
        cropped_img = np.expand_dims(np.expand_dims(cv2.resize(roi_gray_frame,
(48, 48)), -1), 0)

        # predict the emotions
        emotion_prediction = model.predict(cropped_img)
        maxindex = int(np.argmax(emotion_prediction))
        cv2.putText(frame, emotion_dict[maxindex], (x+5, y-20),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2, cv2.LINE_AA)

    cv2.imshow('Emotion Detection', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

تعیین حالات صورت‌های موجود در هر فریم از فیلم

وبكم لبتاب

```
face_detection = cv2.CascadeClassifier('haar_cascade_face_detection.xml')
```

```
camera = cv2.VideoCapture(0)
camera.set(cv2.CAP_PROP_FRAME_WIDTH, 1024)
camera.set(cv2.CAP_PROP_FRAME_HEIGHT, 768)
settings = {
    'scaleFactor': 1.3,
    'minNeighbors': 5,
    'minSize': (50, 50)
}
```

```
emotion_labels = ['Surprise', 'Fear', 'Disgust', 'Happy', 'Sad', 'Angry',
'Neutral']  # Adjust according to your model

while True:
    ret, img = camera.read()
    if not ret:
        break

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    detected = face_detection.detectMultiScale(gray, **settings)

    for x, y, w, h in detected:
        cv2.rectangle(img, (x, y), (x + w, y + h), (245, 135, 66), 2)
        cv2.rectangle(img, (x, y), (x + w // 3, y + 20), (245, 135, 66), -1)
        face = gray[y + 5:y + h - 5, x + 20:x + w - 20]
        face = cv2.resize(face, (48, 48))
        face = face / 255.0

        predictions = model.predict(np.array([face.reshape((48, 48,
1))])).argmax()

        if predictions < len(emotion_labels):
            state = emotion_labels[predictions]
        else:
            state = "Unknown"

        font = cv2.FONT_HERSHEY_SIMPLEX
        cv2.putText(img, state, (x + 10, y + 15), font, 0.5, (255, 255, 255), 2,
cv2.LINE_AA)

    cv2.imshow('Facial Expression', img)
```

```
    if cv2.waitKey(5) != -1:
        break

camera.release()
cv2.destroyAllWindows()
```

انجام حدس روی خروجی وبکم به کد

استفاده از مدل‌ها در IP WebCam

```
# Load the DNN face detection model
face_net = cv2.dnn.readNetFromCaffe('deploy.prototxt',
'res10_300x300_ssd_iter_140000.caffemodel')
```

```
# IP webcam address
ip_camera_url = 'http://192.168.120.145:8080/video'
cap = cv2.VideoCapture(ip_camera_url)
```

```
frame_skip = 10  # Process every 10th frame
frame_count = 0

while True:
    ret, frame = cap.read()
    if not ret:
        print("Error receiving frame from IP camera")
        break

    frame_count += 1
    if frame_count % frame_skip != 0:
        continue

    # Resize the frame for faster processing
    frame = cv2.resize(frame, (320, 240))
    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    # Use DNN model for face detection
    h, w = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 1.0, (300, 300), (104.0, 177.0, 123.0),
swapRB=False)
    face_net.setInput(blob)
    detections = face_net.forward()

    for i in range(detections.shape[2]):
        confidence = detections[0, 0, i, 2]
        if confidence > 0.6:  # Minimum confidence for detection
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (x, y, x1, y1) = box.astype("int")

            # Crop the face from the frame
            face_image = frame_rgb[y:y1, x:x1]

            # Ensure the face is correctly extracted
            if face_image.shape[0] == 0 or face_image.shape[1] == 0:
```

```
                continue

            # 3-second delay before processing the face
            # time.sleep(3)

            # Resize the face image for the model
            face_image_resized = cv2.resize(face_image, (100, 100))

            # Convert the face to an array and normalize
            face_image_resized = face_image_resized.astype('float32') / 255.0
            img_pred = np.expand_dims(face_image_resized, axis=0)

            # Predict emotions
            rslt = model.predict(img_pred)
            label = CATAGORIES[np.argmax(rslt)]
            confidence = np.max(rslt) * 100

            # Display the bounding box and predicted label
            cv2.rectangle(frame, (x, y), (x1, y1), (0, 255, 0), 2)
            cv2.putText(frame, f'{label} ({confidence:.2f}%)', (x, y - 10),
                        cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2, cv2.LINE_AA)

    # Show the frame with predictions
    cv2.imshow('IP Camera Face Detection', frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

انجام حدس روی خروجی دوربین تلفن‌همراه به کد

منابع

https://www.linkedin.com/pulse/facial-emotion-recognition-fer-market-trends-eqfwe/?trackingId=oPJOQdEbQGe9Q%2FXdlKTFDg%3D%3D

https://paperswithcode.com/datasets?task=facial-expression-recognition

https://www.researchgate.net/publication/363425999_Facial_emotion_recognition_based_realtime_learner_engagement_detection_system_in_online_learning_context_using_deep_learning_models

https://paperswithcode.com/sota/facial-expression-recognition-on-affectnet?metric=Accuracy%20(7%20emotion)