

باسمه تعالی



دانشکده مهندسی مکانیک

محمد مهدی انصاری

محمد مهدی برّاز

محمد جواد قاضی خانی

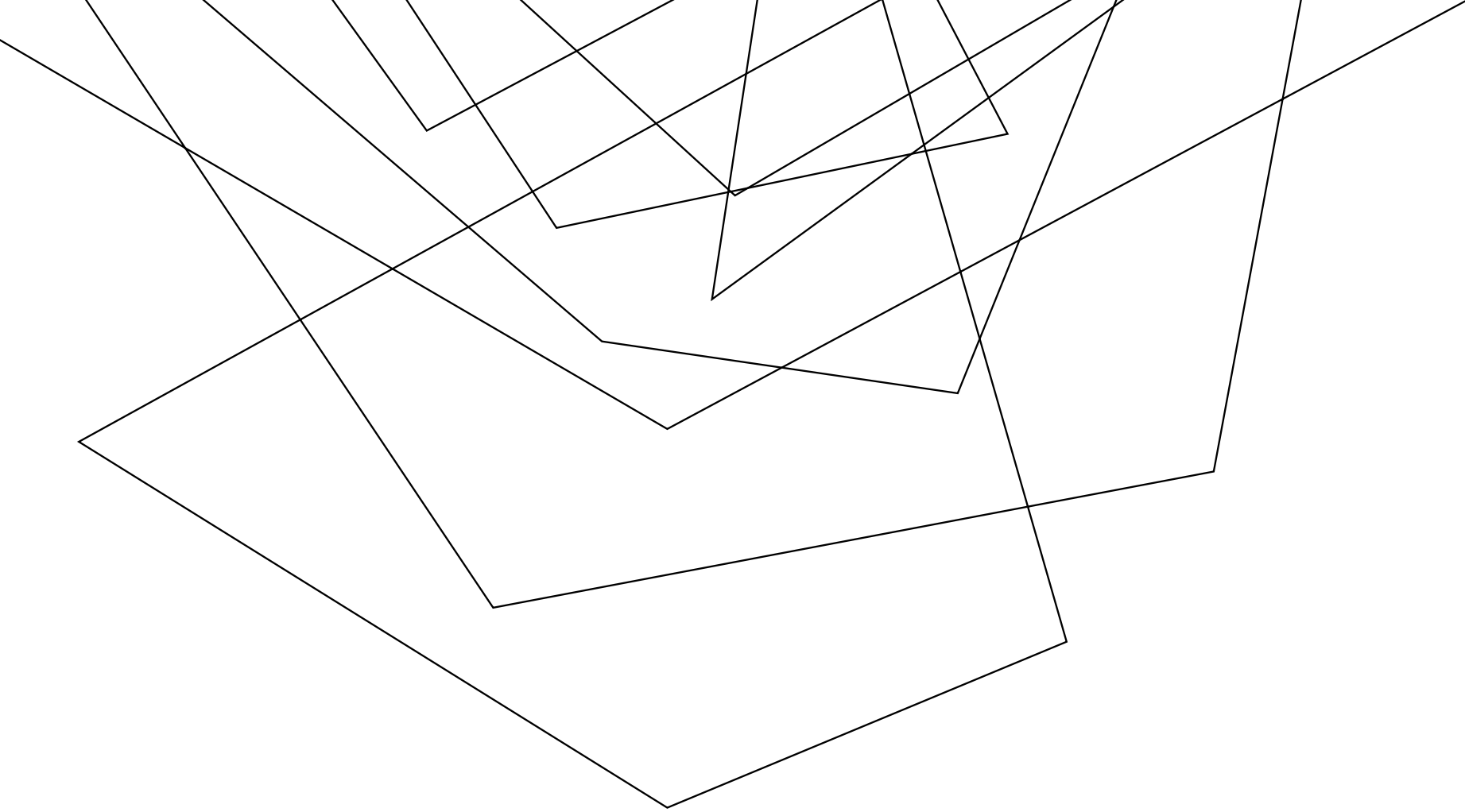
درس:

هوش مصنوعی

استاد:

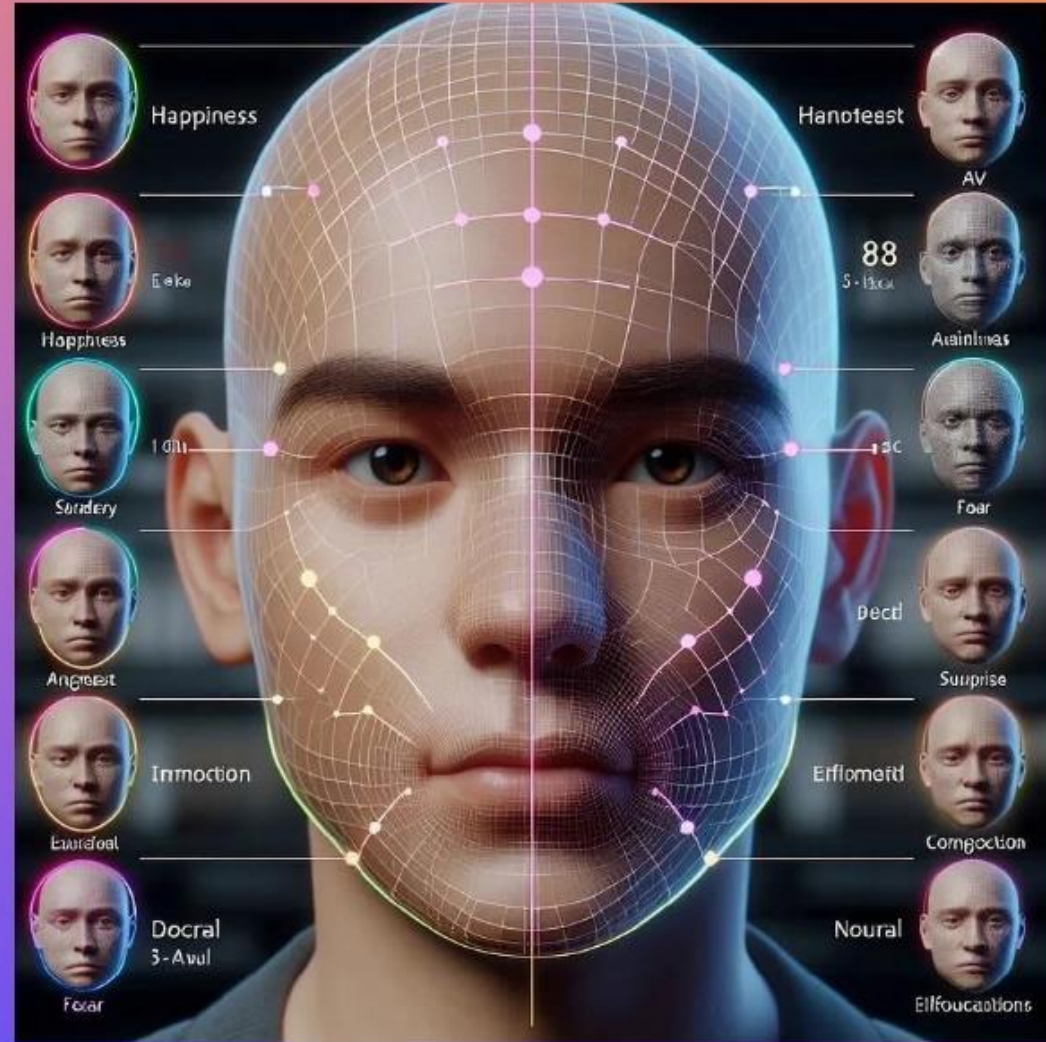
دکتر سید حسن ذبیحی فر

پاییز 1403



FACIAL EXPRESSION RECOGNITION

تشخيص احساسات صورت



FACIAL EMOTION RECOGNITION

Facial emotion
recognition
technology

UNLOCKING EMOTIONS,
ENHANCING CONNECTIONS



فهرست مطالب

مارکت FER

انواع دیتاست‌ها

کتابخانه دیپ‌فیس

شبکه‌های اجرا شده روی دیتاست‌ها

نمایش معماری شبکه‌های عصبی

مقایسه بهترین دقت مدل‌ها با مقالات

مشکلات و راه‌های افزایش دقت

نمونه کد

تست‌های گرفته شده

روش‌های تست

منابع

An abstract graphic design featuring two thin, dark gray lines that intersect on a light gray background. One line is oriented diagonally from the top-left towards the bottom-right, while the other is steeper, running from the top-center towards the bottom-right. The intersection point is located in the upper-left quadrant of the image.

ماركت FER

بازار تشخیص احساسات چهره (Facial Emotion Recognition - FER) تحلیل روندها و چشم اندازهای منطقه ای

بازار جهانی FER در سال 2022 به ارزش 2.9 میلیارد دلار بود و پیش بینی می شود تا سال 2030 به 7.7 میلیارد دلار برسد (نرخ رشد سالانه مرکب 14.5% از 2024 تا 2030). این بازار تحت تأثیر روندهای منطقه ای و تفاوت های فرهنگی به سرعت در حال رشد است.

تحلیل منطقه ای

• آمریکای شمالی:

این منطقه، با تمرکز بر ایالات متحده و کانادا، بیشترین سهم بازار جهانی FER (بیش از 40%) را دارد. شرکت های بزرگی نظیر مایکروسافت و IBM در صنایع مختلف از جمله بهداشت، خرده فروشی و خودروسازی روی توسعه این فناوری سرمایه گذاری می کنند.

• اروپا:

قوانین سخت گیرانه حریم خصوصی، از جمله GDPR، موجب کاهش سرعت پذیرش FER در اروپا شده است. با این حال، پیشرفت در یادگیری ماشین و همکاری های شرکتی به رشد بازار در کشورهایمانند انگلستان، آلمان و فرانسه کمک کرده است.

• آسیا-اقیانوسیه:

رشد سریع این بازار به دلیل سرمایه گذاری در فناوری های هوشمند و تقاضای روزافزون برای اتوماسیون است. چین و هند پیشگام این رشد هستند، اما تفاوت های قوانین ملی مانعی برای توسعه یکنواخت در منطقه است.

• خاورمیانه و آفریقا:

این بازار در مراحل اولیه خود قرار دارد، اما کشورهایمانند امارات و آفریقای جنوبی در حال افزایش سرمایه گذاری در این زمینه برای بهبود تجربه مشتری هستند.

• آمریکای لاتین:

کشورهاییمانند برزیل و آرژانتین به آرامی در حال استفاده از FER در خرده فروشی و امنیت هستند. چالش هایی نظیر بی ثباتی اقتصادی می تواند روند رشد را کند کند.

تقسیم‌بندی بازار

. بر اساس نوع:

- آنلاین

- آفلاین

. بر اساس کاربرد:

- دولت

- خرده‌فروشی

- بهداشت

- سرگرمی

-

شرکت‌های کلیدی فعال در بازار FER

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

روندهای کلیدی و فرصت‌ها

رشد فناوری:

- پیشرفت در الگوریتم‌های یادگیری عمیق و بینایی کامپیوتر.
- ترکیب FER با واقعیت افزوده و مجازی برای ایجاد تجربه‌های شخصی‌سازی شده.

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.

چالش‌ها:

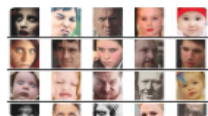
- نگرانی‌های مربوط به حریم خصوصی و رضایت کاربران.
- مسائل اخلاقی در استفاده از داده‌های تشخیص چهره.

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....

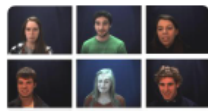
160 PAPERS • 3 BENCHMARKS



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 هوش مصنوعی 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 |

[illegible]

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

Leaderboard

Dataset

View

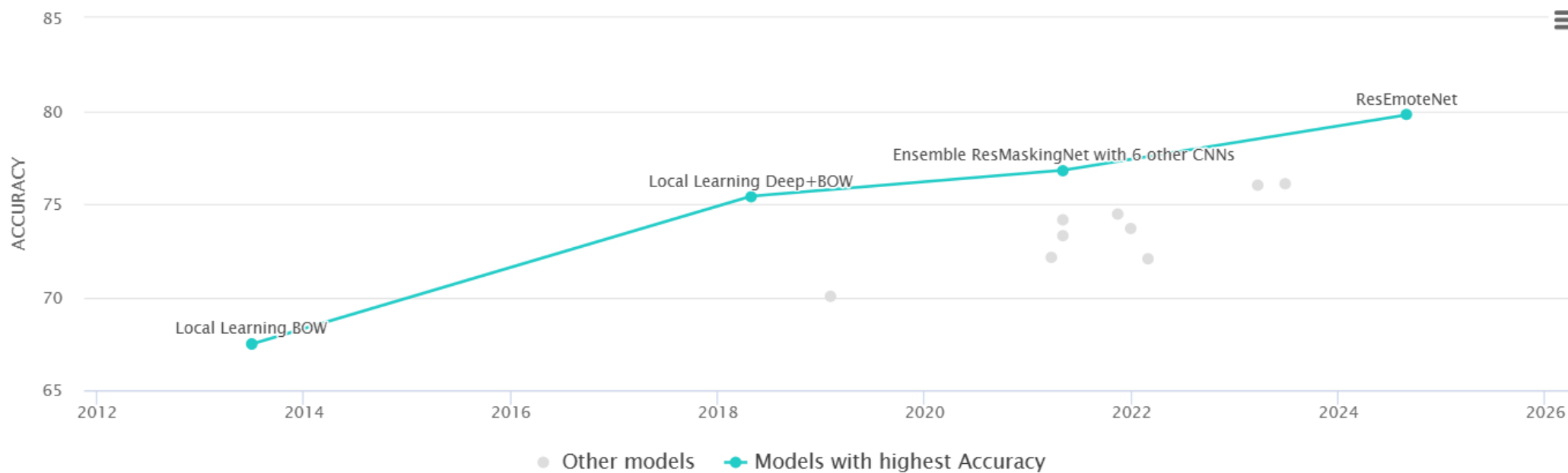
Accuracy ▾

by

Date ▾

for

All models ▾



Facial Expression Recognition (FER) on RAF-DB

Leaderboard

Dataset

View

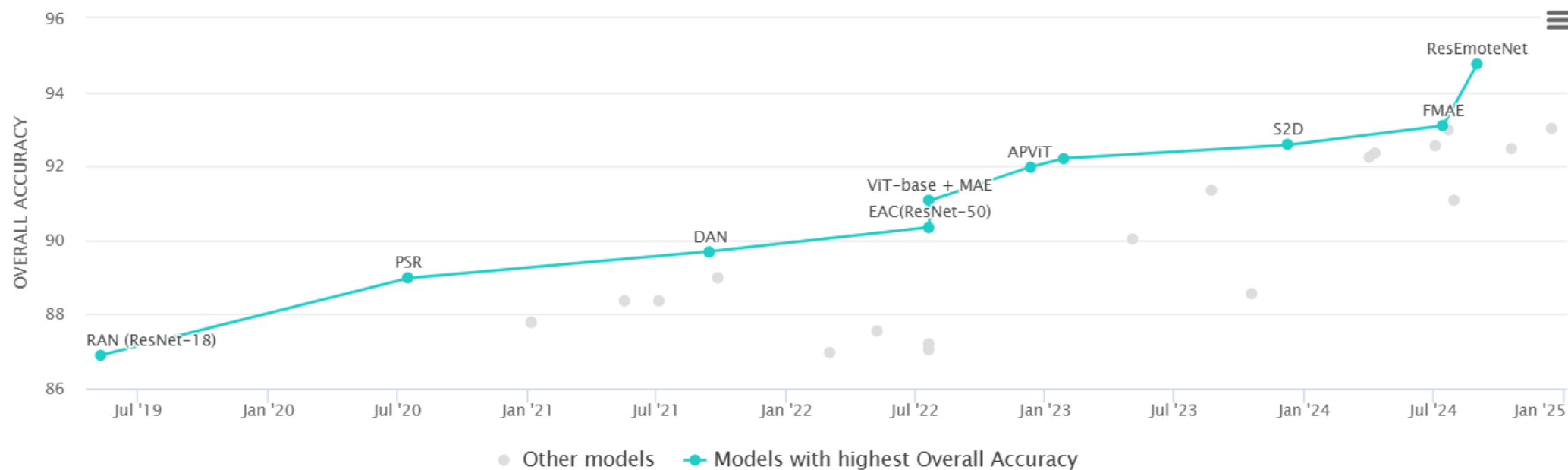
Overall Accuracy

by

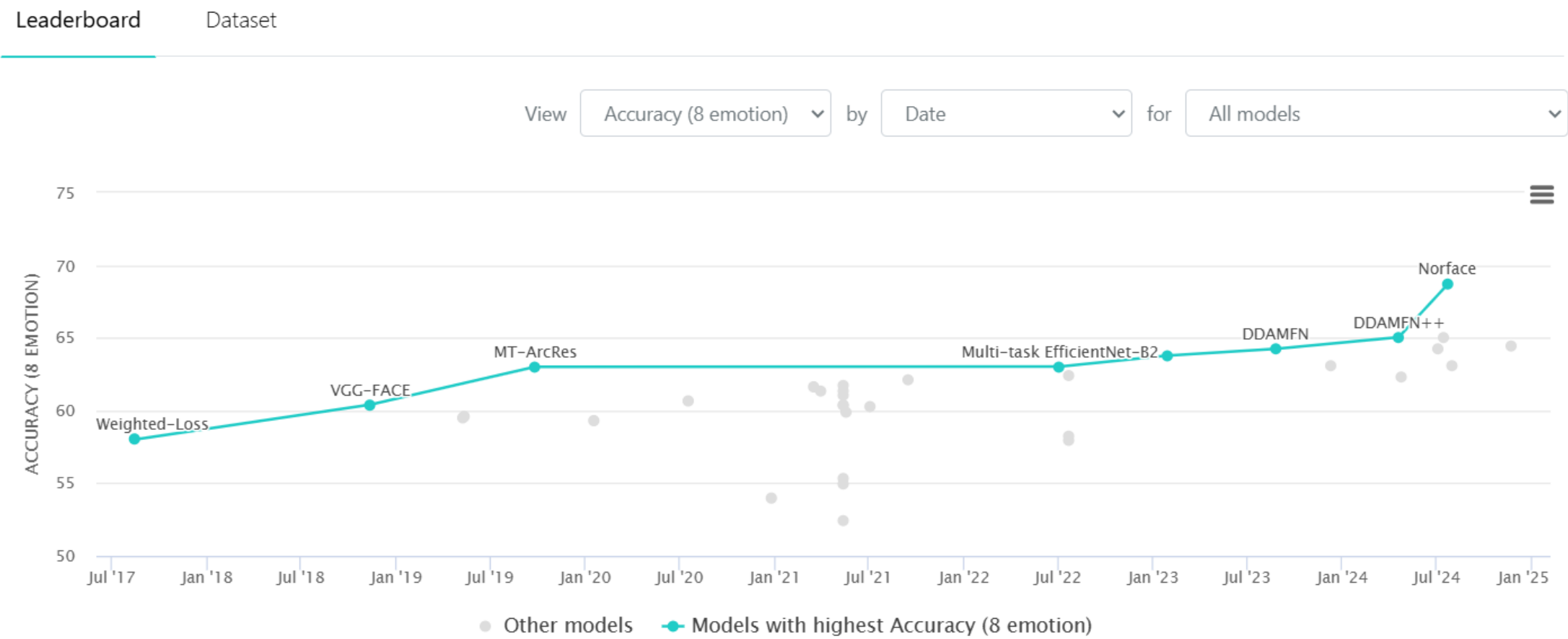
Date

for

All models



Facial Expression Recognition (FER) on AffectNet



کتابخانه دیپ فیس

DeepFace یک کتابخانه منبع باز 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 مانند VGG-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.

An abstract geometric design featuring two thin, dark lines that intersect on a light gray background. One line is oriented diagonally from the top-left towards the bottom-right, while the other is oriented more steeply from the top-left towards the bottom-right, crossing the first line. The intersection point is located in the upper-left quadrant of the image.

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 |
|-----------|----------|---|-------|----|------|------|


```
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 دقت 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 مقایسه مقادیر بدست آورده شده توسط گروه ما برای دقت روی دیتاست‌های مختلف و دقت‌های بدست آمده در مقالات، می‌توان نتیجه گرفت، مدل دیتاست افکت‌نت در سطح بهترین مقالات است؛ همچنین مدل دیتاست رف-دی‌بی با در نظر گرفتن اینکه مقالات از روش‌های ترکیبی و پیچیده استفاده کرده‌اند، مدل ما از دقت مورد قبولی برخوردار است؛ مدل fer2013 نیز نسبت به درصدهای مقالات مورد قبول است.

Two thin, dark gray lines intersect on a light gray background. One line is nearly vertical, and the other is nearly horizontal, creating a large 'X' shape that divides the page.

مشکلات و راه‌های افزایش دقّت

مشکلاتی که شامل بحث FER در حوزه دیتاست‌ها می‌شود، شامل این موارد می‌شود: در بعضی موارد دیتاست کوچک و دقت بالایی را نتیجه می‌دهد، اما تعمیم‌پذیری پایینی دارد؛ در بعضی موارد دیتاست بزرگ بوده و موجب کاهش دقت می‌شود. در دیتاست‌ها بعضاً یک یا چند کلاس 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 دقت می‌کاهد.

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



Santiago Valdarrama • Following

Computer scientist and writer. I teach ...

5h • 🌐



Most people don't know this:

MNIST is the most popular dataset in Machine Learning, and despite millions of people trying, no model has ever solved it with 100% accuracy.

The problem is the initial dataset. There are issues with it.

There's a big lesson here:

You can't out-train bad data.

همین بحث در مورد دیتاست‌های FER نیز صادق است؛ نمونه 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 88 درصدی رف-دی بی برای توضیح نحوه کدنویسی استفاده می شود؛ مابقی با اندکی تغییر، یکسان اند. (به غیر از مدل 68 درصدی افکت نت 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
```

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

```

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)

```


`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)
```

```

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"
                                                                )

```

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

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

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

```

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

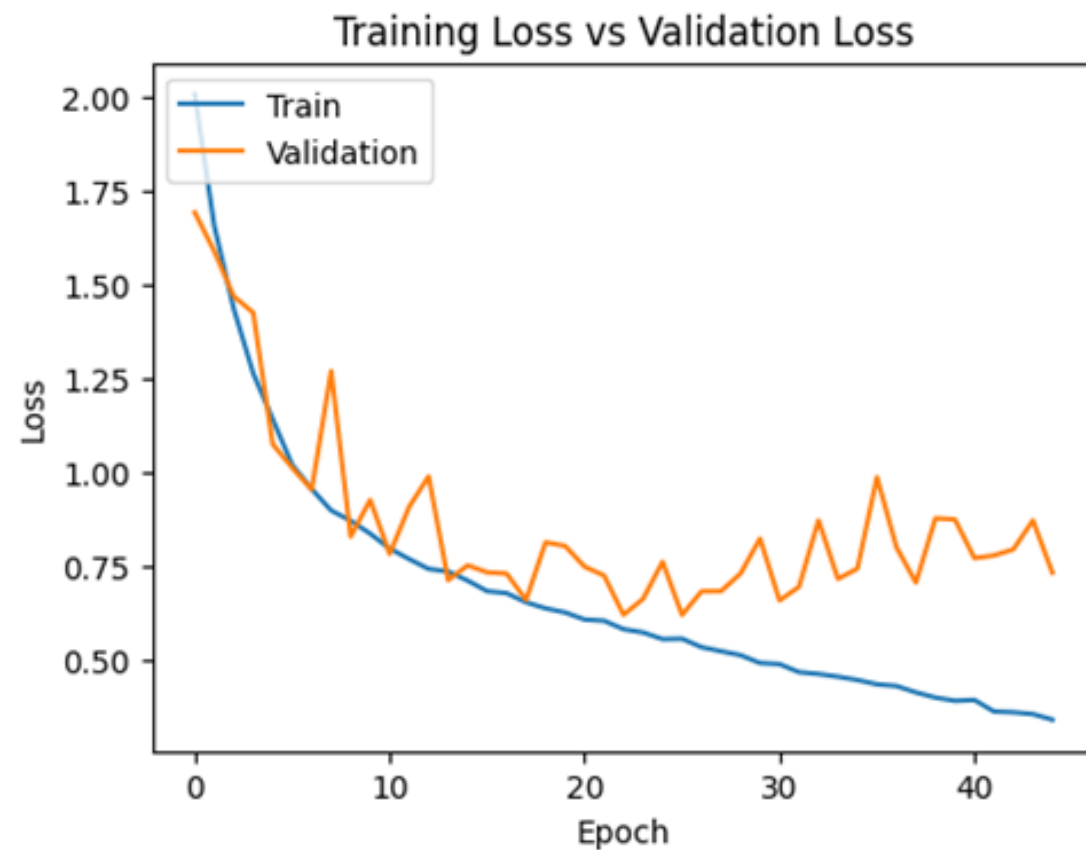
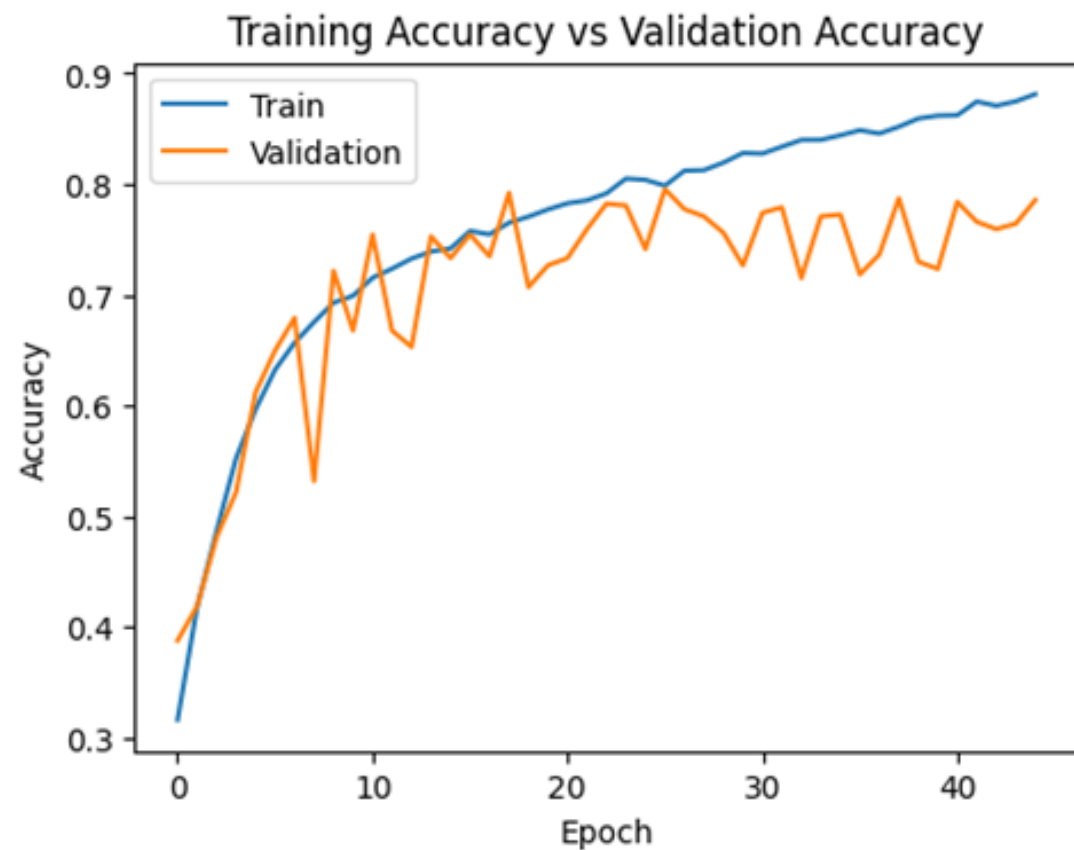
```

```
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()
```





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

A series of white, thin, overlapping geometric lines on a black background, forming a complex, abstract shape on the left side of the slide.

THANK YOU