

In [3]: `from google.colab import files`

`uploaded = files.upload()`

No file chosen

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving kaggle.json to kaggle.json

```
In [4]: # =====
# 1. DOWNLOAD THE IMAGE DATASET
# =====

import os
import csv
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, BatchNormalization, MaxPool2D, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# إعداد ملف الكاجل لتحميل البيانات
os.makedirs('/root/.kaggle', exist_ok=True)
if os.path.exists('/content/data/kaggle.json'):
    !cp /content/data/kaggle.json /root/.kaggle/kaggle.json
elif os.path.exists('/content/kaggle.json'):
    !cp /content/kaggle.json /root/.kaggle/kaggle.json
else:
    if os.path.exists('kaggle.json'):
        !cp kaggle.json /root/.kaggle/kaggle.json

!chmod 600 /root/.kaggle/kaggle.json

print("جاري تحميل البيانات من كاجل...")
!pip install -q kaggle
!kaggle competitions download -c challenges-in-representation-learning-facial-expression-recognition-challenges

print("فك ضغط الملفات...")
!unzip -o challenges-in-representation-learning-facial-expression-recognition-challenges.zip

if os.path.exists('train.csv.zip'):
    !unzip -o train.csv.zip
    print("بنجاح train.csv تم تجهيز وفك ضغط ملف ✓")
```

جاري تحميل البيانات من كاجل...

Downloading challenges-in-representation-learning-facial-expression-recognition-challenges.zip to /content

100% 285M/285M [00:06<00:00, 49.1MB/s]

جاري فك ضغط الملفات...

Archive: challenges-in-representation-learning-facial-expression-recognition-challenges.zip

inflating: example_submission.csv

inflating: fer2013.tar.gz

inflating: icml_face_data.csv

inflating: test.csv

inflating: train.csv

```
In [5]: # =====
# 2. SPLITTING THE DATASET
# =====

train_images = []
train_labels = []

# قراءة البيانات وتحويل نصوص البكسل إلى مصفوفات صور (48x48x1)
with open('train.csv') as train:
    csv_reader = csv.reader(train)
    next(csv_reader)
    for row in csv_reader:
        pixels_str = row[1]
        pixels_list = [int(i) for i in pixels_str.split(' ')]
        pixels_list = np.array(pixels_list, dtype='uint8')
        image = pixels_list.reshape((48, 48, 1))
        train_images.append(image)
        train_labels.append(int(row[0]))

x = np.array(train_images)
y = np.array(train_labels)

# Normalization تقسيم البيانات بنسبة 80% تدريب و 20% اختبار والقيام بالـ
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_
x_train, x_test = x_train / 255.0, x_test / 255.0
```

```
In [6]: # =====
# 3. APPLYING DATA AUGMENTATION
# =====

data_generator = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True
)

flow = data_generator.flow(x_train, y_train, batch_size=256)
```

```
In [8]: # =====
# 4. TRAINING AND TESTING MODELS
# =====

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, BatchNormalization, MaxPool2D, Dropout
from tensorflow.keras.optimizers import Adam

# 1. بناء شبكة CNN
model = Sequential([

    Input(shape=(48, 48, 1)),

    Conv2D(64, (3, 3), padding='same', activation='relu'),
    BatchNormalization(),
    Conv2D(64, (3, 3), padding='same', activation='relu'),
    BatchNormalization(),
    MaxPool2D((2, 2)),
    Dropout(0.25),

    Conv2D(128, (3, 3), padding='same', activation='relu'),
    BatchNormalization(),
```

```

Conv2D(128, (3, 3), padding='same', activation='relu'),
BatchNormalization(),
MaxPool2D((2, 2)),
Dropout(0.3),

Flatten(),
Dense(256, activation='relu'),
BatchNormalization(),
Dropout(0.5),
Dense(7, activation='softmax')
])

# 2. تجميع النموذج
model.compile(
    optimizer=Adam(learning_rate=0.001),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

# (Parameters) عرض هيكل النموذج في المخرجات للتأكد من عدد المعاملات
model.summary()

# 3. تدريب النموذج (Training)
print("\n--- بدء تدريب النموذج على البيانات ---")
history = model.fit(
    flow,
    epochs=30,
    validation_data=(x_test, y_test)
)

# 4. تقييم واختبار النموذج (Testing)
print("\n--- تقييم أداء النموذج على مجموعة الاختبار ---")
test_loss, test_acc = model.evaluate(x_test, y_test)

print("\n" + "="*40)
print(f"🎯 دقة الاختبار (Test Accuracy): {test_acc*100:.2f}%")
print(f"📉 خسارة الاختبار (Test Loss): {test_loss:.4f}")
print("="*40)

```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 48, 48, 64)	640
batch_normalization (BatchNormalization)	(None, 48, 48, 64)	256
conv2d_1 (Conv2D)	(None, 48, 48, 64)	36,928
batch_normalization_1 (BatchNormalization)	(None, 48, 48, 64)	256
max_pooling2d (MaxPooling2D)	(None, 24, 24, 64)	0
dropout (Dropout)	(None, 24, 24, 64)	0
conv2d_2 (Conv2D)	(None, 24, 24, 128)	73,856
batch_normalization_2 (BatchNormalization)	(None, 24, 24, 128)	512
conv2d_3 (Conv2D)	(None, 24, 24, 128)	147,584
batch_normalization_3 (BatchNormalization)	(None, 24, 24, 128)	512
max_pooling2d_1 (MaxPooling2D)	(None, 12, 12, 128)	0
dropout_1 (Dropout)	(None, 12, 12, 128)	0
flatten (Flatten)	(None, 18432)	0
dense (Dense)	(None, 256)	4,718,848
batch_normalization_4 (BatchNormalization)	(None, 256)	1,024
dropout_2 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 7)	1,799



Total params: 4,982,215 (19.01 MB)

Trainable params: 4,980,935 (19.00 MB)

Non-trainable params: 1,280 (5.00 KB)

--- بدء تدريب النموذج على البيانات ---

Epoch 1/30

90/90  41s 275ms/step - accuracy: 0.2515 - loss: 2.1817 - val
_accuracy: 0.2513 - val_loss: 5.4650

Epoch 2/30

90/90  14s 151ms/step - accuracy: 0.3352 - loss: 1.7844 - val
_accuracy: 0.1881 - val_loss: 2.6104

Epoch 3/30

90/90  13s 148ms/step - accuracy: 0.3967 - loss: 1.5910 - val
_accuracy: 0.1797 - val_loss: 1.9150

Epoch 4/30

90/90  13s 148ms/step - accuracy: 0.4334 - loss: 1.4810 - val
_accuracy: 0.1778 - val_loss: 2.0934

Epoch 5/30

90/90  13s 148ms/step - accuracy: 0.4597 - loss: 1.4273 - val
_accuracy: 0.2677 - val_loss: 1.8939

Epoch 6/30

90/90  14s 154ms/step - accuracy: 0.4761 - loss: 1.3710 - val
_accuracy: 0.3981 - val_loss: 1.5073

Epoch 7/30

90/90  13s 148ms/step - accuracy: 0.4904 - loss: 1.3406 - val
_accuracy: 0.4962 - val_loss: 1.3042

Epoch 8/30

90/90  13s 147ms/step - accuracy: 0.5120 - loss: 1.2900 - val
_accuracy: 0.5061 - val_loss: 1.3661

Epoch 9/30

90/90  13s 147ms/step - accuracy: 0.5108 - loss: 1.2919 - val
_accuracy: 0.4422 - val_loss: 1.9785

Epoch 10/30

90/90  13s 150ms/step - accuracy: 0.5142 - loss: 1.2836 - val
_accuracy: 0.5439 - val_loss: 1.1903

Epoch 11/30

90/90  14s 152ms/step - accuracy: 0.5311 - loss: 1.2389 - val
_accuracy: 0.5061 - val_loss: 1.3487

Epoch 12/30

90/90  14s 151ms/step - accuracy: 0.5366 - loss: 1.2118 - val
_accuracy: 0.5409 - val_loss: 1.2092

Epoch 13/30

90/90  14s 151ms/step - accuracy: 0.5483 - loss: 1.1883 - val
_accuracy: 0.5827 - val_loss: 1.1098

Epoch 14/30

90/90  14s 151ms/step - accuracy: 0.5561 - loss: 1.1655 - val
_accuracy: 0.5601 - val_loss: 1.1338

Epoch 15/30

90/90  14s 150ms/step - accuracy: 0.5578 - loss: 1.1599 - val
_accuracy: 0.5728 - val_loss: 1.1260

Epoch 16/30

90/90  14s 150ms/step - accuracy: 0.5691 - loss: 1.1396 - val
_accuracy: 0.5975 - val_loss: 1.0825

Epoch 17/30

90/90  13s 149ms/step - accuracy: 0.5713 - loss: 1.1393 - val
_accuracy: 0.4697 - val_loss: 1.5160

Epoch 18/30

90/90  13s 149ms/step - accuracy: 0.5720 - loss: 1.1448 - val
_accuracy: 0.6017 - val_loss: 1.0745

Epoch 19/30

90/90  14s 150ms/step - accuracy: 0.5810 - loss: 1.1140 - val
_accuracy: 0.5846 - val_loss: 1.0981

Epoch 20/30

90/90  13s 148ms/step - accuracy: 0.5766 - loss: 1.1242 - val

_accuracy: 0.5120 - val_loss: 1.3578
Epoch 21/30
90/90  13s 147ms/step - accuracy: 0.5784 - loss: 1.1107 - val
_accuracy: 0.6102 - val_loss: 1.0445
Epoch 22/30
90/90  13s 149ms/step - accuracy: 0.5917 - loss: 1.0767 - val
_accuracy: 0.5876 - val_loss: 1.1054
Epoch 23/30
90/90  14s 150ms/step - accuracy: 0.5956 - loss: 1.0767 - val
_accuracy: 0.5876 - val_loss: 1.0992
Epoch 24/30
90/90  13s 150ms/step - accuracy: 0.5957 - loss: 1.0834 - val
_accuracy: 0.5724 - val_loss: 1.1830
Epoch 25/30
90/90  13s 149ms/step - accuracy: 0.5984 - loss: 1.0697 - val
_accuracy: 0.5958 - val_loss: 1.0780
Epoch 26/30
90/90  13s 149ms/step - accuracy: 0.6048 - loss: 1.0546 - val
_accuracy: 0.6082 - val_loss: 1.0466
Epoch 27/30
90/90  14s 152ms/step - accuracy: 0.6047 - loss: 1.0450 - val
_accuracy: 0.6191 - val_loss: 1.0176
Epoch 28/30
90/90  13s 149ms/step - accuracy: 0.6086 - loss: 1.0379 - val
_accuracy: 0.6048 - val_loss: 1.0638
Epoch 29/30
90/90  13s 149ms/step - accuracy: 0.6088 - loss: 1.0341 - val
_accuracy: 0.5942 - val_loss: 1.1138
Epoch 30/30
90/90  14s 150ms/step - accuracy: 0.6069 - loss: 1.0400 - val
_accuracy: 0.6299 - val_loss: 0.9853

--- تقييم أداء النموذج على مجموعة الاختبار ---
180/180  1s 5ms/step - accuracy: 0.6299 - loss: 0.9853

=====

 دقة الاختبار (Test Accuracy): 62.99%

 خسارة الاختبار (Test Loss): 0.9853

=====