

Start coding or [generate](#) with AI.

Connctet to Google drive

```
from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

Lib

```
import tensorflow as tf
from tensorflow.keras.preprocessing import image_dataset_from_directory
from tensorflow.keras.applications import EfficientNetB0
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt
import os
```

Dataset path

```
dataset_path = "/content/drive/MyDrive/Kaggle Brain Tumor Classification (MRI)/archive (1)"
```

```
train_dir = os.path.join(dataset_path, "Training")
test_dir = os.path.join(dataset_path, "Testing")
```

```
img_size = (224, 224)
batch_size = 32
```

```
train_ds = image_dataset_from_directory(
    train_dir,
    validation_split=0.2,
    subset="training",
    seed=42,
    image_size=img_size,
    batch_size=batch_size
)
```

```
val_ds = image_dataset_from_directory(
    train_dir,
    validation_split=0.2,
    subset="validation",
    seed=42,
    image_size=img_size,
    batch_size=batch_size
)
```

```
test_ds = image_dataset_from_directory(
    test_dir,
    image_size=img_size,
    batch_size=batch_size
)
```

```
Found 18320 files belonging to 4 classes.
Using 14656 files for training.
Found 18320 files belonging to 4 classes.
Using 3664 files for validation.
Found 394 files belonging to 4 classes.
```

Prefetching

```
AUTOTUNE = tf.data.AUTOTUNE
```

```
train_ds = train_ds.prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.prefetch(buffer_size=AUTOTUNE)
test_ds = test_ds.prefetch(buffer_size=AUTOTUNE)
```

Data augmentation

```
data_augmentation = tf.keras.Sequential([
    layers.RandomFlip("horizontal"),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1),
```

```
layers.RandomContrast(0.1),
])
```

Built EfficientNetB0

```
base_model = EfficientNetB0(
    weights="imagenet",
    include_top=False,
    input_shape=(224, 224, 3)
)

base_model.trainable = False # تجميد الطبقات الأساسية

model = models.Sequential([
    data_augmentation,
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dropout(0.3),
    layers.Dense(4, activation="softmax") # 4 = عدد الفئات
])

model.compile(
    optimizer="adam",
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)

model.summary()
```

Downloading data from https://storage.googleapis.com/keras-applications/efficientnetb0_notop.h5
16705208/16705208 ————— 0s 0us/step
Model: "sequential_1"

Layer (type)	Output Shape	Param #
sequential (Sequential)	?	0 (unbuilt)
efficientnetb0 (Functional)	(None, 7, 7, 1280)	4,049,571
global_average_pooling2d (GlobalAveragePooling2D)	?	0
dropout (Dropout)	?	0
dense (Dense)	?	0 (unbuilt)

Total params: 4,049,571 (15.45 MB)
Trainable params: 0 (0.00 B)
Non-trainable params: 4,049,571 (15.45 MB)

Training model

```
import time

start_time = time.time()

history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=16
)

end_time = time.time()

elapsed_time = end_time - start_time

minutes = elapsed_time // 60
seconds = elapsed_time % 60

print(f"Training time: {int(minutes)} min {int(seconds)} sec")
```

```
Epoch 1/16
458/458 ————— 1856s 4s/step - accuracy: 0.6546 - loss: 0.8257 - val_accuracy: 0.7928 - val_loss: 0.5251
Epoch 2/16
458/458 ————— 1471s 3s/step - accuracy: 0.8254 - loss: 0.4696 - val_accuracy: 0.8182 - val_loss: 0.4579
Epoch 3/16
458/458 ————— 1595s 3s/step - accuracy: 0.8394 - loss: 0.4246 - val_accuracy: 0.8311 - val_loss: 0.4194
Epoch 4/16
458/458 ————— 1484s 3s/step - accuracy: 0.8514 - loss: 0.3944 - val_accuracy: 0.8403 - val_loss: 0.4071
Epoch 5/16
458/458 ————— 1562s 3s/step - accuracy: 0.8586 - loss: 0.3770 - val_accuracy: 0.8210 - val_loss: 0.4593
```

```

Epoch 6/16
458/458 ————— 1476s 3s/step - accuracy: 0.8647 - loss: 0.3618 - val_accuracy: 0.8439 - val_loss: 0.4085
Epoch 7/16
458/458 ————— 1452s 3s/step - accuracy: 0.8627 - loss: 0.3560 - val_accuracy: 0.8422 - val_loss: 0.4076
Epoch 8/16
458/458 ————— 1455s 3s/step - accuracy: 0.8607 - loss: 0.3613 - val_accuracy: 0.8644 - val_loss: 0.3626
Epoch 9/16
458/458 ————— 1455s 3s/step - accuracy: 0.8707 - loss: 0.3488 - val_accuracy: 0.8507 - val_loss: 0.3858
Epoch 10/16
458/458 ————— 1466s 3s/step - accuracy: 0.8631 - loss: 0.3511 - val_accuracy: 0.8679 - val_loss: 0.3489
Epoch 11/16
458/458 ————— 1455s 3s/step - accuracy: 0.8767 - loss: 0.3341 - val_accuracy: 0.8515 - val_loss: 0.3817
Epoch 12/16
458/458 ————— 1464s 3s/step - accuracy: 0.8666 - loss: 0.3487 - val_accuracy: 0.8663 - val_loss: 0.3489
Epoch 13/16
458/458 ————— 1442s 3s/step - accuracy: 0.8685 - loss: 0.3405 - val_accuracy: 0.8649 - val_loss: 0.3535
Epoch 14/16
458/458 ————— 1460s 3s/step - accuracy: 0.8769 - loss: 0.3313 - val_accuracy: 0.8679 - val_loss: 0.3392
Epoch 15/16
458/458 ————— 1490s 3s/step - accuracy: 0.8705 - loss: 0.3433 - val_accuracy: 0.8715 - val_loss: 0.3366
Epoch 16/16
458/458 ————— 1454s 3s/step - accuracy: 0.8733 - loss: 0.3293 - val_accuracy: 0.8734 - val_loss: 0.3301
Training time: 400 min 38 sec

```

Evaluation

```

test_loss, test_acc = model.evaluate(test_ds)
print("Test Accuracy:", test_acc)

```

```

13/13 ————— 53s 4s/step - accuracy: 0.6740 - loss: 0.9658
Test Accuracy: 0.6751269102096558

```

```

test_ds = image_dataset_from_directory(
    test_dir,
    image_size=img_size,
    batch_size=batch_size
)

```

```
class_names = test_ds.class_names
```

```
class_names = test_ds.class_names
```

```
Found 394 files belonging to 4 classes.
```

```
test_ds = test_ds.prefetch(tf.data.AUTOTUNE)
```

Accuracy and Loss

```

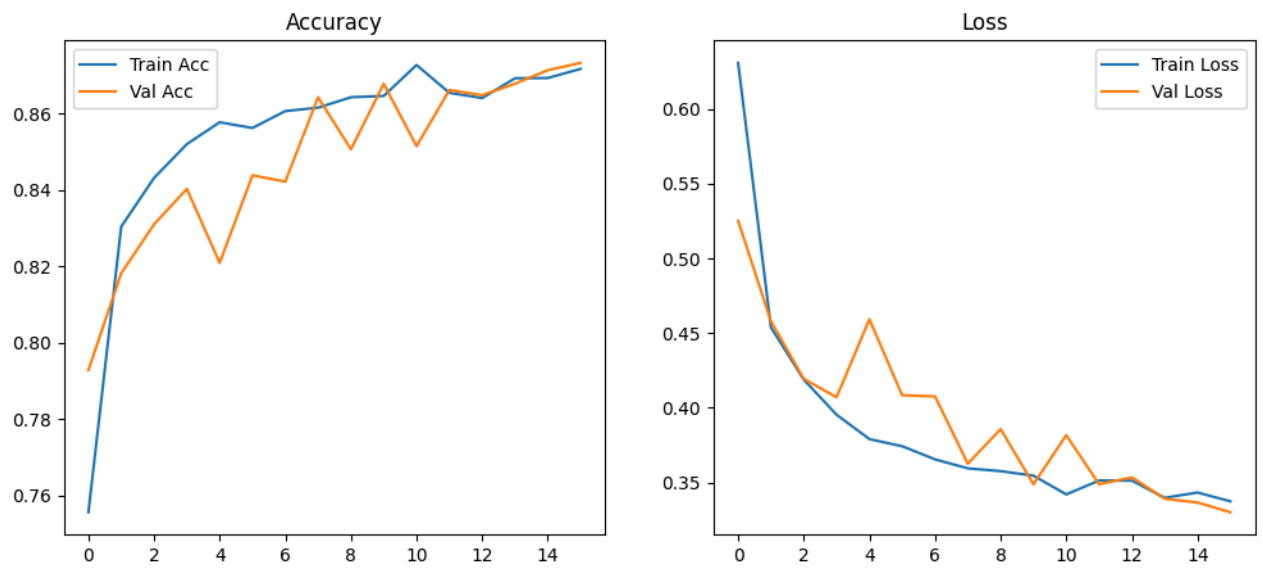
plt.figure(figsize=(12,5))

plt.subplot(1,2,1)
plt.plot(history.history["accuracy"], label="Train Acc")
plt.plot(history.history["val_accuracy"], label="Val Acc")
plt.legend()
plt.title("Accuracy")

plt.subplot(1,2,2)
plt.plot(history.history["loss"], label="Train Loss")
plt.plot(history.history["val_loss"], label="Val Loss")
plt.legend()
plt.title("Loss")

plt.show()

```



Confusion Matrix

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, classification_report

# استخراج الليبلز الحقيقية
y_true = []
for images, labels in test_ds:
    y_true.extend(labels.numpy())
y_true = np.array(y_true)

# توقعات النموذج
y_pred = model.predict(test_ds)
y_pred = np.argmax(y_pred, axis=1)
```

13/13 ————— 36s 3s/step

```
cm = confusion_matrix(y_true, y_pred)

plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=class_names,
            yticklabels=class_names)
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix")
plt.show()
```

