

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

```

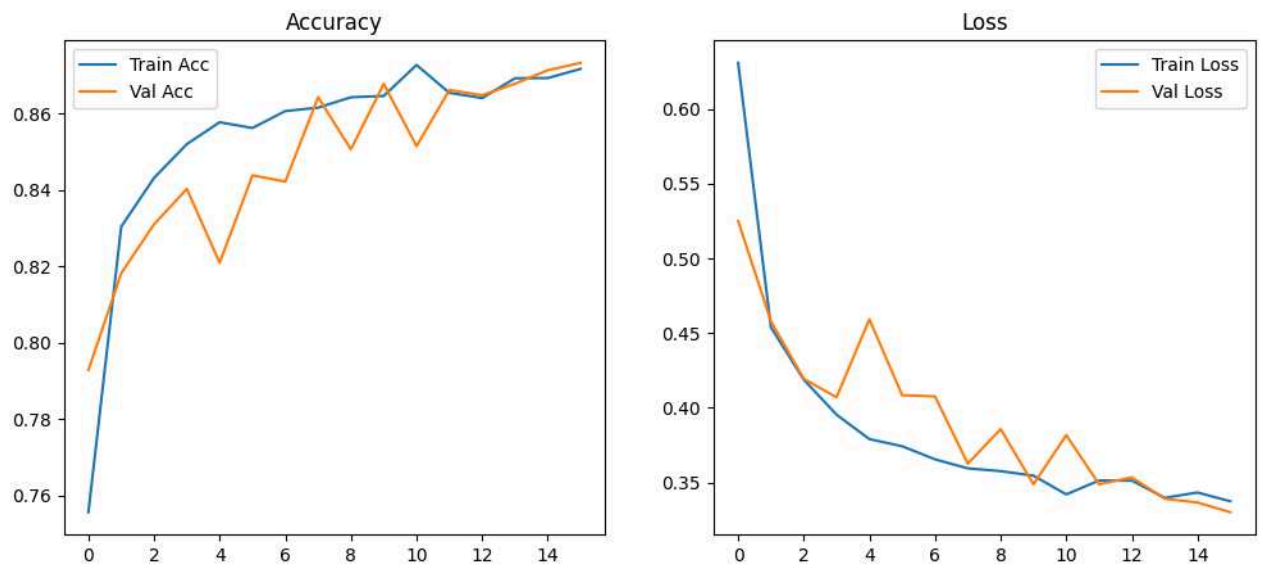
plt.figure(figsize=(12,5))

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

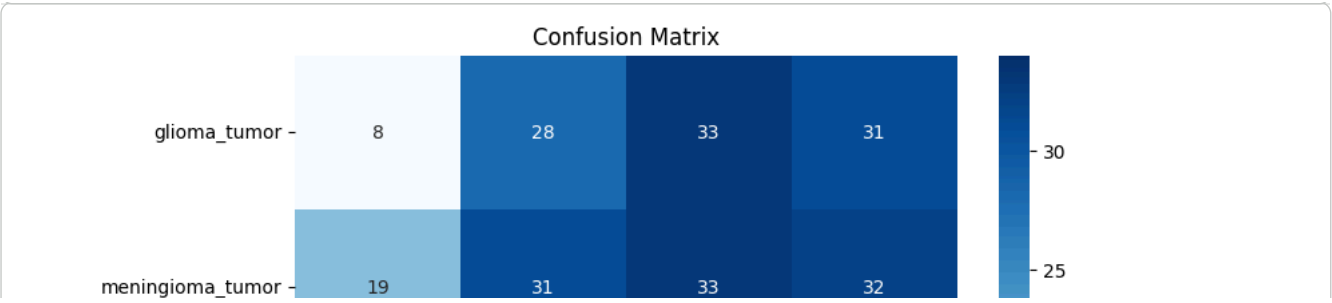
# استخراج الليبلز الحقيقية
y_true = []
for images, labels in test_ds:
    y_true.extend(labels.numpy())
y_true = np.array(y_true)

# 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()
```



1- Connect To Google Drive

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

Mounted at /content/drive

2- Install Packages

```
!pip install tensorflow
!pip install opencv-python
!pip install scikit-image
!pip install numpy pandas
!pip install matplotlib seaborn
!pip install scikit-learn
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.12/dist-packages (2.19.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (25.12.19)
Requirement already satisfied: gast!=0.5.0,!0.5.1,!0.5.2,>=0.2.1 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.12/dist-packages (from tensorflow) (26.0)
Requirement already satisfied: protobuf!=4.21.0,!4.21.1,!4.21.2,!4.21.3,!4.21.4,!4.21.5,<6.0.0dev,>=3.20.3 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (3.20.3)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (2.32.4)
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from tensorflow) (75.2.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.17.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (4.15.0)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (2.1.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.78.1)
Requirement already satisfied: tensorboard~=2.19.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (2.19.0)
Requirement already satisfied: keras>=3.5.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (3.10.0)
Requirement already satisfied: numpy<2.2.0,>=1.26.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (2.0.2)
Requirement already satisfied: h5py>=3.11.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (3.15.1)
Requirement already satisfied: ml-dtypes<1.0.0,>=0.5.1 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (0.5.4)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.12/dist-packages (from astunparse>=1.6.0->tensorflow) (0.44.0)
Requirement already satisfied: rich in /usr/local/lib/python3.12/dist-packages (from keras>=3.5.0->tensorflow) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.12/dist-packages (from keras>=3.5.0->tensorflow) (0.1.0)
Requirement already satisfied: optree in /usr/local/lib/python3.12/dist-packages (from keras>=3.5.0->tensorflow) (0.19.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.21.0->tensorflow) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.21.0->tensorflow) (3.10.1)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.21.0->tensorflow) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.21.0->tensorflow) (2025.11.11)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.12/dist-packages (from tensorboard~=2.19.0->tensorflow) (3.7.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.12/dist-packages (from tensorboard~=2.19.0->tensorflow) (0.7.0)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from tensorboard~=2.19.0->tensorflow) (3.1.0)
Requirement already satisfied: markupsafe>=2.1.1 in /usr/local/lib/python3.12/dist-packages (from werkzeug>=1.0.1->tensorboard~=2.19.0->tensorflow) (3.0.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.12/dist-packages (from rich->keras>=3.5.0->tensorflow) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.12/dist-packages (from rich->keras>=3.5.0->tensorflow) (2.19.0)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.12/dist-packages (from markdown-it-py>=2.2.0->rich->keras>=3.5.0->tensorflow) (0.1.2)
Requirement already satisfied: opencv-python in /usr/local/lib/python3.12/dist-packages (4.13.0.92)
Requirement already satisfied: numpy>=2 in /usr/local/lib/python3.12/dist-packages (from opencv-python) (2.0.2)
Requirement already satisfied: scikit-image in /usr/local/lib/python3.12/dist-packages (0.25.2)
Requirement already satisfied: numpy>=1.24 in /usr/local/lib/python3.12/dist-packages (from scikit-image) (2.0.2)
Requirement already satisfied: scipy>=1.11.4 in /usr/local/lib/python3.12/dist-packages (from scikit-image) (1.16.3)
Requirement already satisfied: networkx>=3.0 in /usr/local/lib/python3.12/dist-packages (from scikit-image) (3.6.1)
Requirement already satisfied: pillow>=10.1 in /usr/local/lib/python3.12/dist-packages (from scikit-image) (11.3.0)
Requirement already satisfied: imageio!=2.35.0,>=2.33 in /usr/local/lib/python3.12/dist-packages (from scikit-image) (2.37.0)
Requirement already satisfied: tifffile>=2022.8.12 in /usr/local/lib/python3.12/dist-packages (from scikit-image) (2026.2.2)
Requirement already satisfied: packaging>=21 in /usr/local/lib/python3.12/dist-packages (from scikit-image) (26.0)
Requirement already satisfied: lazy-loader>=0.4 in /usr/local/lib/python3.12/dist-packages (from scikit-image) (0.4)
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages (2.0.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packages (2.2.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas) (2025.3)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (3.10.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.12/dist-packages (0.13.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.3.3)
```

3- Import Libraries

```
import tensorflow as tf
from tensorflow.keras import layers, models
```

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from tensorflow.keras.preprocessing import image
```

```
!cat /proc/cpuinfo
```

```
processor       : 0
vendor_id      : GenuineIntel
cpu family     : 6
model          : 79
model name     : Intel(R) Xeon(R) CPU @ 2.20GHz
stepping       : 0
microcode      : 0xffffffff
cpu MHz        : 2199.998
cache size     : 56320 KB
physical id    : 0
siblings       : 2
core id        : 0
cpu cores      : 1
apicid         : 0
initial apicid : 0
fpu            : yes
fpu_exception  : yes
cpuid level    : 13
wp             : yes
flags          : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ss ht sy
bugs          : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf mds swapgs taa mmio_stale_data retbleed bhi its
bogomips       : 4399.99
clflush size   : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:

processor       : 1
vendor_id      : GenuineIntel
cpu family     : 6
model          : 79
model name     : Intel(R) Xeon(R) CPU @ 2.20GHz
stepping       : 0
microcode      : 0xffffffff
cpu MHz        : 2199.998
cache size     : 56320 KB
physical id    : 0
siblings       : 2
core id        : 0
cpu cores      : 1
apicid         : 1
initial apicid : 1
fpu            : yes
fpu_exception  : yes
cpuid level    : 13
wp             : yes
flags          : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ss ht sy
bugs          : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf mds swapgs taa mmio_stale_data retbleed bhi its
bogomips       : 4399.99
clflush size   : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:
```

4- Oversampling

```
import os
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
import numpy as np

def oversample_folder(folder_path, target_count):
    datagen = ImageDataGenerator(
        rotation_range=10,
        width_shift_range=0.1,
        height_shift_range=0.1,
        zoom_range=0.1,
        horizontal_flip=True
    )

    images_list = os.listdir(folder_path)
    current_count = len(images_list)

    print(f"Current count in {folder_path}: {current_count}")
```

```

if current_count >= target_count:
    print("No oversampling needed.")
    return

needed = target_count - current_count
print(f"Generating {needed} new images...")

i = 0
while i < needed:
    img_name = np.random.choice(images_list)
    img_path = os.path.join(folder_path, img_name)

    img = image.load_img(img_path, target_size=(224, 224))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, 0)

    aug_iter = datagen.flow(img_array, batch_size=1)

    aug_img = next(aug_iter)[0].astype("uint8")

    save_path = os.path.join(folder_path, f"aug_{i}.jpg")
    image.save_img(save_path, aug_img)

    i += 1

print(f"Folder {folder_path} is now balanced to {target_count} images.")

# Applying Oversampling to each category

base_path = "/content/drive/MyDrive/Kaggle Brain Tumor Classification (MRI)/archive (1)/Training"

folders = [
    "no_tumor",
    "glioma_tumor",
    "meningioma_tumor",
    "pituitary_tumor"
]

# Count the largest number of images
counts = [len(os.listdir(os.path.join(base_path, f))) for f in folders]
target = max(counts)

print("Target count:", target)

# Applying oversampling
for f in folders:
    folder_path = os.path.join(base_path, f)
    oversample_folder(folder_path, target)

```

4- Download data (Training and Testing).

```

train_path = "/content/drive/MyDrive/Kaggle Brain Tumor Classification (MRI)/archive (1)/Training"

train_ds_raw = tf.keras.utils.image_dataset_from_directory(
    train_path,
    image_size=(224, 224),
    batch_size=16,
    shuffle=True
)

test_path = "/content/drive/MyDrive/Kaggle Brain Tumor Classification (MRI)/archive (1)/Testing"
test_ds_raw = tf.keras.utils.image_dataset_from_directory(
    test_path,
    image_size=(224, 224),
    batch_size=16,
    shuffle=False
)

class_names = train_ds_raw.class_names
print("Classes:", class_names)

```

```

Found 13444 files belonging to 4 classes.
Found 394 files belonging to 4 classes.
Classes: ['glioma_tumor', 'meningioma_tumor', 'no_tumor', 'pituitary_tumor']

```

5- Preprocessing

```
normalization = layers.Rescaling(1./255)

train_ds = train_ds_raw.map(lambda x, y: (normalization(x), y))
test_ds = test_ds_raw.map(lambda x, y: (normalization(x), y))

AUTOTUNE = tf.data.AUTOTUNE
train_ds = train_ds.cache().shuffle(1000).prefetch(AUTOTUNE)
test_ds = test_ds.cache().prefetch(AUTOTUNE)
```

6- Building the VisionNet model

```
def build_visionnet(input_shape=(224, 224, 3), num_classes=4):
    model = models.Sequential([
        layers.Conv2D(32, (3,3), activation='relu', padding='same', input_shape=input_shape),
        layers.MaxPooling2D(),

        layers.Conv2D(64, (3,3), activation='relu', padding='same'),
        layers.MaxPooling2D(),

        layers.Conv2D(128, (3,3), activation='relu', padding='same'),
        layers.MaxPooling2D(),

        layers.Flatten(),
        layers.Dense(128, activation='relu'),
        layers.Dropout(0.4),
        layers.Dense(num_classes, activation='softmax')
    ])

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

    return model

visionnet = build_visionnet()
visionnet.summary()
```

[Show hidden output](#)

7- VisionNet training

```
history_vn = visionnet.fit(
    train_ds,
    validation_data=test_ds,
    epochs=16
)
```

```
Epoch 1/16
841/841 ━━━━━━━━━━━ 300s 97ms/step - accuracy: 0.6757 - loss: 0.7949 - val_accuracy: 0.6523 - val_loss: 1.1326
Epoch 2/16
841/841 ━━━━━━━━━━━ 24s 28ms/step - accuracy: 0.8905 - loss: 0.2857 - val_accuracy: 0.7513 - val_loss: 0.7215
Epoch 3/16
841/841 ━━━━━━━━━━━ 23s 28ms/step - accuracy: 0.9433 - loss: 0.1562 - val_accuracy: 0.8223 - val_loss: 0.5941
Epoch 4/16
841/841 ━━━━━━━━━━━ 23s 27ms/step - accuracy: 0.9654 - loss: 0.0994 - val_accuracy: 0.9162 - val_loss: 0.2049
Epoch 5/16
841/841 ━━━━━━━━━━━ 23s 28ms/step - accuracy: 0.9755 - loss: 0.0697 - val_accuracy: 0.9594 - val_loss: 0.1188
Epoch 6/16
841/841 ━━━━━━━━━━━ 23s 28ms/step - accuracy: 0.9826 - loss: 0.0473 - val_accuracy: 0.9492 - val_loss: 0.1533
Epoch 7/16
841/841 ━━━━━━━━━━━ 23s 27ms/step - accuracy: 0.9828 - loss: 0.0487 - val_accuracy: 0.9873 - val_loss: 0.0509
Epoch 8/16
841/841 ━━━━━━━━━━━ 23s 28ms/step - accuracy: 0.9840 - loss: 0.0527 - val_accuracy: 0.9619 - val_loss: 0.1002
Epoch 9/16
841/841 ━━━━━━━━━━━ 23s 28ms/step - accuracy: 0.9885 - loss: 0.0344 - val_accuracy: 0.9848 - val_loss: 0.0396
Epoch 10/16
841/841 ━━━━━━━━━━━ 23s 28ms/step - accuracy: 0.9902 - loss: 0.0294 - val_accuracy: 0.9772 - val_loss: 0.1313
Epoch 11/16
841/841 ━━━━━━━━━━━ 23s 27ms/step - accuracy: 0.9911 - loss: 0.0322 - val_accuracy: 0.9898 - val_loss: 0.0720
Epoch 12/16
841/841 ━━━━━━━━━━━ 23s 28ms/step - accuracy: 0.9926 - loss: 0.0204 - val_accuracy: 0.9924 - val_loss: 0.0230
Epoch 13/16
841/841 ━━━━━━━━━━━ 23s 28ms/step - accuracy: 0.9915 - loss: 0.0241 - val_accuracy: 0.9949 - val_loss: 0.0566
Epoch 14/16
841/841 ━━━━━━━━━━━ 23s 27ms/step - accuracy: 0.9921 - loss: 0.0246 - val_accuracy: 0.9924 - val_loss: 0.0395
Epoch 15/16
```

841/841 ————— 23s 27ms/step - accuracy: 0.9922 - loss: 0.0216 - val_accuracy: 0.9949 - val_loss: 0.0247
 Epoch 16/16
 841/841 ————— 23s 28ms/step - accuracy: 0.9955 - loss: 0.0137 - val_accuracy: 0.9924 - val_loss: 0.0138

8- Drawing Accuracy and Loss Curves

```
acc = history_vn.history['accuracy']
val_acc = history_vn.history['val_accuracy']
loss = history_vn.history['loss']
val_loss = history_vn.history['val_loss']

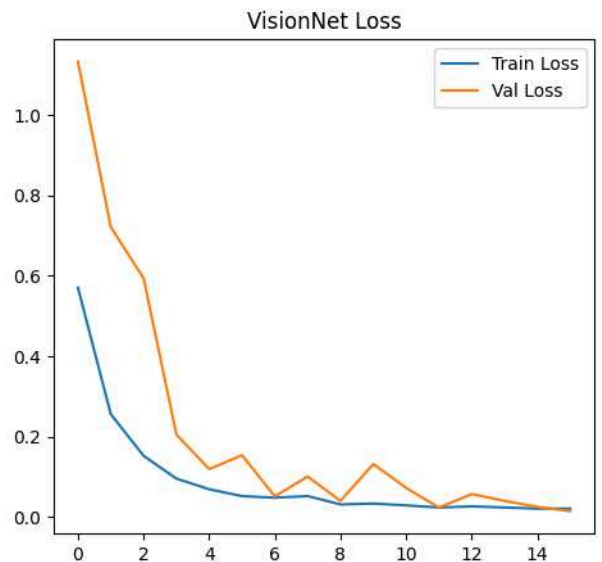
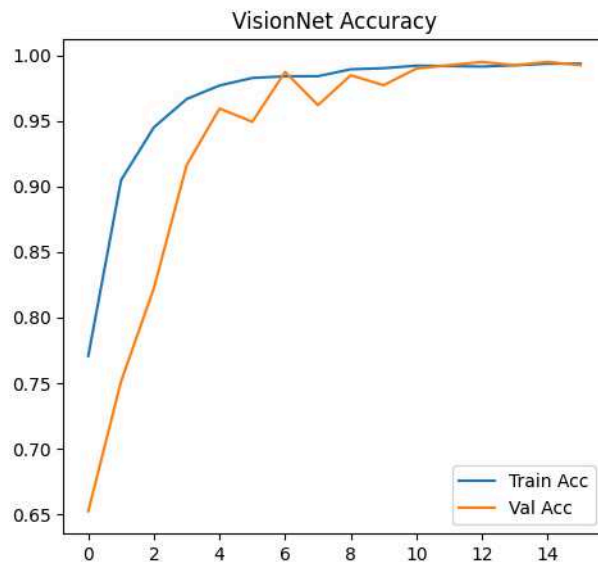
epochs = range(len(acc))

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

plt.subplot(1,2,1)
plt.plot(epochs, acc, label="Train Acc")
plt.plot(epochs, val_acc, label="Val Acc")
plt.legend()
plt.title("VisionNet Accuracy")

plt.subplot(1,2,2)
plt.plot(epochs, loss, label="Train Loss")
plt.plot(epochs, val_loss, label="Val Loss")
plt.legend()
plt.title("VisionNet Loss")

plt.show()
```



9- Evaluate Model

```
test_loss, test_acc = visionnet.evaluate(test_ds)
print("VisionNet Test Accuracy:", test_acc)
```

25/25 ————— 0s 10ms/step - accuracy: 0.9910 - loss: 0.0211
 VisionNet Test Accuracy: 0.9923858046531677

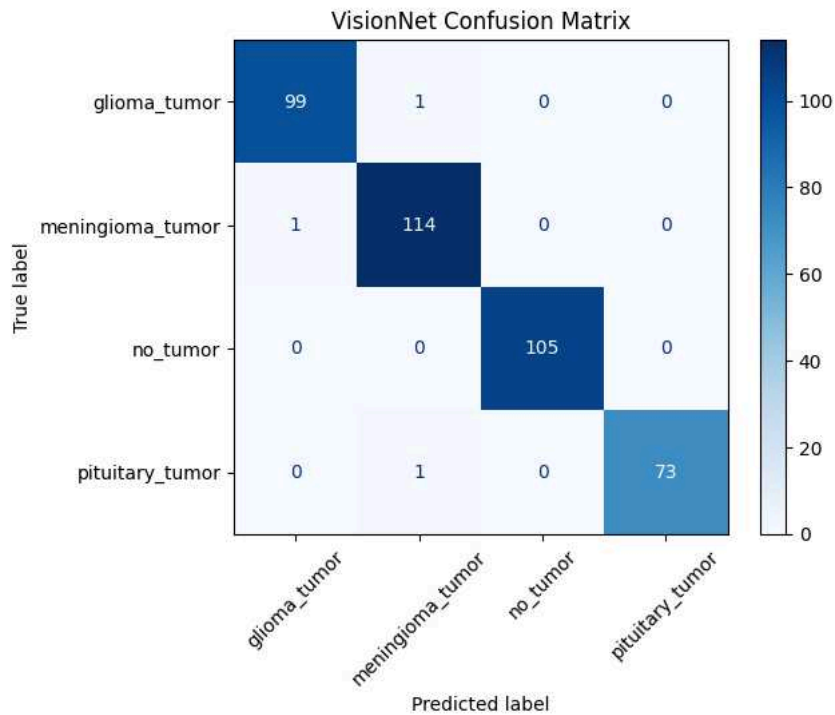
10- Confusion Matrix

```
y_true = []
y_pred = []

for images, labels in test_ds:
    preds = visionnet.predict(images)
    y_true.extend(labels.numpy())
    y_pred.extend(np.argmax(preds, axis=1))

cm = confusion_matrix(y_true, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=class_names)
disp.plot(cmap="Blues", xticks_rotation=45)
plt.title("VisionNet Confusion Matrix")
plt.show()
```

1/1 ————— 0s 396ms/step
 1/1 ————— 0s 50ms/step
 1/1 ————— 0s 54ms/step
 1/1 ————— 0s 50ms/step
 1/1 ————— 0s 52ms/step
 1/1 ————— 0s 49ms/step
 1/1 ————— 0s 50ms/step
 1/1 ————— 0s 48ms/step
 1/1 ————— 0s 48ms/step
 1/1 ————— 0s 47ms/step
 1/1 ————— 0s 49ms/step
 1/1 ————— 0s 48ms/step
 1/1 ————— 0s 47ms/step
 1/1 ————— 0s 50ms/step
 1/1 ————— 0s 60ms/step
 1/1 ————— 0s 48ms/step
 1/1 ————— 0s 48ms/step
 1/1 ————— 0s 51ms/step
 1/1 ————— 0s 49ms/step
 1/1 ————— 0s 48ms/step
 1/1 ————— 0s 52ms/step
 1/1 ————— 0s 53ms/step
 1/1 ————— 0s 48ms/step
 1/1 ————— 0s 49ms/step
 1/1 ————— 0s 392ms/step



11- Prediction for one img

```
img_path = "/content/drive/MyDrive/Te-no_107.jpg" # ضعني مسار الصورة هنا

img = image.load_img(img_path, target_size=(224, 224))
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0) / 255.0

prediction = visionnet.predict(img_array)
pred_class = class_names[np.argmax(prediction)]

print("Predicted Class:", pred_class)
```

1/1 ————— 0s 55ms/step
 Predicted Class: no_tumor

