

```
!unzip archive.zip -d /content/dataset #الملفات يظهر الضغط ويظهر الملفات
```

```

inflating: /content/dataset/dataset/normal/3414_right.jpg
inflating: /content/dataset/dataset/normal/3415_left.jpg
inflating: /content/dataset/dataset/normal/3415_right.jpg
inflating: /content/dataset/dataset/normal/3416_right.jpg
inflating: /content/dataset/dataset/normal/3417_left.jpg
inflating: /content/dataset/dataset/normal/3417_right.jpg
inflating: /content/dataset/dataset/normal/3418_left.jpg
inflating: /content/dataset/dataset/normal/3419_right.jpg
inflating: /content/dataset/dataset/normal/3420_left.jpg
inflating: /content/dataset/dataset/normal/3420_right.jpg
inflating: /content/dataset/dataset/normal/3421_left.jpg
inflating: /content/dataset/dataset/normal/3422_left.jpg
inflating: /content/dataset/dataset/normal/3422_right.jpg
inflating: /content/dataset/dataset/normal/3423_left.jpg
inflating: /content/dataset/dataset/normal/3424_left.jpg
inflating: /content/dataset/dataset/normal/3424_right.jpg
inflating: /content/dataset/dataset/normal/3425_left.jpg
inflating: /content/dataset/dataset/normal/3425_right.jpg
inflating: /content/dataset/dataset/normal/3426_right.jpg
inflating: /content/dataset/dataset/normal/3427_left.jpg
inflating: /content/dataset/dataset/normal/3428_right.jpg
inflating: /content/dataset/dataset/normal/3430_left.jpg
inflating: /content/dataset/dataset/normal/3431_left.jpg
inflating: /content/dataset/dataset/normal/3431_right.jpg
inflating: /content/dataset/dataset/normal/3432_left.jpg
inflating: /content/dataset/dataset/normal/3432_right.jpg
inflating: /content/dataset/dataset/normal/3433_left.jpg
inflating: /content/dataset/dataset/normal/3433_right.jpg
inflating: /content/dataset/dataset/normal/3434_left.jpg
inflating: /content/dataset/dataset/normal/3434_right.jpg
inflating: /content/dataset/dataset/normal/3436_right.jpg
inflating: /content/dataset/dataset/normal/3437_left.jpg
inflating: /content/dataset/dataset/normal/3438_right.jpg
inflating: /content/dataset/dataset/normal/3441_left.jpg
inflating: /content/dataset/dataset/normal/3442_left.jpg
inflating: /content/dataset/dataset/normal/3442_right.jpg
inflating: /content/dataset/dataset/normal/3443_left.jpg
inflating: /content/dataset/dataset/normal/3444_right.jpg
inflating: /content/dataset/dataset/normal/3445_left.jpg
inflating: /content/dataset/dataset/normal/3446_left.jpg
inflating: /content/dataset/dataset/normal/3446_right.jpg
inflating: /content/dataset/dataset/normal/3448_right.jpg
inflating: /content/dataset/dataset/normal/3485_left.jpg
inflating: /content/dataset/dataset/normal/4290_right.jpg
inflating: /content/dataset/dataset/normal/4637_left.jpg
inflating: /content/dataset/dataset/normal/4637_right.jpg
inflating: /content/dataset/dataset/normal/530_left.jpg
inflating: /content/dataset/dataset/normal/530_right.jpg
inflating: /content/dataset/dataset/normal/695_left.jpg
inflating: /content/dataset/dataset/normal/695_right.jpg
inflating: /content/dataset/dataset/normal/84_left.jpg
inflating: /content/dataset/dataset/normal/84_right.jpg
inflating: /content/dataset/dataset/normal/8_left.jpg
inflating: /content/dataset/dataset/normal/8_right.jpg
inflating: /content/dataset/dataset/normal/939_left.jpg
inflating: /content/dataset/dataset/normal/939_right.jpg
inflating: /content/dataset/dataset/normal/951_left.jpg
inflating: /content/dataset/dataset/normal/951_right.jpg

```

```
import os
```

```

print("Checking 'dataset' directory content:")
if os.path.exists('dataset'):
    print(os.listdir('dataset'))
    #التحقق مما إذا كانت الصور داخل مجلد فرعي
    for item in os.listdir('dataset'):
        subpath = os.path.join('dataset', item)
        if os.path.isdir(subpath):
            print(f" Inside '{item}': {os.listdir(subpath)[5:]}") # طباعة أول 5 ملفات
else:
    print("Folder 'dataset' NOT found! Please check where you unzipped the file.")

```

```

Checking 'dataset' directory content:
['dataset']
Inside 'dataset': ['cataract', 'normal', 'diabetic_retinopathy', 'glaucoma']

```

```

## الخطوة : تقليل حجم الصور (المعالجة المسبقة)
#####
import os
from PIL import Image
from tqdm import tqdm

print("\nStep 1: Resizing Images (Preprocessing) - FINAL CORRECTED...")

# تحديد المسار الصحيح بناءً على فحصك السابق
input_root = 'dataset/dataset/'
output_root = 'output/'
resize_size = (256, 256)

# التحقق للتأكد من المسار
if not os.path.exists(input_root):

    input_root = 'dataset/'

print(f"Reading images from: {input_root}")
os.makedirs(output_root, exist_ok=True)

# التأكد من وجود مجلدات داخل المسار
if os.path.exists(input_root):

    for class_name in os.listdir(input_root):
        class_path = os.path.join(input_root, class_name)

        if not os.path.isdir(class_path) or class_name == 'masks':
            continue

        output_class_path = os.path.join(output_root, class_name)
        os.makedirs(output_class_path, exist_ok=True)

        print(f"Processing class: {class_name}")

        count = 0
        # معالجة الصور
        for image_name in tqdm(os.listdir(class_path)):
            if image_name.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp', '.tiff')):
                input_image_path = os.path.join(class_path, image_name)
                # تغيير الامتداد إلى jpg
                output_image_name = os.path.splitext(image_name)[0] + '.jpg'
                output_image_path = os.path.join(output_class_path, output_image_name)

                try:
                    with Image.open(input_image_path) as img:
                        img = img.convert('RGB')
                        img = img.resize(resize_size)
                        img.save(output_image_path, 'JPEG')
                        count += 1
                except Exception as e:
                    print(f"Failed to process {input_image_path}: {e}")
            print(f"-> Processed {count} images for {class_name}")
        else:
            print(f"Error: The directory {input_root} was not found.")

print("\nImage resizing complete.")

```

```

Step 1: Resizing Images (Preprocessing) - FINAL CORRECTED...
Reading images from: dataset/dataset/
Processing class: cataract
100%|██████████| 1038/1038 [00:15<00:00, 68.21it/s]
-> Processed 1038 images for cataract
Processing class: normal
100%|██████████| 1074/1074 [00:05<00:00, 181.77it/s]
-> Processed 1074 images for normal
Processing class: diabetic_retinopathy
100%|██████████| 1098/1098 [00:06<00:00, 158.83it/s]
-> Processed 1098 images for diabetic_retinopathy
Processing class: glaucoma
100%|██████████| 1007/1007 [00:15<00:00, 66.26it/s] -> Processed 1007 images for glaucoma

Image resizing complete.

```

```

## الخطوة 2 : تقسيم مجموعة البيانات
#####
print("\nStep 2: Splitting Dataset into Train/Validation/Test (Safe Mode)...")
import os
import shutil
from sklearn.model_selection import train_test_split

# المسارات
original_dataset_dir = 'output'
base_dir = 'new_dataset'
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'validation')
test_dir = os.path.join(base_dir, 'test')

# إنشاء المجلدات
for dir_path in [train_dir, validation_dir, test_dir]:
    os.makedirs(dir_path, exist_ok=True)

# نسب التقسيم
train_ratio, validation_ratio, test_ratio = 0.7, 0.1, 0.2

def split_and_copy_files(class_dir, dest_base_dir):
    # قراءة الملفات (مع التأكد أنها صور فقط)
    all_images = [
        os.path.join(class_dir, f) for f in os.listdir(class_dir)
        if os.path.isfile(os.path.join(class_dir, f))
        and f.lower().endswith(('.jpg', '.jpeg', '.png', '.bmp'))
    ]

    # --- حماية من الخطأ: إذا كان المجلد فارغاً، تخطاه ---
    if len(all_images) == 0:
        print(f"⚠ Warning: No images found in '{class_dir}'. Skipping this folder.")
        return

    # [cite_start]التقسيم باستخدام sklearn
    train_images, temp_images = train_test_split(all_images, test_size=(1 - train_ratio), random_state=42)
    val_test_ratio = test_ratio / (validation_ratio + test_ratio)
    validation_images, test_images = train_test_split(temp_images, test_size=val_test_ratio, random_state=42)

    datasets = {
        'train': train_images,
        'validation': validation_images,
        'test': test_images
    }

    # نسخ الملفات
    for folder_name, images in datasets.items():
        dest_dir = os.path.join(dest_base_dir, folder_name, os.path.basename(class_dir))
        os.makedirs(dest_dir, exist_ok=True)
        for file_path in images:
            shutil.copy(file_path, dest_dir)

    # تنفيذ التقسيم على كل الفئات
    if os.path.exists(original_dataset_dir):
        for class_name in os.listdir(original_dataset_dir):
            # تخطي المجلدات المخفية مثل .ipynb_checkpoints
            if class_name.startswith('.'):
                continue

            full_class_dir = os.path.join(original_dataset_dir, class_name)
            if os.path.isdir(full_class_dir):
                print(f"Splitting class: {class_name}...")
                split_and_copy_files(full_class_dir, base_dir)
            print("Dataset splitting complete! You can check 'new_dataset' folder.")
    else:
        print("Error: 'output' folder not found. Please run Step 1 again.")

```

```

Step 2: Splitting Dataset into Train/Validation/Test (Safe Mode)...
Splitting class: cataract...
Splitting class: normal...
Splitting class: diabetic_retinopathy...
Splitting class: glaucoma...
Dataset splitting complete! You can check 'new_dataset' folder.

```

```

## الخطوة 3: التحميل
#####
import torch
from torchvision import datasets, transforms as T, models
from torch import nn, optim
from torch.utils.data import DataLoader
import numpy as np
import os
from tqdm import tqdm
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, classification_report
import matplotlib.pyplot as plt
import seaborn as sns

# إعداد الجهاز
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print(f"Using device: {device}")

# --- الخطوة 3: إعداد البيانات ---
print("\nStep 3: Preparing Data Loaders...")
dataset_path = "new_dataset"
batch_size = 32

# دوال التحويل والزيادة
def get_data_loaders(data_dir, batch_size, train=False):
    if train:
        transform = T.Compose([
            T.RandomHorizontalFlip(),
            T.RandomVerticalFlip(),
            T.RandomRotation(10),
            T.Resize((299, 299)),
            T.ToTensor(),
            T.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
        ])
        # تأكد من المسار الصحيح
        train_path = os.path.join(data_dir, "train")
        data = datasets.ImageFolder(train_path, transform=transform)
        loader = DataLoader(data, batch_size=batch_size, shuffle=True, num_workers=2)
        return loader, len(data)
    else:
        transform = T.Compose([
            T.Resize((299, 299)),
            T.ToTensor(),
            T.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
        ])
        val_path = os.path.join(data_dir, "validation")
        test_path = os.path.join(data_dir, "test")

        val_data = datasets.ImageFolder(val_path, transform=transform)
        test_data = datasets.ImageFolder(test_path, transform=transform)

        val_loader = DataLoader(val_data, batch_size=batch_size, shuffle=False, num_workers=2)
        test_loader = DataLoader(test_data, batch_size=batch_size, shuffle=False, num_workers=2)
        return val_loader, test_loader, len(val_data), len(test_data)

# الحصول على الفئات وتحميل البيانات
if os.path.exists(os.path.join(dataset_path, "train")):
    temp_data = datasets.ImageFolder(os.path.join(dataset_path, "train"))
    classes = temp_data.classes
    num_classes = len(classes)
    print(f"Classes found: {classes}")

    (train_loader, train_len) = get_data_loaders(dataset_path, batch_size, train=True)
    (val_loader, test_loader, val_len, test_len) = get_data_loaders(dataset_path, batch_size, train=False)
    dataloaders = {"train": train_loader, "val": val_loader}
    dataset_sizes = {"train": train_len, "val": val_len}
else:
    print("Error: 'new_dataset/train' not found. Please check Step 2.")

```

Using device: cuda

Step 3: Preparing Data Loaders...

Classes found: ['cataract', 'diabetic_retinopathy', 'glaucoma', 'normal']

```

# الخطوة 4: دوال التدريب والتقييم
def train_model(model, criterion, optimizer, scheduler, num_epochs=2):
    for epoch in range(num_epochs):
        print(f"Epoch {epoch+1}/{num_epochs}")
        for phase in ['train', 'val']:
            if phase == 'train': model.train()
            else: model.eval()

            running_loss = 0.0
            running_corrects = 0

            # متابعة التقدم tqdm استخدام
            for inputs, labels in tqdm(dataloaders[phase], leave=False):
                inputs = inputs.to(device)
                labels = labels.to(device)
                optimizer.zero_grad()

                with torch.set_grad_enabled(phase == 'train'):
                    outputs = model(inputs)

                    # الخاصة InceptionV3 التعامل مع مخرجات (Auxiliary Logits)
                    if isinstance(outputs, tuple) or hasattr(outputs, "logits"):
                        main = outputs.logits if hasattr(outputs, "logits") else outputs[0]
                        aux = outputs.aux_logits if hasattr(outputs, "aux_logits") else None

                        loss = criterion(main, labels)
                        if phase == 'train' and aux is not None:
                            loss += 0.4 * criterion(aux, labels)
                    else:
                        main = outputs
                        loss = criterion(main, labels)

                _, preds = torch.max(main, 1)

                if phase == 'train':
                    loss.backward()
                    optimizer.step()

                running_loss += loss.item() * inputs.size(0)
                running_corrects += torch.sum(preds == labels.data)

            if phase == 'train': scheduler.step()

            epoch_loss = running_loss / dataset_sizes[phase]
            epoch_acc = running_corrects.double() / dataset_sizes[phase]
            print(f" {phase} Loss: {epoch_loss:.4f} Acc: {epoch_acc:.4f}")
    return model

def evaluate_model(model_name, model, test_loader, classes):
    print(f"\n--- Evaluating: {model_name} ---")
    model.eval()
    y_true, y_pred = [], []

    with torch.no_grad():
        for x, y in tqdm(test_loader):
            output = model(x.to(device))

            # انشاء التقييم InceptionV3 التعامل مع
            if isinstance(output, tuple) or hasattr(output, "logits"):
                output = output.logits if hasattr(output, "logits") else output[0]

            probabilities = torch.softmax(output, dim=1)
            preds = torch.argmax(probabilities, axis=1).cpu().numpy()

            y_true.extend(y.numpy())
            y_pred.extend(preds)

    print(classification_report(y_true, y_pred, target_names=classes))

    # رسم مصفوفة
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(6,5))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=classes, yticklabels=classes)
    plt.title(f"Confusion Matrix: {model_name}")
    plt.ylabel('True Label')

```

```
plt.xlabel('Predicted Label')
plt.show()
```

```
# --- الخطوة 5: التدريب والمقارنة ---
if 'train_loader' in locals():
    print("\nStep 5: Training Models...")
    EPOCHS = 2

# 1. EfficientNet-B0 (Transfer Learning)
print("\n1. EfficientNet-B0 (Pretrained)")
m1 = models.efficientnet_b0(pretrained=True)
m1.classifier[1] = nn.Linear(m1.classifier[1].in_features, num_classes)
m1 = m1.to(device)
opt = optim.AdamW(m1.parameters(), lr=0.001)
sch = optim.lr_scheduler.StepLR(opt, step_size=3, gamma=0.97)
m1 = train_model(m1, nn.CrossEntropyLoss(), opt, sch, EPOCHS)
evaluate_model("EfficientNet-B0", m1, test_loader, classes)
```

```
/usr/local/lib/python3.12/dist-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated
warnings.warn(
/usr/local/lib/python3.12/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight enum or `
warnings.warn(msg)
```

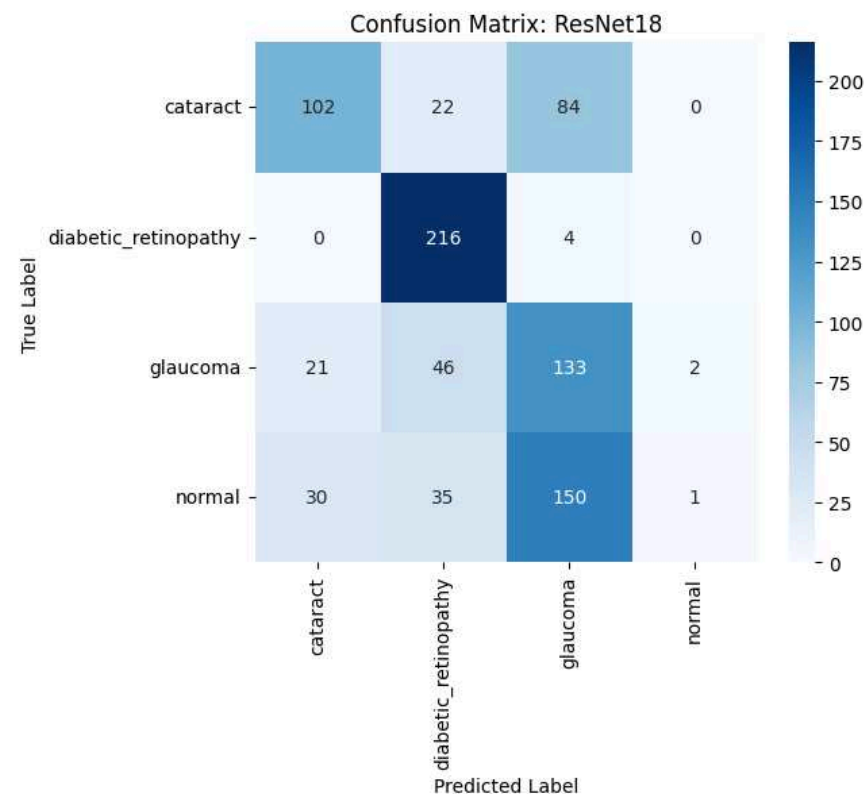
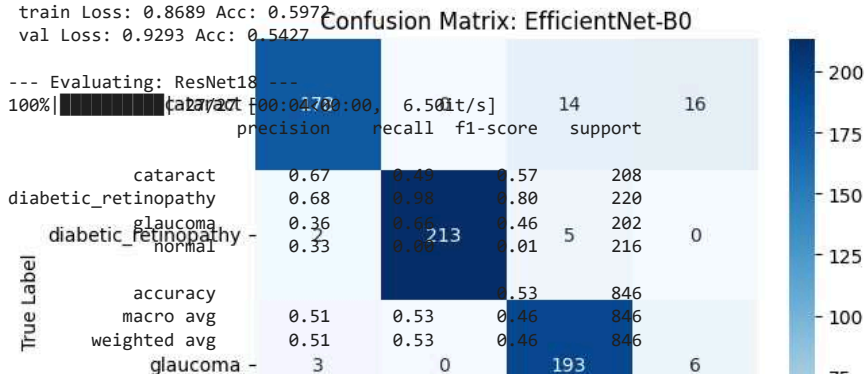
Step 5: Training Models...

```
# 2. ResNet18 (From Scratch)
print("\n2. ResNet18 (From Scratch)")
m2 = models.resnet18(pretrained=False)
m2.fc = nn.Linear(m2.fc.in_features, num_classes)
m2 = m2.to(device)
opt = optim.AdamW(m2.parameters(), lr=0.001)
sch = optim.lr_scheduler.StepLR(opt, step_size=3, gamma=0.97)
m2 = train_model(m2, nn.CrossEntropyLoss(), opt, sch, EPOCHS)
evaluate_model("ResNet18", m2, test_loader, classes)
```

```
/usr/local/lib/python3.12/dist-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated
warnings.warn(
/usr/local/lib/python3.12/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight enum or `
warnings.warn(msg)
```

	precision	recall	f1-score	support
cataract	0.95	0.86	0.90	208
diabetic_retinopathy	0.99	0.97	0.98	220
glaucoma	0.68	0.96	0.79	202
normal	0.86	0.63	0.73	216

```
2. ResNet18 (From Scratch)
Epoch 1/2 accuracy
train Loss: 0.9918 Acc: 0.9027
val Loss: 0.5024 Acc: 0.5048
Epoch 2/2
train Loss: 0.8689 Acc: 0.5972
val Loss: 0.9293 Acc: 0.5427
```



```
# 3. VGG16 (From Scratch)
print("\n3. VGG16 (From Scratch)")
```

```

print(m3.vgg16_from_scratch())
m3 = models.vgg16(pretrained=False)
m3.classifier[6] = nn.Linear(m3.classifier[6].in_features, num_classes)
m3 = m3.to(device)
opt = optim.AdamW(m3.parameters(), lr=0.001)
sch = optim.lr_scheduler.StepLR(opt, step_size=3, gamma=0.97)
m3 = train_model(m3, nn.CrossEntropyLoss(), opt, sch, EPOCHS)
evaluate_model("VGG16", m3, test_loader, classes)

```

3. VGG16 (From Scratch)

```

/usr/local/lib/python3.12/dist-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated
warnings.warn(
/usr/local/lib/python3.12/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight enum or `
warnings.warn(msg)
Epoch 1/2
  train Loss: 5.0500 Acc: 0.2547
  val Loss: 1.3863 Acc: 0.2536
Epoch 2/2
  train Loss: 1.3902 Acc: 0.2516
  val Loss: 1.3860 Acc: 0.2464

```

--- Evaluating: VGG16 ---

100%|██████████| 27/27 [00:08<00:00, 3.31it/s]

```

/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined
_warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined
_warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined
_warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))

```

	precision	recall	f1-score	support
cataract	0.25	1.00	0.39	208
diabetic_retinopathy	0.00	0.00	0.00	220
glaucoma	0.00	0.00	0.00	202
normal	0.00	0.00	0.00	216
accuracy			0.25	846
macro avg	0.06	0.25	0.10	846
weighted avg	0.06	0.25	0.10	846

