

Cell 1 — Imports

```
import os
os.kill(os.getpid(), 9)
```

```
!pip uninstall -y torch torchvision torchaudio
!pip install torch torchvision torchaudio
```

```
Found existing installation: torch 2.10.0+cpu
Uninstalling torch-2.10.0+cpu:
  Successfully uninstalled torch-2.10.0+cpu
Found existing installation: torchvision 0.25.0+cpu
Uninstalling torchvision-0.25.0+cpu:
  Successfully uninstalled torchvision-0.25.0+cpu
Found existing installation: torchaudio 2.10.0+cpu
Uninstalling torchaudio-2.10.0+cpu:
  Successfully uninstalled torchaudio-2.10.0+cpu
Collecting torch
  Downloading torch-2.10.0-cp312-cp312-manylinux_2_28_x86_64.whl.metadata (31 kB)
Collecting torchvision
  Downloading torchvision-0.25.0-cp312-cp312-manylinux_2_28_x86_64.whl.metadata (5.4 kB)
Collecting torchaudio
  Downloading torchaudio-2.10.0-cp312-cp312-manylinux_2_28_x86_64.whl.metadata (6.9 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from torch) (3.25.1)
Requirement already satisfied: typing-extensions>=4.10.0 in /usr/local/lib/python3.12/dist-packages (from torch) (4.10.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from torch) (75.1.0)
Requirement already satisfied: sympy>=1.13.3 in /usr/local/lib/python3.12/dist-packages (from torch) (1.13.3)
Requirement already satisfied: networkx>=2.5.1 in /usr/local/lib/python3.12/dist-packages (from torch) (3.3)
Requirement already satisfied: Jinja2 in /usr/local/lib/python3.12/dist-packages (from torch) (3.1.6)
Requirement already satisfied: fsspec>=0.8.5 in /usr/local/lib/python3.12/dist-packages (from torch) (2024.10.0)
Collecting cuda-bindings==12.9.4 (from torch)
  Downloading cuda_bindings-12.9.4-cp312-cp312-manylinux_2_24_x86_64.manylinux_2_28_x86_64.whl.metadata (1.0 kB)
Collecting nvidia-cuda-nvrtc-cu12==12.8.93 (from torch)
  Downloading nvidia_cuda_nvrtc_cu12-12.8.93-py3-none-manylinux2010_x86_64.manylinux_2_12_x86_64.whl.metadata (1.0 kB)
Collecting nvidia-cuda-runtime-cu12==12.8.90 (from torch)
  Downloading nvidia_cuda_runtime_cu12-12.8.90-py3-none-manylinux2014_x86_64.manylinux_2_17_x86_64.whl.metadata (1.0 kB)
Collecting nvidia-cuda-cupti-cu12==12.8.90 (from torch)
  Downloading nvidia_cuda_cupti_cu12-12.8.90-py3-none-manylinux2014_x86_64.manylinux_2_17_x86_64.whl.metadata (1.0 kB)
Collecting nvidia-cudnn-cu12==9.10.2.21 (from torch)
  Downloading nvidia_cudnn_cu12-9.10.2.21-py3-none-manylinux_2_27_x86_64.whl.metadata (1.8 kB)
Collecting nvidia-cublas-cu12==12.8.4.1 (from torch)
  Downloading nvidia_cublas_cu12-12.8.4.1-py3-none-manylinux_2_27_x86_64.whl.metadata (1.7 kB)
Collecting nvidia-cufft-cu12==11.3.3.83 (from torch)
  Downloading nvidia_cufft_cu12-11.3.3.83-py3-none-manylinux2014_x86_64.manylinux_2_17_x86_64.whl.metadata (1.0 kB)
Collecting nvidia-curand-cu12==10.3.9.90 (from torch)
  Downloading nvidia_curand_cu12-10.3.9.90-py3-none-manylinux_2_27_x86_64.whl.metadata (1.7 kB)
Collecting nvidia-cusolver-cu12==11.7.3.90 (from torch)
  Downloading nvidia_cusolver_cu12-11.7.3.90-py3-none-manylinux_2_27_x86_64.whl.metadata (1.8 kB)
Collecting nvidia-cusparselt-cu12==0.7.1 (from torch)
  Downloading nvidia_cusparselt_cu12-0.7.1-py3-none-manylinux2014_x86_64.whl.metadata (7.0 kB)
Collecting nvidia-nccl-cu12==2.27.5 (from torch)
  Downloading nvidia_nccl_cu12-2.27.5-py3-none-manylinux2014_x86_64.manylinux_2_17_x86_64.whl.metadata (1.0 kB)
Collecting nvidia-nvshmem-cu12==3.4.5 (from torch)
  Downloading nvidia_nvshmem_cu12-3.4.5-py3-none-manylinux2014_x86_64.manylinux_2_17_x86_64.whl.metadata (1.0 kB)
Collecting nvidia-nvtx-cu12==12.8.90 (from torch)
  Downloading nvidia_nvtx_cu12-12.8.90-py3-none-manylinux2014_x86_64.manylinux_2_17_x86_64.whl.metadata (1.0 kB)
Collecting nvidia-nvjitlink-cu12==12.8.93 (from torch)
  Downloading nvidia_nvjitlink_cu12-12.8.93-py3-none-manylinux2010_x86_64.manylinux_2_12_x86_64.whl.metadata (1.0 kB)
Collecting nvidia-cufile-cu12==1.13.1.3 (from torch)
  Downloading nvidia_cufile_cu12-1.13.1.3-py3-none-manylinux2014_x86_64.manylinux_2_17_x86_64.whl.metadata (1.0 kB)
Collecting triton==3.6.0 (from torch)
  Downloading triton-3.6.0-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl.metadata (1.7 kB)
Collecting cuda-pathfinder~=1.1 (from cuda-bindings==12.9.4->torch)
  Downloading cuda_pathfinder-1.1.0-py3-none-any.whl.metadata (1.0 kB)
```

```
import torch
print(torch.__version__)
print("CUDA available:", torch.cuda.is_available())
```

```
2.10.0+cu128
CUDA available: False
```

```
import os
import numpy as np
import torch
```

```
from torchvision import datasets, transforms as T, models
from torch.utils.data import DataLoader
from torch import nn, optim
```

Cell 2 — Resize Images **bold text** Lecture resizes images to 256×256.

```
input_root = "dataset"
output_root = "output"
resize_size = (256,256)

os.makedirs(output_root, exist_ok=True)

for class_name in os.listdir(input_root):

    class_path = os.path.join(input_root,class_name)

    if not os.path.isdir(class_path):
        continue

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

    for image_name in os.listdir(class_path):

        if image_name.lower().endswith(('.png','.jpg','.jpeg','.bmp','.tiff')):

            input_image = os.path.join(class_path,image_name)
            output_image = os.path.join(output_class,image_name)

            try:
                with Image.open(input_image) as img:
                    img = img.convert("RGB")
                    img = img.resize(resize_size)
                    img.save(output_image,"JPEG")
            except:
                pass
```

Cell 3 — Split Dataset (70 / 10 / 20)

```
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")

for dir in [train_dir,validation_dir,test_dir]:
    os.makedirs(dir,exist_ok=True)

train_ratio = 0.7
validation_ratio = 0.1
test_ratio = 0.2
```

Cell 4 — Splitting Function

```
def split_and_copy_files(class_dir, dest_base_dir):

    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))
    ]

    train_images, temp_images = train_test_split(
        all_images,
        test_size=(1-train_ratio),
        random_state=42
    )

    validation_images, test_images = train_test_split(
        temp_images,
        test_size=test_ratio/(validation_ratio+test_ratio),
```

```

        random_state=42
    )

    for images, folder in zip(
        [train_images, validation_images, test_images],
        ["train", "validation", "test"]
    ):

        dest = os.path.join(dest_base_dir, folder, os.path.basename(class_dir))
        os.makedirs(dest, exist_ok=True)

        for file in images:
            import shutil
            shutil.copy(file, dest)

```

Cell 5 — Apply Split

```
from sklearn.model_selection import train_test_split
```

```

for class_dir in os.listdir(original_dataset_dir):

    full_class = os.path.join(original_dataset_dir, class_dir)

    if os.path.isdir(full_class):
        split_and_copy_files(full_class, base_dir)

```

Cell 6 — Data Augmentation

```

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]
            )
        ])

        train_data = datasets.ImageFolder(
            os.path.join(data_dir, "train"),
            transform=transform
        )

        train_loader = DataLoader(train_data, batch_size=batch_size, shuffle=True)

        return train_loader, len(train_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_data = datasets.ImageFolder(
            os.path.join(data_dir, "validation"),
            transform=transform
        )

        test_data = datasets.ImageFolder(
            os.path.join(data_dir, "test"),
            transform=transform
        )

```

```

val_loader = DataLoader(val_data,batch_size=batch_size)
test_loader = DataLoader(test_data,batch_size=batch_size)

return val_loader,test_loader,len(val_data),len(test_data)

```

Cell 7 — Load Dataset

```

dataset_path = "new_dataset"
batch_size = 32

train_loader, train_data_len = get_data_loaders(dataset_path,batch_size,train=True)

val_loader, test_loader, valid_data_len, test_data_len = get_data_loaders(dataset_path,batch_size,train

```

Cell 8 — Classes

```

def get_classes(data_dir):

    data = datasets.ImageFolder(data_dir)

    return data.classes

classes = get_classes("new_dataset/train")

num_classes = len(classes)

print(classes)
print("Number of classes:", num_classes)

['cataract', 'diabetic_retinopathy', 'glaucoma', 'normal']
Number of classes: 4

```

Cell 9 — Setup Training Variables

```

dataloaders = {
    "train": train_loader,
    "val": val_loader
}

dataset_sizes = {
    "train": train_data_len,
    "val": valid_data_len
}

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

```

Cell 10 — EfficientNet Model

```

model = models.efficientnet_b0(pretrained=True)

model.classifier[1] = nn.Linear(
    model.classifier[1].in_features,
    num_classes
)

model = model.to(device)

/usr/local/lib/python3.12/dist-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pr
warnings.warn(
/usr/local/lib/python3.12/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other t
warnings.warn(msg)

```

Cell 11 — Loss + Optimizer

```

# Cell 11 - Loss + Optimizer

criterion = nn.CrossEntropyLoss().to(device)

```

```
optimizer = optim.AdamW(
    model.parameters(),
    lr=0.001
)

exp_lr_scheduler = optim.lr_scheduler.StepLR(
    optimizer,
    step_size=3,
    gamma=0.97
)
```

Cell 12 — Training Function

```
def train_model(model,criterion,optimizer,scheduler,num_epochs=2):

    for epoch in range(num_epochs):

        print(f"Epoch {epoch+1}/{num_epochs}")
        print("-"*10)

        for phase in ["train","val"]:

            if phase=="train":
                model.train()
            else:
                model.eval()

            running_loss=0
            running_corrects=0

            for inputs,labels in dataloaders[phase]:

                inputs=inputs.to(device)
                labels=labels.to(device)

                optimizer.zero_grad()

                with torch.set_grad_enabled(phase=="train"):

                    outputs=model(inputs)

                    loss=criterion(outputs,labels)

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

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

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

            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}")

            scheduler.step()

    return model
```

Cell 13 — Train

```
model = train_model(
    model,
    criterion,
    optimizer,
    exp_lr_scheduler,
    num_epochs=2
)
```

Epoch 1/2

train Loss: 0.5228 Acc: 0.7955

```
val Loss: 0.3843 Acc: 0.8791
Epoch 2/2
-----
train Loss: 0.3249 Acc: 0.8776
val Loss: 0.3043 Acc: 0.9052
```

Cell 14 – Evaluate

```
from tqdm import tqdm
```

```
y_true=[]
y_pred=[]

with torch.no_grad():

    for x,y in tqdm(test_loader):

        output=model(x.to(device))

        probabilities=torch.softmax(output,dim=1)

        pred=torch.argmax(probabilities,axis=1).cpu().numpy()

        y_true.extend(y.numpy())
        y_pred.extend(pred)
```

100%|██████████| 27/27 [01:12<00:00, 2.68s/it]

Cell 15 – Metrics

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

```
print("Accuracy:", accuracy_score(y_true, y_pred))
print("Precision:", precision_score(y_true, y_pred, average="weighted"))
print("Recall:", recall_score(y_true, y_pred, average="weighted"))
print("F1:", f1_score(y_true, y_pred, average="weighted"))
```

```
Accuracy: 0.8865248226950354
Precision: 0.886738430334663
Recall: 0.8865248226950354
F1: 0.8860510841000871
```

Cell 16 – Confusion Matrix

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
```

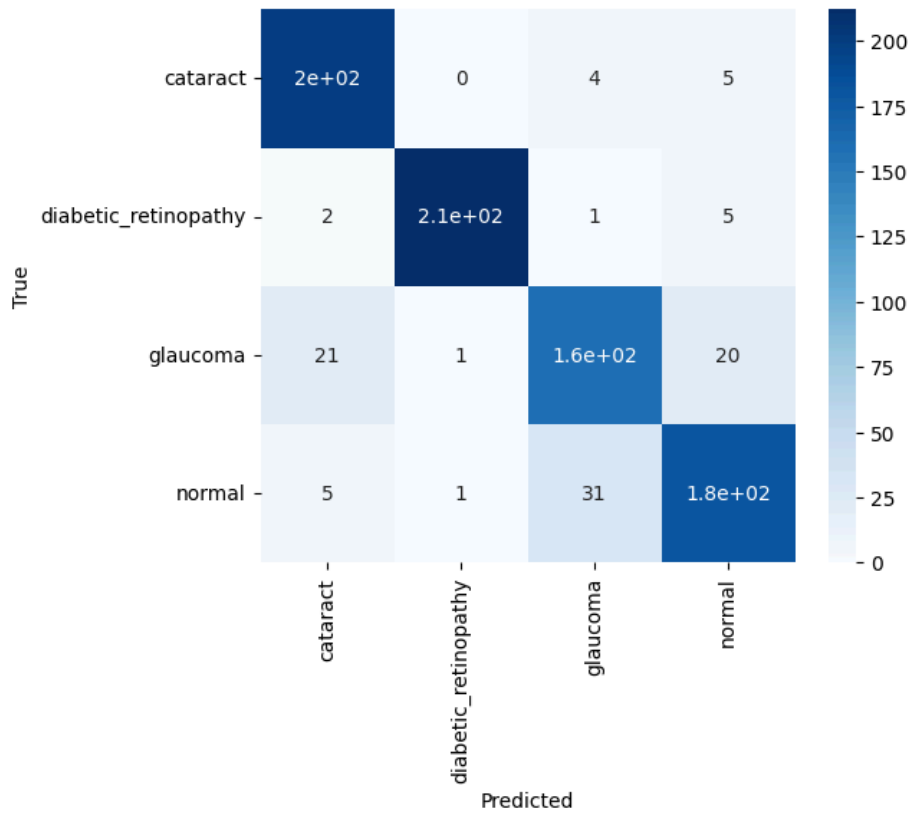
```
cm = confusion_matrix(y_true, y_pred)

plt.figure(figsize=(6,5))

sns.heatmap(
    cm,
    annot=True,
    cmap="Blues",
    xticklabels=classes,
    yticklabels=classes
)

plt.xlabel("Predicted")
plt.ylabel("True")

plt.show()
```



Double-click (or enter) to edit