```
import os

import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
from PIL import Image

import torch
from torch.utils.data import DataLoader, Dataset,Subset
from torchvision import transforms , datasets
from sklearn.model_selection import train_test_split
from tqdm import tqdm
import timm
from sklearn.metrics import f1_score, accuracy_score



from google.colab import drive
drive.mount('/content/drive')
print(os.listdir('/content/drive/My Drive/Colab Notebooks/Aerial_Landscapes'))
```

```
Mounted at /content/drive
['Airport', 'Agriculture', 'City', 'Beach', 'Desert', 'Forest', 'Highway', 'Grassland', 'Lake', 'Mountain', 'Port', 'Parking', 'Residential', 'Ra
```

```
img_path ='/content/drive/My Drive/Colab Notebooks/Aerial_Landscapes'

transform = transforms.Compose([
        transforms.RandomHorizontalFlip(),
        transforms.RandomRotation(30),
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406],
                                            std=[0.229, 0.224, 0.225])
])


dataset_path = img_path


dataset = datasets.ImageFolder(root=dataset_path, transform=transform)

print("类别名称:", dataset.classes)
print("数据集大小:", len(dataset))

data_loader = DataLoader(dataset, batch_size=32, shuffle=True, num_workers=4)

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(torch.cuda.get_device_name())
torch.cuda.empty_cache()
```

```
类别名称: ['Agriculture', 'Airport', 'Beach', 'City', 'Desert', 'Forest', 'Grassland', 'Highway', 'Lake', 'Mountain', 'Parking', 'Port', 'Railway
数据集大小: 12000
Tesla T4
/usr/local/lib/python3.11/dist-packages/torch/utils/data/dataloader.py:624: UserWarning: This DataLoader will create 4 worker processes in total.
  warnings.warn(
```

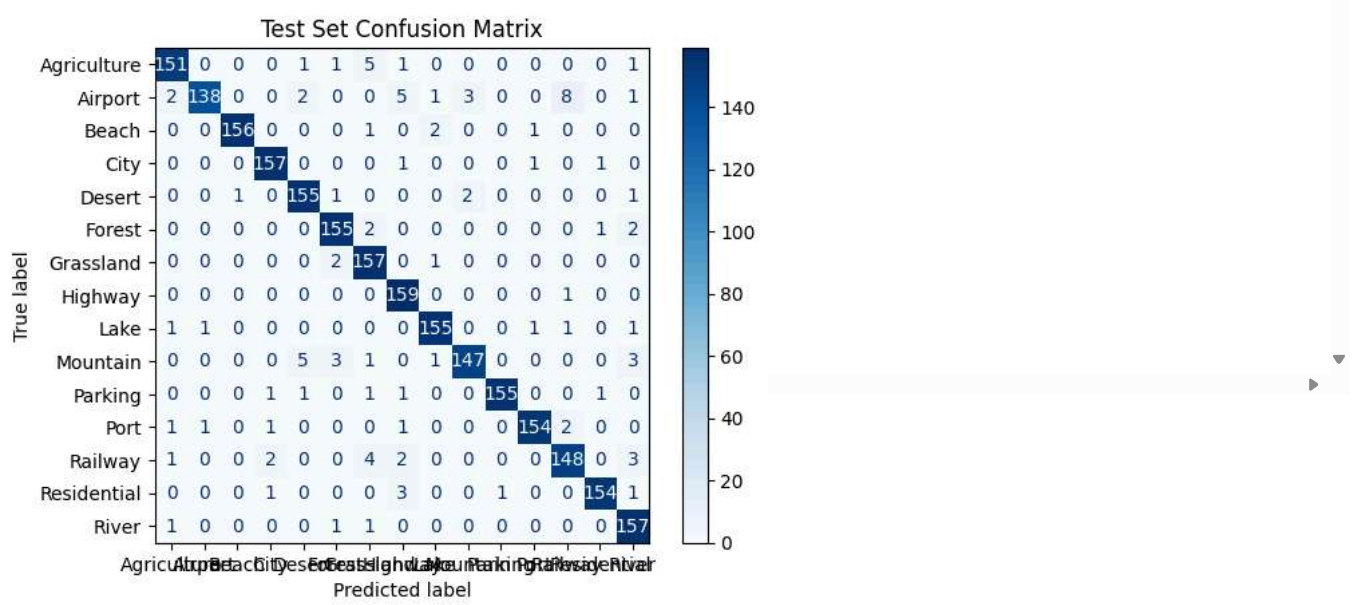
```
indices = np.arange(len(dataset))

y = np.array(dataset.targets)


train_idx, test_idx = train_test_split(indices, test_size=0.2, random_state=42, stratify=y)
print("train data:", len(train_idx))
print("test data:", len(test_idx))


train_dataset = Subset(dataset, train_idx)
test_dataset = Subset(dataset, test_idx)

batch_size = 32
train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True, num_workers=16, pin_memory=True)
test_loader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False, num_workers=16, pin_memory=True)


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

```
num_classes = len(dataset.classes)
model = timm.create_model('efficientnet_b0', pretrained=True, num_classes=num_classes)
model = model.to(device)


criterion = torch.nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=1e-3)


num_epochs = 10
for epoch in range(num_epochs):
    model.train()
    running_loss = 0.0
    for images, labels in tqdm(train_loader, desc=f"Epoch {epoch+1}"):
        images = images.to(device)
        labels = labels.to(device)

        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item() * images.size(0)

    epoch_loss = running_loss / len(train_dataset)
    print(f"Epoch {epoch + 1}/{num_epochs} - Loss: {epoch_loss:.4f}")

    model.eval()
    correct, total = 0, 0
    y_true, y_pred = [], []
    with torch.no_grad():
        for images, labels in test_loader:
            images = images.to(device)
            labels = labels.to(device)
            outputs = model(images)
            preds = outputs.argmax(dim=1)
            correct += (preds == labels).sum().item()
            total += labels.size(0)
            y_true.extend(labels.cpu().tolist())
            y_pred.extend(preds.cpu().tolist())
    accuracy = correct / total
    print(f"Test Accuracy: {accuracy:.4f}")
    print("=== Classification Report ===")
    print(classification_report(
        y_true,
        y_pred,
        target_names=dataset.classes,
        digits=4
    ))
```

```
训练集样本数: 9600
测试集样本数: 2400
/usr/local/lib/python3.11/dist-packages/torch/utils/data/dataloader.py:624: UserWarning: This DataLoader will create 16 worker processes in to
  warnings.warn(
Epoch 1: 100%|██████████| 300/300 [00:56<00:00,  5.27it/s]Epoch 1/10 - Loss: 0.4906

Test Accuracy: 0.9408
=== Classification Report ===
              precision    recall  f1-score   support

 Agriculture     0.9671    0.9187    0.9423       160
     Airport     0.9346    0.8938    0.9137       160
       Beach     0.9244    0.9938    0.9578       160
        City     0.9455    0.9750    0.9600       160
      Desert     0.9608    0.9187    0.9393       160
      Forest     0.9675    0.9313    0.9490       160
   Grassland     0.9235    0.9812    0.9515       160
     Highway     0.9850    0.8187    0.8942       160
        Lake     0.9603    0.9062    0.9325       160
    Mountain     0.9728    0.8938    0.9316       160
     Parking     0.9349    0.9875    0.9605       160
        Port     0.9811    0.9750    0.9781       160
     Railway     0.8112    0.9938    0.8933       160
 Residential     0.9815    0.9938    0.9876       160
       River     0.9085    0.9313    0.9198       160

    accuracy                         0.9408      2400
   macro avg     0.9439    0.9408    0.9407      2400
weighted avg     0.9439    0.9408    0.9407      2400

Epoch 2:   0%|          | 0/300 [00:00<?, ?it/s]/usr/local/lib/python3.11/dist-packages/torch/utils/data/dataloader.py:624: UserWarning: This
  warnings.warn(
Epoch 2: 100%|██████████| 300/300 [00:57<00:00,  5.18it/s]Epoch 2/10 - Loss: 0.1741

Test Accuracy: 0.9463
=== Classification Report ===
```

	precision	recall	f1-score	support
Agriculture	0.9444	0.9563	0.9503	160
Airport	0.9494	0.9375	0.9434	160
Beach	0.9693	0.9875	0.9783	160
City	0.8541	0.9875	0.9159	160
Desert	0.9865	0.9125	0.9481	160
Forest	0.9796	0.9000	0.9381	160
Grassland	0.8870	0.9812	0.9318	160
Highway	0.9812	0.9812	0.9812	160
Lake	0.9655	0.8750	0.9180	160
Mountain	0.9317	0.9375	0.9346	160
Parking	0.9576	0.9875	0.9723	160
Port	0.9737	0.9250	0.9487	160
Railway	0.9796	0.9000	0.9381	160
Residential	0.9815	0.9938	0.9876	160
River	0.8869	0.9313	0.9085	160
accuracy			0.9463	2400
macro avg	0.9485	0.9463	0.9463	2400
weighted avg	0.9485	0.9463	0.9463	2400

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
all_preds, all_labels = [], []
model.eval()
with torch.no_grad():
    for images, labels in test_loader:
        images, labels = images.to(device), labels.to(device)
        logits = model(images)
        preds = logits.argmax(dim=1)
        all_preds.extend(preds.cpu().numpy())
        all_labels.extend(labels.cpu().numpy())

cm = confusion_matrix(all_labels, all_preds)
disp = ConfusionMatrixDisplay(cm, display_labels=dataset.classes)
disp.plot(cmap=plt.cm.Blues)
plt.title("Test Set Confusion Matrix")
plt.show()
```



开始借助 AI 编写或生成代码。