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

```
Mounted at /content/drive
```

Download the folder and extract the files

- Run this immediately when you start the test.
- Once the execution is successful, there will be train.csv, test.csv and images folder.

```
!pip install gdown
!gdown --id 1AvFmitLUYqFGVSVUTN67R17he8mnq9Go
!unzip HV-AI-2024.zip
!rm -rf /content/__MACOSX
!mv /content/HV-AI-2024/* /content/
!rm -rf /content/HV-AI-2024
!rm /content/HV-AI-2024.zip
!rm -rf /content/sample_data
from google.colab import output
output.clear()
```

```
import pandas as pd
import numpy as np
import os
import torch
from torch.utils.data import Dataset, DataLoader
from torchvision import transforms, models
import torch.nn as nn
import torch.optim as optim
from PIL import Image
import requests
```

```
# Helper function to send results for evaluation
def send_results_for_evaluation(name, csv_file, email):
    url = "http://43.205.49.236:5050/inference"
    files = {'file': open(csv_file, 'rb')}
    data = {'email': email, 'name': name}
    response = requests.post(url, files=files, data=data)
    return response.json()
```

```
class CustomImageDataset(Dataset):
    def __init__(self, csv_file, img_dir, transform=None):
        self.img_labels = pd.read_csv(csv_file)
        self.img_dir = img_dir
        self.transform = transform

    def __len__(self):
        return len(self.img_labels)

    def __getitem__(self, idx):
        img_path = os.path.join(self.img_dir, self.img_labels.iloc[idx, 0])
        image = Image.open(img_path).convert('RGB')
        label = int(self.img_labels.iloc[idx, 1])

        if self.transform:
            image = self.transform(image)

        return image, label

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

train_dataset = CustomImageDataset(csv_file='/content/train.csv', img_dir='/content', transform=transform)
train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True, num_workers=4)
```

```
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:558: UserWarning: This DataLoader will create 4 w
  warnings.warn(_create_warning_msg(
```

```
# Step 3: Model Initialization and Training
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print(device)
```

```
def train_model(model, num_epochs=30):
```

```
def train_model(model, num_epochs=30):
    model = model.to(device)
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.Adam(model.parameters(), lr=0.001)

    for epoch in range(num_epochs):
        model.train()
        running_loss = 0.0
        for inputs, labels in train_loader:
            inputs = inputs.to(device)
            labels = labels.to(device)

            optimizer.zero_grad()
            outputs = model(inputs)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

            running_loss += loss.item()

        print(f"Epoch {epoch+1}/{num_epochs}, Loss: {running_loss/len(train_loader)}")

    return model

# Initialize and train models
model_resnet18 = models.resnet18(pretrained=True)
num_features = model_resnet18.fc.in_features
model_resnet18.fc = nn.Linear(num_features, 200)
model_resnet18 = train_model(model_resnet18)

model_vgg16 = models.vgg16(pretrained=True)
model_vgg16.classifier[6] = nn.Linear(4096, 200)
model_vgg16 = train_model(model_vgg16)

model_efficientnet_b0 = models.efficientnet_b0(pretrained=True)
num_features = model_efficientnet_b0.classifier[1].in_features
model_efficientnet_b0.classifier[1] = nn.Linear(num_features, 200)
model_efficientnet_b0 = train_model(model_efficientnet_b0)
```

```
Epoch 7/30, Loss: 5.302143665070229
Epoch 8/30, Loss: 5.302409473885882
Epoch 9/30, Loss: 5.301917748248323
Epoch 10/30, Loss: 5.303339539690221
Epoch 11/30, Loss: 5.302336984492363
Epoch 12/30, Loss: 5.3012469119214
Epoch 13/30, Loss: 5.301356536276797
```

```
Epoch 21/30, Loss: 0.09723124459094 93
Epoch 22/30, Loss: 0.10893162868500865
Epoch 23/30, Loss: 0.1063529466120328
Epoch 24/30, Loss: 0.14852494604518993
Epoch 25/30, Loss: 0.09276428017913899
Epoch 26/30, Loss: 0.056883756584671145
Epoch 27/30, Loss: 0.09211438953222588
Epoch 28/30, Loss: 0.09344004776992618
Epoch 29/30, Loss: 0.11793002773799557
Epoch 30/30, Loss: 0.09210252421759466
```

```
# Step 4: Model Inference
class CustomTestImageDataset(Dataset):
    def __init__(self, csv_file, img_dir, transform=None):
        self.img_labels = pd.read_csv(csv_file)
        self.img_dir = img_dir
        self.transform = transform

    def __len__(self):
        return len(self.img_labels)

    def __getitem__(self, idx):
        img_path = os.path.join(self.img_dir, self.img_labels.iloc[idx, 0])
        image = Image.open(img_path).convert('RGB')
        if self.transform:
            image = self.transform(image)
        return image, self.img_labels.iloc[idx, 0]

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

test_dataset = CustomTestImageDataset(csv_file='/content/test.csv', img_dir='/content', transform=test_transform)
test_loader = DataLoader(test_dataset, batch_size=32, shuffle=False, num_workers=4)

def evaluate_model(model):
    model.eval()
    predictions = []
    with torch.no_grad():
        for inputs, paths in test_loader:
            inputs = inputs.to(device)
            outputs = model(inputs)
            probs = nn.functional.softmax(outputs, dim=1)
            confidence, preds = torch.max(probs, 1)
            for path, pred, conf in zip(paths, preds, confidence):
                predictions.append({'path': path, 'predicted_label': pred.item(), 'confidence_score': conf.item()})

    return predictions

# Evaluate all models
predictions_resnet18 = evaluate_model(model_resnet18)
predictions_vgg16 = evaluate_model(model_vgg16)
predictions_efficientnet_b0 = evaluate_model(model_efficientnet_b0)
```

```
/usr/local/lib/python3.10/dist-packages/torch/utils/data/dataloader.py:558: UserWarning: This DataLoader will create 4 w
  warnings.warn(_create_warning_msg(
```

```
best_predictions = predictions_efficientnet_b0
```

```
best_predictions_df = pd.DataFrame(best_predictions)
best_predictions_df.to_csv('/content/predictions.csv', index=False)
```

```
send_results_for_evaluation('Priyanka Dash', '/content/predictions.csv', 'pd8042@srmist.edu.in')
```

```
{'overall_accuracy (%)': 63.91094,
 'max_accuracy_class': 27,
 'max_accuracy (%)': 100.0,
 'min_accuracy_class': 48,
 'min_accuracy (%)': 20.0}
```

Start coding or generate with AI.