期末報告

四資管AI二A-B11123224-余雨芹

報告連結

主題

練習 kaggle - Dogs vs. Cats (Pytorch)

↳ 程式 貓咪品種辨識

解決問題

- 寵物識別：在動物收容所或失蹤寵物的情況下，這樣的辨識器可以幫助識別是否是貓或狗，從而加快尋找遺失寵物或識別動物種類。
- 獲取數據進行研究：幫助研究人員收集大量關於貓和狗的圖像數據，進行行為、品種、健康等方面的研究。
- 娛樂軟體：在應用程式上，可以自動辨識用戶上傳的貓狗圖像，以及篩選加上相應的濾鏡、貼圖等。
- 家庭娛樂及教育：用於開發貓狗相關的遊戲或者互動，增加家庭娛樂的樂趣。

使用的技術

Pytorch

在程式碼中，使用了 PyTorch 的 torch 和 torchvision 模組來處理數據、構建和訓練模型

```
!pip install torch torchvision


!wget http://www.robots.ox.ac.uk/~vgg/data/pets/data/images.tar.gz
!wget http://www.robots.ox.ac.uk/~vgg/data/pets/data/annotations.tar.gz


!tar -xvf images.tar.gz
!tar -xvf annotations.tar.gz


import os
print(os.listdir('images'))
print(os.listdir('annotations'))
```

```
import torch
from torchvision import datasets, transforms
from torch.utils.data import DataLoader, random_split

transform = transforms.Compose([
        transforms.Resize((150, 150)),
        transforms.ToTensor(),
        transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])

full_dataset = datasets.ImageFolder('organized_cats', transform=transform)
train_size = int(0.8 * len(full_dataset))
val_size = len(full_dataset) - train_size
train_dataset, val_dataset = random_split(full_dataset, [train_size, val_size])

train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=32, shuffle=False)
```

1. 定義數據增強和標準化操作。
2. 加載並處理圖像數據集。
3. 將數據集隨機分割為訓練集和驗證集。
4. 創建 DataLoader 以便在訓練和驗證過程中批量加載數據。

使用的技術

CNN架構

定義了一個基於 ResNet18 的卷積神經網絡（CNN）模型。並修改了其最後一層以適應特定分類任務。

```
import torch.nn as nn
import torch.optim as optim
from torchvision import models

class CNNModel(nn.Module):
        def __init__(self, num_classes):
                super(CNNModel, self).__init__()
                self.cnn = models.resnet18(pretrained=True)
                num_ftrs = self.cnn.fc.in_features
                self.cnn.fc = nn.Linear(num_ftrs, num_classes)

        def forward(self, x):
                x = self.cnn(x)
                return x

num_classes = len(cat_breeds)
model = CNNModel(num_classes)

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

criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

1. 使用預訓練的 ResNet18 模型，並將其最後一層替換為適應我們分類任務的全連接層。
2. 初始化模型，並將其移動到可用的設備（GPU 或 CPU）。
3. 設置損失函數和優化器，以便在訓練過程中更新模型參數。

訓練模型

```python
import time

num_epochs = 5

for epoch in range(num_epochs):
    start_time = time.time()

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

        optimizer.zero_grad()

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

        running_loss += loss.item()

    epoch_time = time.time() - start_time
    print(f"Epoch {epoch+1}, Loss: {running_loss/len(train_loader)}, Time: {epoch_time:.2f} seconds")

    model.eval()
    correct = 0
    total = 0
    with torch.no_grad():
        for inputs, labels in val_loader:
            inputs, labels = inputs.to(device), labels.to(device)
            outputs = model(inputs)
            _, predicted = torch.max(outputs, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

    print(f"Validation Accuracy: {100 * correct / total}%")
```

```
Epoch 1, Loss: 0.1262036571684091, Time: 692.76 seconds
Validation Accuracy: 67.89115646258503%
Epoch 2, Loss: 0.13428654702653622, Time: 689.74 seconds
Validation Accuracy: 67.82312925170068%
Epoch 3, Loss: 0.12975623016245663, Time: 688.56 seconds
Validation Accuracy: 67.34693877551021%
Epoch 4, Loss: 0.12841779678685428, Time: 686.61 seconds
Validation Accuracy: 68.16326530612245%
Epoch 5, Loss: 0.12741945325599416, Time: 677.71 seconds
Validation Accuracy: 68.0952380952381%
```

1. 將模型設置為訓練模式。
2. 迭代訓練集，進行前向傳播、計算損失、反向傳播和參數更新。
3. 計算並打印每個 epoch 的訓練時間和平均損失。
4. 將模型設置為驗證模式。
5. 迭代驗證集，計算並打印準確率。

模型的保存

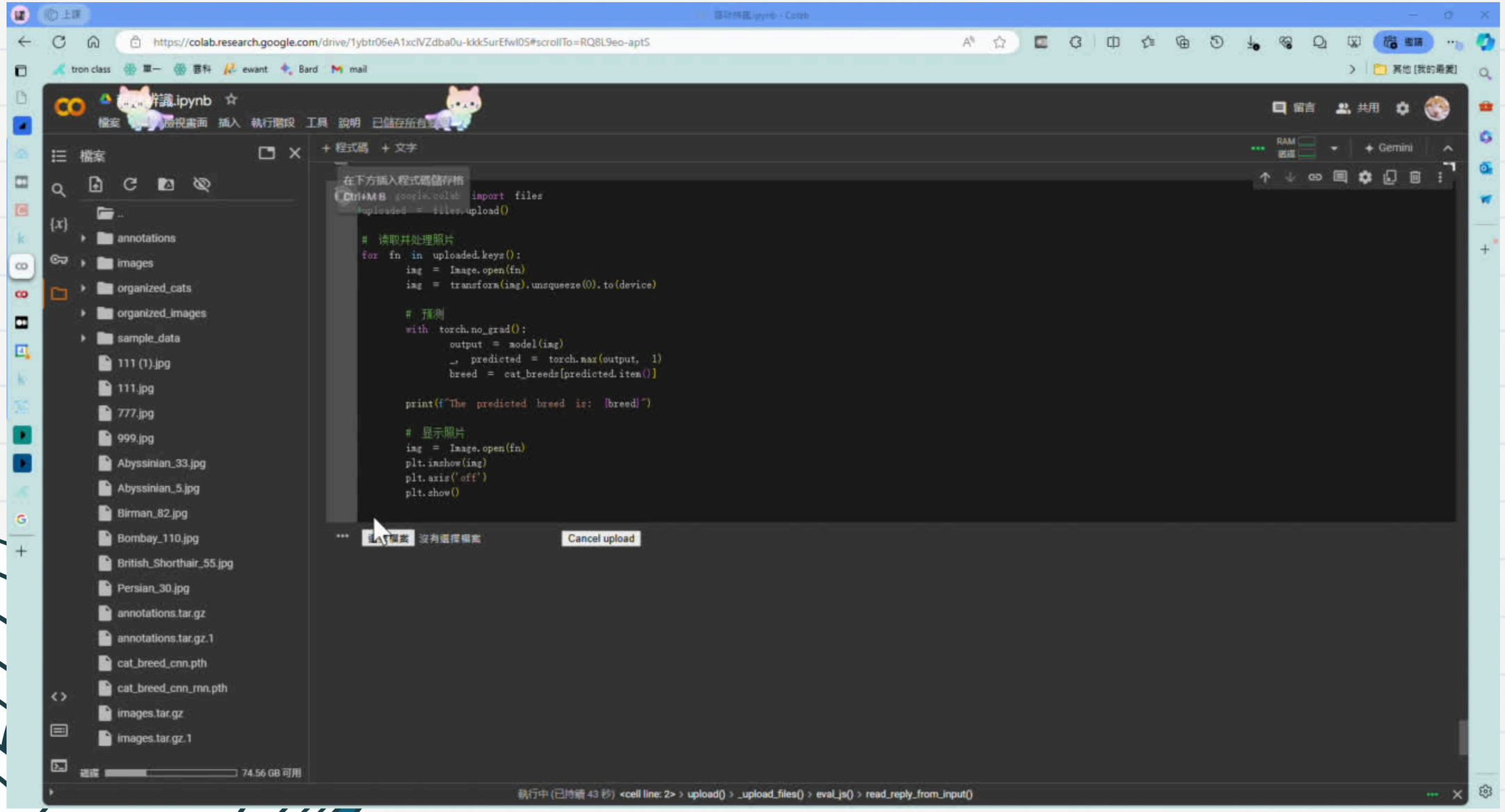
```
torch.save(model.state_dict(),  'cat_breed_cnn.pth')

model  =  CNNModel(num_classes)
model.load_state_dict(torch.load('cat_breed_cnn.pth'))
model.to(device)
model.eval()
```

1.保存訓練好的模型參數，方便以後加載使用。
2.創建一個新的模型實例，並從保存的文件中加載模型參數。
3.將模型移動到指定的設備上，並設置為推理模式。

預測結果

```python
from PIL import Image
from google.colab import files
import matplotlib.pyplot as plt
uploaded = files.upload()


for fn in uploaded.keys():
    img = Image.open(fn)
    img = transform(img).unsqueeze(0).to(device)

    with torch.no_grad():
        output = model(img)
        _, predicted = torch.max(output, 1)
        breed = cat_breeds[predicted.item()]

    print(f"The predicted breed is: {breed}")


    img = Image.open(fn)
    plt.imshow(img)
    plt.axis('off')
    plt.show()
```

預測結果


https://colab.research.google.com/drive/1ybtr06eA1xclVZdba0u-kkk5urEfwI0S#scrollTo=RQ8L9eo-aptS
檔案
annotations
images
organized_cats
organized_images
sample_data
111 (1).jpg
111.jpg
777.jpg
999.jpg
Abyssinian_33.jpg
Abyssinian_5.jpg
Birman_82.jpg
Bombay_110.jpg
British_Shorthair_55.jpg
Persian_30.jpg
annotations.tar.gz
annotations.tar.gz.1
cat_breed_cnn.pth
cat_breed_cnn_rnn.pth
images.tar.gz
images.tar.gz.1
74.56 GB 可用
+ 程式碼 + 文字
Gemini
在下方插入程式碼儲存格
Ctrl+M B
from google.colab import files
uploaded = files.upload()
# 读取并处理照片
for fn in uploaded.keys():
img = Image.open(fn)
img = transform(img).unsqueeze(0).to(device)
# 预测
with torch.no_grad():
output = model(img)
_, predicted = torch.max(output, 1)
breed = cat_breeds[predicted.item()]
print(f"The predicted breed is: {breed}")
# 显示照片
img = Image.open(fn)
plt.imshow(img)
plt.axis('off')
plt.show()
Cancel upload
執行中 (已持續 43 秒) <cell line: 2> › upload() › _upload_files() › eval_js() › read_reply_from_input()

資料來源

CNN實作Kaggle貓狗影像辨識(Pytorch)

Oxford-IIIT寵物數據集

ChatGPT

Thank's For Watching