十種動物識別模型

洪學誠、陳冠霖
許子衡
資訊工程系

摘要

我們設計一個基於卷積神經網絡（CNN）的模型，用於區分十種不同的動物。透過深度學習技術，我們嘗試建立一個高效且準確的分類系統，以提高動物識別的準確性和效率。我們使用了大量的動物影像資料進行訓練和測試，最終期望能夠應用此模型於生態研究、自然保育等領域。

1. 前言

近年來，隨著深度學習技術的發展，卷積神經網絡（CNN）在影像分類領域取得了顯著的成就。動物的自然多樣性使得動物分類成為一個具有挑戰性的問題。本研究旨在應用 CNN 模型，通過訓練一個能夠準確區分十種動物的系統，來解決這一問題。這將有助於提高動物識別的效率，同時為生態學和自然保育等領域提供有價值的工具。

2. 背景

動物分類一直是生物學和生態學研究的一個重要課題。然而，傳統的分類方法往往需要大量的人力和時間，且受限於專業知識。隨著影像識別技術的進步，深度學習模型成為自動分類的強大工具。卷積神經網絡通過學習特徵，能夠在大型數據集上實現高度準確的分類，這使得其在動物分類任務上具有巨大的應用潛力。

3. 動機及目的

我們的動機源於傳統動物分類方法的局限性，以及深度學習在影像識別領域的卓越表現。我們希望利用現代技術提高動物分類的效率，同時降低人力成本。透過訓練一個準確的 CNN 模型，我們的目標是實現在不同場景和條件下對十種動物的精確識別，為生態學、自然保育等領域的研究提供強有力的支持。

4. 執行方法及步驟

在這份專案中我們設計了一個模型訓練的檔案(CNN.py)和一個用來將圖片輸入模型預測的前端(app.py)。

使用資料集:
https://www.kaggle.com/datasets/alessiocorrado99/animals10

一、模型訓練檔案(CNN.py)

```python
# 定義 collate_fn 函數
def collate_fn(batch):
    # 從批次中提取圖像和類別名稱
    images, class_names = zip(*batch)

    # 將圖像調整為相同的大小，並傳遞 antialias 參數
    resized_images = [Resize((100, 100), antialias=True)(image) for image in images]

    # 將調整後的圖像和類別名稱返回
    return torch.stack(resized_images), class_names
```

先設計 collaew_fn 函數來定義輸入圖片的格式。

```python
# 定義數據集類
class CustomDataset(Dataset):
    def __init__(self, data_dir, transform=None):
        self.dataset = ImageFolder(root=data_dir, transform=transform)
        self.class_names = self.dataset.classes

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

    def __getitem__(self, idx):
        img, label = self.dataset[idx]
        class_name = self.class_names[label]
        return img, class_name  # 返回圖像和類別名稱
```

設計 Custom Dataset 函數來將數據集根據資料夾名稱來建構訓練用的類別。

```python
# 指定資料集路徑和轉換
custom_dataset = CustomDataset(data_dir='raw-img/', transform=ToTensor())

# 使用 DataLoader 進行批次載入，指定 collate_fn
data_loader = DataLoader(dataset=custom_dataset, batch_size=64, shuffle=True, collate_fn=collate_fn)
```

利用 Custom Dataset 來建立類別再將裡面的資料變成 collaew_fn 定義的圖片格式。

```python
class SimpleCNN(nn.Module):
    def __init__(self, num_classes=10):
        super(SimpleCNN, self).__init__()
        # 定義卷積層和池化層
        self.conv1 = nn.Conv2d(3, 16, kernel_size=5, stride=1, padding=2)
        self.relu = nn.ReLU()
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        self.conv2 = nn.Conv2d(16, 32, kernel_size=5, stride=1, padding=2)


        # 定義全連接層
        self.fc1 = nn.Linear(32 * 25 * 25, 512)  # 注意更新這裡的輸入維度
        self.fc2 = nn.Linear(512, num_classes)

    def forward(self, x):
        # 卷積層1 + 激活函數 + 池化層
        x = self.pool(self.relu(self.conv1(x)))
        # 卷積層2 + 激活函數 + 池化層
        x = self.pool(self.relu(self.conv2(x)))
        # 新增的卷積層3 + 激活函數 + 池化層


        # 展平
        x = x.view(-1, 32 * 25 * 25)  # 注意更新這裡的輸入維度
        # 全連接層1 + 激活函數
        x = self.relu(self.fc1(x))
        # 全連接層2（輸出層）
        x = self.fc2(x)

        return x
```

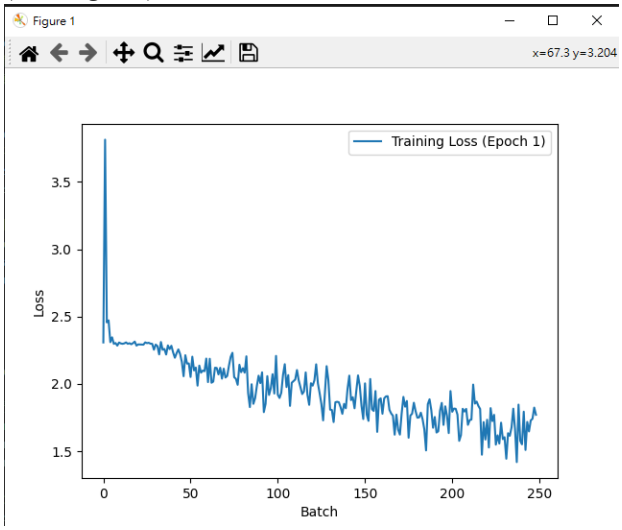
設計 CNN 模型，模型使用了 2 個捲積層和 2 個池化層和 2 個全連接層。

```python
# 檢查是否存在已經訓練好的模型檔案
model_save_path = "simple_model.pth"
if os.path.exists(model_save_path):
    # 如果存在模型檔案，加載模型參數
    model.load_state_dict(torch.load(model_save_path))
    print("Model loaded from existing file.")
    sample_indices = random.sample(range(len(custom_dataset)), 10)
    sample_images = [custom_dataset[i][0] for i in sample_indices]
    sample_labels = [custom_dataset[i][1] for i in sample_indices]

    # 將圖像移到 GPU 上，並進行預測
    sample_images = torch.stack([Resize((100, 100), antialias=True)(image) for image in sample_images]).to(device)
    predictions = model(sample_images)
        # 顯示預測結果
    for i in range(10):
        plt.subplot(5, 2, i+1)
        plt.imshow(sample_images[i].permute(1, 2, 0).cpu().numpy())
        predicted_label = custom_dataset.class_names[predictions[i].argmax().item()]
        plt.title(f"Actual Label: {sample_labels[i]}\nPredicted Label: {predicted_label}")
        plt.axis('off')
        plt.show()
```

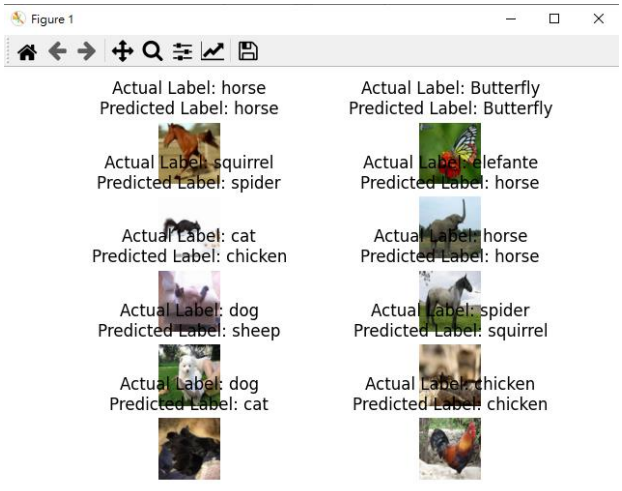
判斷模型是否存在，是的話就拿測試用的資料丟入模型去判斷。

```
else:
    # 如果不存在模型檔案，進行訓練
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.Adam(model.parameters(), lr=0.001)

    # 儲存訓練損失的列表
    for epoch in range(num_epochs):
        train_losses = []  # 清除訓練損失列表
        model.train()  # 設置模型為訓練模式

        # 迭代資料集的每個批次
        for batch in data_loader:
            # 獲取批次資料，並將資料移到 GPU 上
            images, class_names = batch[0].to(device), batch[1]
            labels = torch.tensor([custom_dataset.class_names.index(class_name) for class_name in class_names]).to(device)

            # 在每個批次上進行訓練
            optimizer.zero_grad()
            outputs = model(images)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

            # 儲存訓練損失
            train_losses.append(loss.item())

    # 隨機選擇四張圖像並顯示預測結果
    sample_indices = random.sample(range(len(custom_dataset)), 10)
    sample_images = [custom_dataset[i][0] for i in sample_indices]
    sample_labels = [custom_dataset[i][1] for i in sample_indices]

    # 將圖像移到 GPU 上，並進行預測
    sample_images = torch.stack([Resize((100, 100), antialias=True)(image) for image in sample_images]).to(device)
    predictions = model(sample_images)
```

諾模型不存在則開始訓練，將資料集輸入模型進行訓練訓練十次每一次訓練都使用 Adam 優化器學習率 (learning rate) 設定為 0.001。

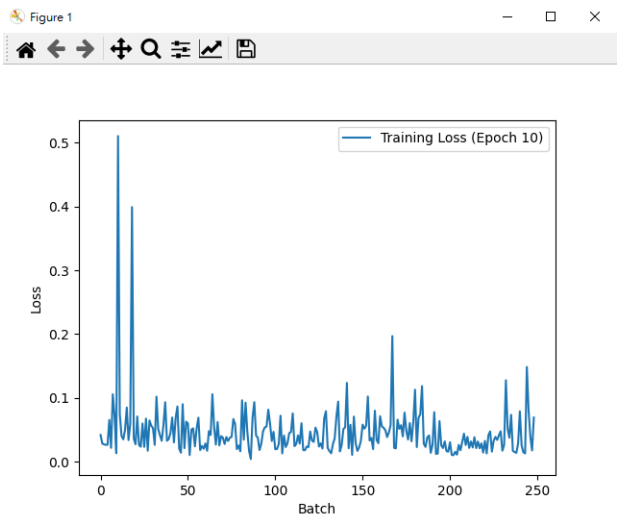


圖一、第一次訓練 loss 率

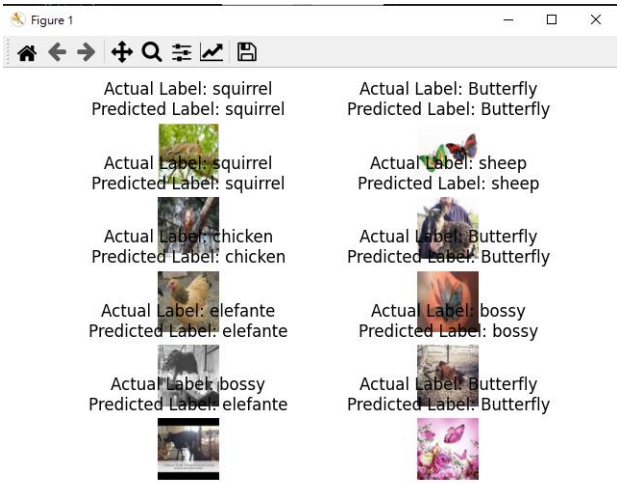


圖二、第一次猜測結果

繼續訓練剩餘 9 次



圖三、第十次訓練 loss 率



圖四、第十次猜測結果

```
    # 儲存訓練好的模型
    torch.save(model.state_dict(), model_save_path)
    print(f"Model saved at: {model_save_path}")
```

儲存訓練好的模型 simple_model.pth

二、前端檔案(app.py)

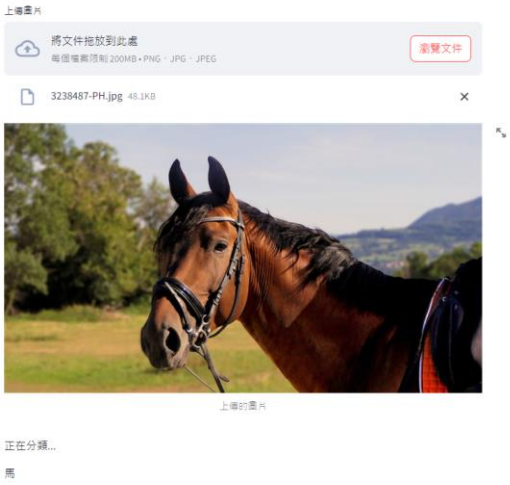
```
def do_prediction(fileIOObj):
    pil_image = Image.open(fileIOObj).convert('RGB')
    # 將圖像轉換為張量並調整大小以適應我們訓練的模型大小
    img_loader = transforms.Compose([
        transforms.Resize((100, 100)),
        transforms.ToTensor()
    ])
    ts_image = img_loader(pil_image).float()
    ts_image.unsqueeze_(0)
    outputs = net(ts_image)
    _, predicted = torch.max(outputs.data, 1)
    # 預測是一個列表，因此我們必須獲取第一個元素
    return classes[predicted[0]]
```

設計 do_prediction 函數來定義輸入圖片的格式

```python
PATH = './simple_model.pth'

classes = ('boosy', 'Butterfly', 'cat', 'chicken', 'dog', 'elephants', 'horse', 'sheep', 'spider', 'squirrel')

uploaded_file = st.file_uploader("上傳圖像", type=["png", "jpg", "jpeg"])

if uploaded_file is not None:
    st.image(uploaded_file, caption='上傳的圖像', use_column_width=True)
    st.write("")
    st.write("正在分類...")

    net = SimpleCNN()
    net.load_state_dict(torch.load(PATH))

    for parameter in net.parameters():
        parameter.requires_grad = False

    prediction = do_prediction(uploaded_file)
    st.write(prediction)
```

利用 streamlit 設計個網頁

將圖片利用網頁輸入模型進行預測



圖五、預測馬

圖六、預測松鼠

參考文獻

1. https://pytorch.org/tutorials/beginner/basics/quickstart_tutorial.html
2. https://streamlit.io/
3. https://www.kaggle.com/datasets/alessiocorrado99/animals10
4. https://chat.openai.com/