台師大 613k0007c 余品誼

這個作業的主要任務是二元語義分割，目標是將每個圖像中的像素分類為前（感興趣的物體）或背景。在這個實驗中，你會使用 0xford-IIIT Pet Dataset，該數據集包括貓和狗的圖片，並且每張圖片都有像素級別的標註來識別出寵物區域

I. 實現細節：

在這個專案中，我從頭開始實現了訓練、評估和推理代碼。訓練過程包括載入資料集、執行資料增強，並將數據傳遞到模型中。我使用了自定義的訓練迴圈，並使用特定的損失函數（BCE + Dice Loss）來優化模型。評估函數計算了 Dice Score，這是用來評估模型在驗證集上的表現的指標。

使用兩個模型 UNet 和 ResNet34_Unet 做訓練，並用一張完全沒看過的圖片利用訓練最好的模型做驗證。

在訓練過程中，我使用了 Adam 優化器，並使用學習率調度器來動態調整學習率。我根據驗證集的 Dice Score 保存了最佳模型。

```
# 定義損失函數
critertion = nn.BCEWithLogitsLoss()
optimizer = optim.Adam(model.parameters(), lr=args.learning_rate)
```

1. 模型架構：

A. UNet 架構：

UNet 架構包含一個收縮的編碼器路徑和一個擴展的解碼器路徑。編碼器在不同的解析度下捕捉特徵圖，解碼器將特徵圖上採樣至原始輸入大小。使用跳層連接來將編碼器的特徵圖與解碼器進行串接，從而保留空間信息。

UNet 的基本單元，包含兩個 3×3 卷積層，利用 ReLU 函數增加非線性特徵。

```
self.conv = nn.Sequential(
    #3x3捲積
    nn.Conv2d(in_channels, out_channels, kernel_size=3, padding=1),
    #加速收斂並穩定訓練
    nn.BatchNorm2d(out_channels),
    #加入非線性特性
    nn.ReLU(inplace=True),
    nn.Conv2d(out_channels, out_channels, kernel_size=3, padding=1),
    nn.BatchNorm2d(out_channels),
    nn.ReLU(inplace=True)
)
```

四層 Encoder 和四層 Decoder。

```
#Encoder
#進行下採樣，保留編碼特徵
enc1 = self.enc1(x)
enc2 = self.enc2(self.pool(enc1))
enc3 = self.enc3(self.pool(enc2))
enc4 = self.enc4(self.pool(enc3))

#Bottleneck
bottleneck = self.bottleneck(self.pool(enc4))

#Decoder
#ConvTranspose2d 上採樣
dec4 = self.up4(bottleneck)
#將 Encoder 層的輸出與 Decoder 層的輸入拼接。
dec4 = torch.cat((enc4, dec4), dim=1)
dec4 = self.dec4(dec4)

dec3 = self.up3(dec4)
dec3 = torch.cat((enc3, dec3), dim=1)
dec3 = self.dec3(dec3)

dec2 = self.up2(dec3)
dec2 = torch.cat((enc2, dec2), dim=1)
dec2 = self.dec2(dec2)

dec1 = self.up1(dec2)
dec1 = torch.cat((enc1, dec1), dim=1)
dec1 = self.dec1(dec1)
```

B. ResNet34_UNet 架構：

使用 ResNet34 作為編碼器的骨幹。ResNet34 是一個擁有 34 層的深度殘差網絡，用在處理圖像識別任務。我使用 ResNet34 的層作為編碼器，並將跳層連接合併到解碼器中。解碼器使用轉置卷積層來上採樣特徵圖。我通過將對應的編碼器層與解碼器在每個層級進行串接來實現 U-Net 的架構。

U-Net 基本單元基本和以上一樣，差別在於第一層改成 nn.ConvTranspose2d 步長改為二。

```
nn.ConvTranspose2d(in_channels, out_channels, kernel_size = 3, stride = 2, padding=1, output_padding=1),
```

五層 Encoder

```
#encoder
enc1 = self.enc1(x)
enc2 = self.enc2(enc1)
enc3 = self.enc3(enc2)
enc4 = self.enc4(enc3)
enc5 = self.enc5(enc4)
```

Decoder 串接兩個模型使用 torch.nn.functional 對齊，實現跳層連接。

```
dec5 = self.dec5(enc5)
dec5 = F.interpolate(dec5, size=enc4.shape[2:], mode="bilinear", align_corners=False)

dec4 = self.dec4(torch.cat([dec5, enc4], dim=1))
dec4 = F.interpolate(dec4, size=enc3.shape[2:], mode="bilinear", align_corners=False)

dec3 = self.dec3(torch.cat([dec4, enc3], dim=1))
dec3 = F.interpolate(dec3, size=enc2.shape[2:], mode="bilinear", align_corners=False)

dec2 = self.dec2(torch.cat([dec3, enc2], dim=1))
dec2 = F.interpolate(dec2, size=enc1.shape[2:], mode="bilinear", align_corners=False)

dec1 = self.dec1(dec2)
```

C. 訓練設置與優化策略

我以批次大小 4 訓練模型，分別針對兩個模型訓練 200 個和 50 個 epoch。

使用的損失函數是二元交叉熵損失和 Dice 損失的組合，以處理數據集中的不平衡問題。

我使用了學習率調度器來調整學習率，並在訓練的後期進行減少，以幫助模型更好地收斂。

II. 數據預處理

在這個部分，我們設計了自定義的數據預處理方法，利用 Albumentations 進行數據增強：

1. 水平翻轉（Horizontal Flip）：隨機進行水平翻轉，概率為 50%。這有助於模型學會處理物體的對稱性。
2. 垂直翻轉（Vertical Flip）：隨機進行垂直翻轉，概率為 20%。這能讓模型學會處理圖像的上下翻轉，增加其多樣性。
3. 隨機旋轉 90 度（Random Rotate 90）：隨機旋轉圖像 90 度，概率為

30%。這對於學習圖像在不同方向下的特徵有很大幫助。

4. 平移縮放旋轉（ShiftScaleRotate）：這是一個複合變換，隨機對圖像進行平移、縮放和旋轉，具有較高的隨機性。這個操作的 shift_limit 設置為 0.1，scale_limit 設置為 0.2，rotate_limit 設置為 15，並且應用的概率為 50%。
5. 隨機亮度對比度調整（RandomBrightnessContrast）：隨機調整圖像的亮度和對比度，概率為 20%。這樣的變換能夠使模型更具穩健性，對不同光照條件下的圖像進行處理。
6. 高斯模糊（Gaussian Blur）：隨機對圖像進行高斯模糊，模糊範圍設置為 3，並且概率為 30%。這樣可以模擬在不同焦距和清晰度條件下的圖像。
7. 標準化（Normalize）：對圖像進行標準化處理，將像素值的均值設為 [0.5, 0.5, 0.5]，標準差設為 [0.5, 0.5, 0.5]，這有助於加速神經網絡的訓練過程，並且提高其收斂速度。
8. 轉換為 PyTorch Tensor：最終，所有變換後的圖像都會被轉換為 PyTorch Tensor，這樣可以直接餵入神經網絡中進行訓練。

```
transform = A.Compose([
    A.HorizontalFlip(p=0.5),
    A.VerticalFlip(p=0.2),
    A.RandomRotate90(p=0.3),
    A.ShiftScaleRotate(shift_limit=0.1, scale_limit=0.2, rotate_limit=15, p=0.5),
    A.RandomBrightnessContrast(p=0.2),
    A.GaussianBlur(blur_limit=3, p=0.3),
    A.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5]),
    ToTensorV2()  # 轉換為 PyTorch Tensor
])
```

III. 實驗結果分析

1. train.py

A. ResNet34_UNet 架構 (200 epoch)

訓練階段：

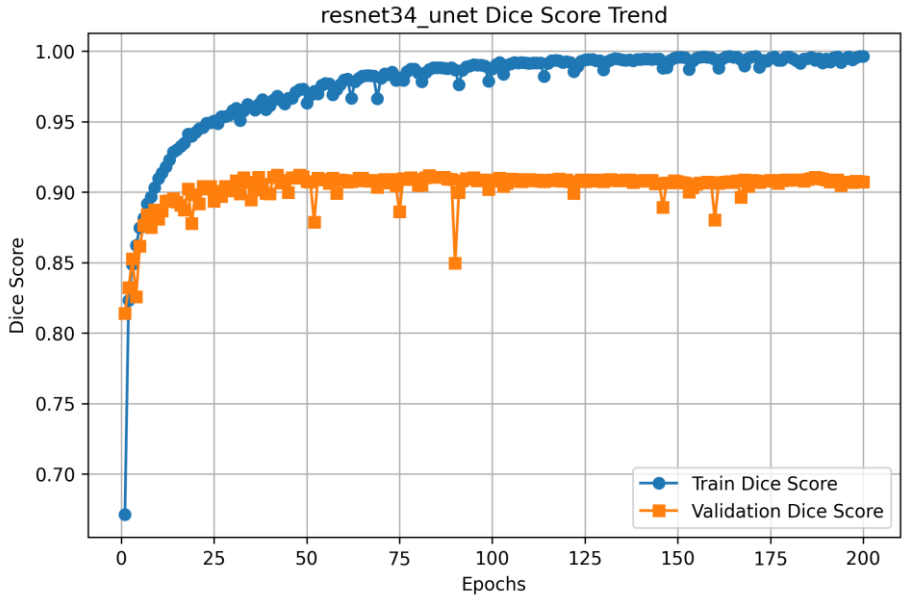
藍色線條表示訓練集的 Dice Score。從圖中可以看出，隨著訓練輪數的增加，訓練集的 Dice Score 快速提升並穩定在較高的值，最終接近 1.0。

訓練集的 Dice Score 幾乎上升，這表示模型對訓練數據的學習效果良好。

驗證階段：

橙色線條表示驗證集的 Dice Score。與訓練集的變化趨勢相比，驗證集的 Dice Score 在較高範圍內波動，並沒有像訓練集那樣持續提升。

驗證集的 Dice Score 在 0.85 到 0.90 之間波動，這顯示驗證集的表現沒有持續上升，可能存在過擬合的現象。



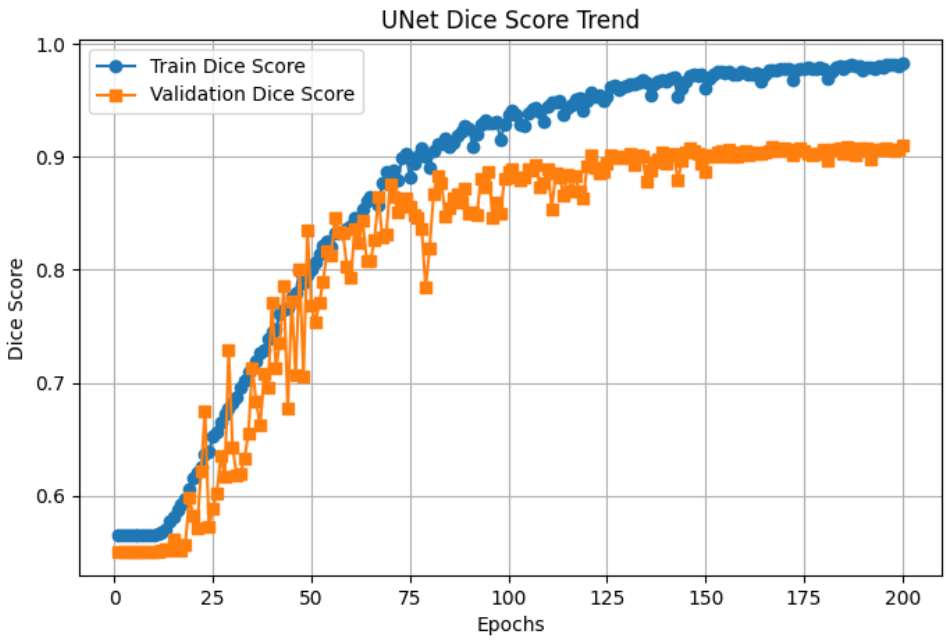
B. UNet 架構 (200 epoch)

訓練階段：

在 200 個 epoch 的訓練過程中，訓練集的 Dice 得分持續穩定上升，並接近 1，顯示出模型在訓練數據上擁有很強的學習能力。這表明模型正在有效地擬合訓練數據。

驗證階段：

驗證集的 Dice 得分也顯示出上升趨勢，但整體增長較為波動。在大約 40 到 50 個 epoch 之後，驗證集的得分進入了一個相對穩定的區間，並最終達到約 0.9。這表明模型在驗證集上逐步取得了穩定表現，但並未像訓練集那樣達到接近 1 的得分。



C. 小節：

ResNet34_UNet 雖然在訓練集和驗證集上表現更穩定，但其在驗證集上仍然存在一定的過擬合（較小的波動），這可能是因為模型較複雜，容易對訓練數據過度擬合。

UNet 則在驗證集上有較大震動，這意味著可能需要更好的正則化技巧來提高其穩定性。

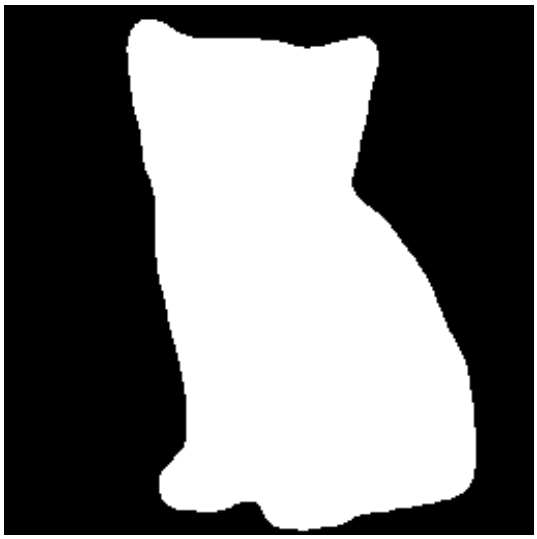
總結來說，ResNet34_UNet 在穩定性和泛化能力上表現較好，而 UNet 可能需要額外的調整來提高其在驗證集上的穩定性。

2. Inference.py

kitten-1.jpg

A. resnet34_unet 實驗結果：

resnet34_unet_200_best_model.pth



分析結果：預測的貓的輪廓清晰，這顯示出模型能夠有效地識別圖像中的物體，並且將其區分出來。這是二進制語義分割任務中的成功指標。

B. Unet

best_model1-200.pth

分析結果：整體來說，模型成功地將目標物體切分出來，但仍然存在一些雜訊或不需要的點散布在外圍，顯示出部分區域的分割精度有待提高。

IV. 執行步驟

A. 請先移到到 src 目錄下

B. 訓練命令和參數

python train.py --model unet --data_path ../dataset/oxford-iiit-pet --epochs 200 --batch_size 4 --learning_rate 0.0001 --download no

C. 推理命令

python inference.py --model resnet34_unet --model_path "../saved_models/best_model1-50res34.pth" --data_path "./kitten-1.jpg" --output_path "./output.png"

V. 討論

1. 模型性能比較：

根據訓練過程中的 Dice Score 指標，ResNet34_UNet 模型在較少的 epochs 中便達到了較高的訓練準確度，並且訓練集和驗證集的結果穩定。這表明 ResNet34_UNet 模型的深度學習能力相對較強，能夠更快

速地學習並找到準確的特徵。而 UNet 模型則在訓練過程中表現出較大的震盪，可能是因為它較為基礎的結構缺乏像 ResNet34 那樣的深層特徵學習能力。UNet 在較早期的訓練中存在較大的不穩定性，並且驗證集的 Dice Score 也顯示出不穩定的波動。

2. 推理效果

   在推理階段，兩個模型在分割結果上都有一定的表現，但 ResNet34_UNet 在推理過程中通常能夠更好地捕捉到物體的輪廓，而 UNet 雖然也能夠進行有效分割，但在某些較細的區域存在一些邊界模糊的情況，特別是在較為複雜或細節較多的圖像中，這些區域可能會被誤分為背景。

3. 模型未來優化策略

   A. 引入更深層的網絡結構，如 ResNet、DenseNet 等，來增強特徵學習能力。

   B. 在模型中增加更多的正則化技術，例如 Dropout 或數據增強，以防止過擬合。

   C. 調整學習率或使用更先進的優化器，如 AdamW，以更有效地尋找最優解。