REPORT

312552052鄭博元

Overview

• 這次的作業主要是實現並訓練一個深度學習模型,而過程中我們主 要採用binary semantic segmentation的方式來完成訓練,我們會先 從老師給予的py檔路徑下載Oxford-lll pet的數據集,分別使用unet與resnet34_unet完成訓練,訓練的部分我們採用交叉商來監督模型loss下降的狀況,最後將其與背景分離完成訓練以及評估,測試的動作,並求出功課要求之dice score.

Implement detail(1)

(Train.py)

- 先導入Rdam優化以及tqdm進度條還有matplotlib.pyplot做為圖相變換工具,剩餘為基本函式庫導入,一開始先藉由get device獲取訓練設備(cuda),並列印出內存使用情況,而後建立record/pics資料夾,get_transforms部分取圖像均值和標準差做歸一化
- 而image變換部分將圖像調整為256*256在做水平及垂直的機率翻轉最後將圖像轉為張量歸一化到[0,1],mask也是但少做了歸一化的動作並使用了插值法,而common transform使我們圖像在測試集驗證集得到一致結果,接著將數據保存成.npy檔,在進行加載訓練model之動作
- (預先寫好等等inference要call的definition)在visualize中,我們先設定評估模式,並且禁止梯度更新動作,從數據提取一個樣本並增加其維度使其可以符合模型輸入需求,將其預測並進行四捨五入,利用歸一化均值標準差調整形狀為(channel,1,1)已進行廣播運算,做返歸一化恢復原始色彩最後用plt繪製可視結果,將目錄創建(看是否有)並保存路徑命名
- 獲取gpu進行數據變化,接著創建數據集跟加載器,使用load model載入對應model,定義bce loss跟優化器rdam,訓練loss,dice,total sample並使用最佳dice機制更新best_score,最後保存至save_model中,train部分跑指定epoch數量並設置為訓練模式,將每輪更新之dice,loss印出並反覆更新模型參數,最後驗證模型並保存最佳的
- 解析args並call主函數做train(args)

Implement detail(1)

- (Inference.py)

  這裡引用radom用於設計隨機種子,使用torch.bankends.deterministic及benchmark鎖定cuda中特定操作,進行解析命令參數並在主函數執行,定義test之數據轉換調整圖像大小並用totensor縮放到0,1範圍內,加載測試數據集然後選擇架構,加載模型權重,最後印出測試可視圖

- (evaluate.py)

  evaluate中將model先設成evaluate模式然後初始化loss,dice跟sample,接著禁止梯度更新,遍歷所有data加載器,將圖像移至gpu並進行預測mask,使用criteriom計算loss(交叉商)然後依批次大小進行加權累加損失,最後批次大小累加後加權計算dice,並計算平均驗證損失,下方相同步驟只是將進度條換成test.

Implement detail(2)

- Unet

  rgb有三色且輸出只有灰色,因此設定為3,1,32(特徵)在unet的encode階段,每個feature都做一次block的卷積用於提取特徵,然後再用pool降低空間分辨率壓縮特徵圖大小,而經過同樣的四次動作到達bottle neck,到bottle neck後,對每一層做Convtranpose2d反卷積將空間分辨率加倍然後與對應之編碼器特徵圖做cat拼接(也就是def forward中在dec=up.convolution cat enc做的事情),而每一個得到之解碼塊在做block函數處理變成我們要得特徵圖,同樣的步驟進行四次會使空間信息恢復得很準確且保留解碼階段特徵信息,最終輸出1x1卷基層,所以out_channel設1為輸出的通道數(二元切割)且使用sigmoid因為其將輸出限制[0,1]適合切割前景,背景.

- Resnet

  首先我們從construct_residual_layer構建residual layer,其由多個residual units組成,而若輸出與輸入特徵圖不匹配我們會用downsample_layer匹配,residual units部分我們使用兩個convolution並shortcut connection連接輸入加到輸出保留低級特徵

  Forward part:在編碼器時,由剛開始的四個residual layer組成使我們通道數由64->128->256->512然後每個layer購艦多個residualunit做shortcut connection提取特徵經過四層生成bottleneck,而解碼階段,我們使用unet技巧構成up.convolution增加空間分辨率並讓特徵圖與編碼器中對應特徵圖concat,然後經過卷基層融合,最終1x1的卷基層從64channel減少至1並用sigmoid規劃到[0,1]範圍

Data Preprocessing

- How you preprocessed your data?
- 我調整了圖像和遮罩大小，所有輸入圖像和遮罩都被調整為統一的大小（256x256像素）。這樣做是為了確保輸入尺寸的一致性，除此之外，我還對數據進行了歸一化，歸一化步驟有助於穩定訓練過程，將像素值帶到相似的範圍，從而有利於梯度下降過程，還有就是轉換張量部分，我把圖像和遮罩轉為 PyTorch 張量以符合輸入要求
- What makes your method unique?
- 我的歸一化過程用的是均值和標準差是基於數據集的特徵專門計算的,而在掩碼處理時我有用近鄰差值法小心調整掩碼大小確保標籤分割完整,而我的預處理也能適應不同的模型架構,像這次訓練的unet,resnet34_unet都可以重複使用預處理步驟

Analyze on the experiment results

- What did you explore during the training process?
- 我使用了不同模型架構,我發現同樣的時間下resnet34_unet的訓練速度較為緩慢,且在復原圖像的dice score上也較unet低分,而為了使模型能提升泛化,我也增加了隨機水平和垂直翻轉來增強我的model適應能力,且我在驗證的過程中只會保留最佳的dice分數
- Found any characteristics of the data?
- 我發現遮罩在處理邊界時容易產生模糊,推測是特徵較複雜難以捕捉,而數據有不均衡的現象,也使得模型在預測時偏差值很明顯

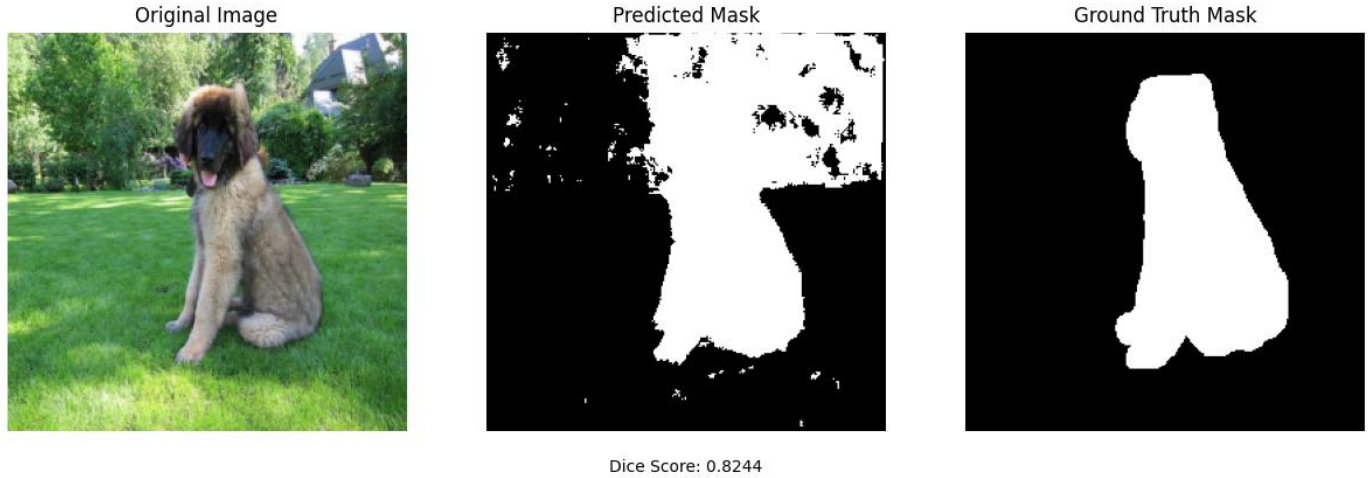
Execution command

- 首先我用python將資料集做載入至oxford-iiit-pet
- Training部分 --datapath抓取oxford-iiit-pet的資料集絕對位置

--epochs次數抓20(兩個都是),--model則調成unet or resnet34_unet

- record/pics中會有不同次訓練之結果 也有對應之pth在save_model
- Inference部分 -- model 輸入你想測試之pth,--datapath也是使用oxford-iiit-pet的資料絕對位置,--model_type則調成對應pth檔的unet or resnet34_unet

Inference score(DICE)



Discussion

- 我認為UNet 表現得比較好。UNet 的優勢在於其跳接連接（skip connections）能夠在上採樣過程中保留更多的細節信息，這對於圖像分割任務中特別重要。UNet 結構簡潔有效，適合處理需要高精度的分割任務

- 可以結合其他損失函數使model穩定性提高,或是可以探索更深層次的unet變體適應更大維度的尺寸,或是往半監督學習的方式去應用可以提高分割效果