IMAGE PROCESSING HW2 Report

P76134838 黃柏融

Submission Date: January 6, 2025

Contents

1 Outline

以下說明提交的程式碼檔案概要

ui.py

- for Demo
- Load Folder : 選擇 image dir，讀取後 Image Display 區域將顯示第一張圖片，並以 [index/Totol image number] 顯示目前的資料夾資料總數、目前影像名稱等等
- Previous, Next : 切換上/下一張影像，切換後會自動執行 Detection，在區域中以紅框標示
- Load label : 設定舟骨 label 資料夾，若有設定，且路徑中存在與當前影像相同名稱的 json，以綠色框標示
- load label2 : 設定 fracture label，只有在有設定舟骨 label 資料夾時才會生效，因為該 label 依賴舟骨 label 來將相對座標轉為絕對座標
- Detection : 將目前影像輸入模型運算。並計算 IoU, Accuracy, Precision, Recall 等指標 (因為我骨折檢測沒做出來，因此顯示的是舟骨區域的指標)

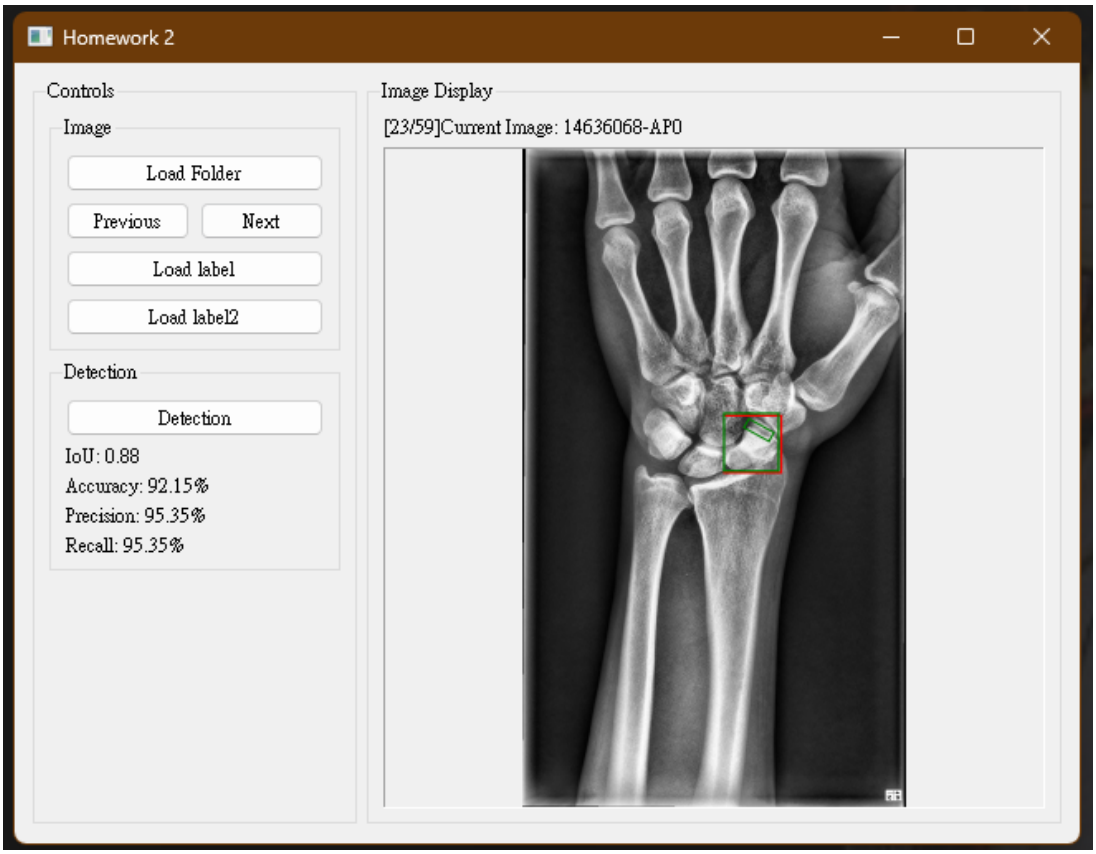

Figure 1: 圖形介面展示

train.py

- 訓練偵測舟骨的模型

tools 存放模型的定義、以及一些會用到的函式

- `basic_load.py`：提供 ui 進行基本的讀取操作
- `dataloader.py`：提供 train 讀取資料集的服務，class 繼承自 torch.dataset
- `fasterRCNN.py`：實現 fasterRCNN，依賴另外定義的部件 ROI 及 RPN
- `tool.py`： 各種供給模型運算的工具函式，例如計算 iou、計算 loss、旋轉框轉換... 等等

2 環境說明

執行環境

- Python 3.11.11
- Pytorch 2.5.1

- numpy 2.2.1
- opencv-python 4.10.0.84
- pillow 11.1.0

訓練參數

- lr: 0.001
- `num_epochs`: 60
- 在第 36, 48 epochs 時會下降學習率

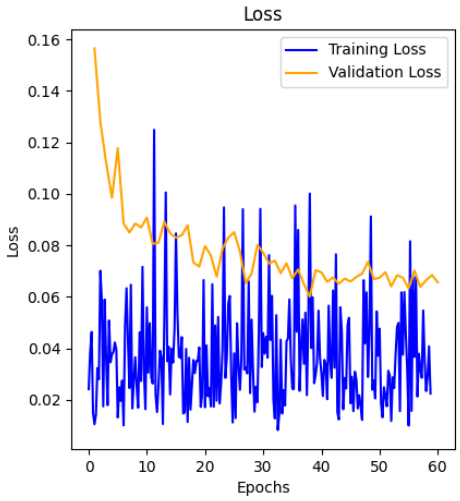

Figure 2: 訓練過程 loss 趨勢變化

3 模型架構

3.1 class FasterRCNN

FasterRCNN 網路

架構

- backbone : vgg16
  - 載入預訓練模型
  - 去除最後的分類層
  - 前 10 層為基本特徵，不進行梯度計算
- RPN 層，參考 class RPN (3.2)

- ROIHead 層，參考 class ROIHead (3.3)
- num_classes=2，舟骨及背景
- img_mean = 0.5，灰階影像進行單通道正規化
- img_std = 0.3，同上
- min_size = 1200，對輸入影像大小限制
- max_size = 1400

forward

1. 增加靈活性，檢測輸入影像若不為 Tensor 先行轉換
2. 將影像大小縮放至符合標準。(訓練時) label bbox 一起轉換
3. 為了符合 vgg16 的要求，複製單通道成 3 通道
4. 影像經過 backbone 網路 (vgg16) ，產生的特徵圖
5. 特徵圖經過 RPN ，產生 proposals
6. 特徵圖與 proposals 經過 ROIHead，其包含了最後的分類及回歸，得到最終輸出的框及 scores
7. (非訓練時) 將提議框 resize ，以對應到未進行縮放的影像
8. return rpn_output, frcnn_output 以供訓練

其他工具

- norm_resize_image 輸入前會進行 resize，因此 frcnn 產出的框不符合原影像座標，此函數以線性插值將框轉換成原圖尺寸

3.2 class RPN

包含 RPN 網路

架構

- anchors size : 64, 128, 256 三種
- anchors aspect ratios : 0.5, 1.0, 1.5
- 每個點上共 9 個 anchors
- 輸入 vgg16 提取的特徵圖，dim=512
- rpn_conv一層 3×3 conv，dim = 512
- 輸出結果分別輸入分類層及回歸層，dim 分別為 9、9×4

forward

1. 輸入 vgg16 產生的特徵圖
2. 經過 `rpn_conv` 並 ReLU 處理
3. 產生的結果分別以 `cls_layer`, `bbox_reg_layer` 產生提議框
4. 產生的資料轉成提議框:
   (a) 根據影像大小以及設定的 anchor size 產出所有 anchors
   (b) 展開輸出，得到一個提議框列表
   (c) 將回歸層的輸出套用在原始 anchor，得到真正提議
5. 提議框初步篩選
   - 分數太低的先濾掉
   - nms 過濾掉重疊框
6. (訓練時) 處理 target
   (a) 根據與 label 的 IOU 判斷提議框為 positive or negative
   (b) 計算 label 標出的框與提議框的轉換距離，中心點標記是線性距離，寬高差距取 log
   (c) 避免訓練資料中 positive negative 比例懸殊造成失衡，依固定比例進行採樣
7. (訓練時) 計算 loss
   (a) 回歸層使用 `smooth_l1_loss` , $beta = 1/9$ (只計算 positive)
   (b) 分類層使用 cross entropy
8. return proposals, scores, (訓練時) loss

其他工具

- `gen_anchors` : 在特徵圖的每個座標上產生 9 個 anchor，對應到原始 image 的位置上
- `assign_targets_to_anchors` : 判定 anchors 是不是背景
- `filter_proposals` : 輸入 proposals 與 scores，篩選掉不良的提議 (依大小、分數、重疊)

3.3 class ROIHead

包含 ROIpooling 以及之後的分類回歸層

架構

- `num_classes=2`, 分類背景或是舟骨
- `in_channels=512`，來自 vgg16 產生的特徵圖
- `fc_inner_dim=1024`，兩層 fc
- 一個 `cls_layer`，一個 `bbox_reg_layer`

forward

1. 輸入特徵圖 (dim = 512)、proposals
2. (訓練時) 處理 target，控制 positive 提議框佔總 target 比例，以免資料失衡
3. `roi_pool`, 將 proposals 框出的區域剪切並 pool 成同樣大小
4. 平坦層 $\rightarrow fc \times 2$
5. 分別用分類層與回歸層取得最後輸出
6. (訓練時) 計算 loss
   - 分類: `cross_entropy`
   - 回歸 (限非背景框) : `smooth_l1_loss`
   - `return cls_loss, loc_loss`
7. (非訓練時) :
   (a) 將回歸結果套用在 proposals 上
   (b) 分類結果使用 softmax 歸一
   (c) 刪除分類為背景的框
   (d) 過濾過小、分數過低、NMS 篩選
   (e) 依據分數高低 (信心程度) 排序框，return boxes, scores, labels

4 骨折檢測紀錄 (我最後沒做出來，紀錄一下)

4.1 加入旋轉角

原本我打算修改 Frcnn 的架構，加入旋轉角作為提議框維度，也在 tool 寫出將座標轉換成五個維度 (中心 x,y、寬高、旋轉)。

但是修改到計算 roi 那關時遇到困難，無法有效計算兩個旋轉提議框的重疊面積。之後只好放棄

4.2 使用 mmrotate 檢測

之後嘗試使用開源的 mmrotate 模型建構檢測骨折的模型。但是搞了很久連環境都安裝不起來。因為截止時間快到我還沒製作 ui ，因此決定放棄後半部分的骨折檢測，至少將舟骨檢測的功能做出 ui 來 demo