311706030 陳婕儀

DLMIA HW5 X 光影像中常見胸部疾病之物件偵測

一、 背景前言

胸部 X 光攝影(Chest film)，又稱為 X 光攝影或胸部攝影，使用於胸部區域的放射造影技術，負責檢視胸腔與其周邊的器官狀況。胸部 X 光是目前在醫療上最常見的檢驗項目之一，他能判讀：

- 胸腔，例如：肺炎,肺癌及縱膈腔病變等。
- 心臟，例如：心臟腫大, 先天性心臟病, 後天心臟 疾病及心衰竭等疾病。

本次作業是對胸部 X 光影像進行以下七種疾病的預測：

1. 心臟肥大(cardiac hypertrophy)
2. 主動脈硬鈣化(aortic atherosclerosis calcification)
3. 主動脈彎曲 (aortic curvature)
4. 肺尖肋膜增厚 (intercostal pleural thickening)
5. 肺野浸潤增加 (lung field infiltration)
6. 胸椎退化性關節病變 (degenerative joint disease of the thoracic spine)
7. 脊椎側彎 (scoliosis)

另外還包含正常的胸部 X 光影像，總共八類的類別預測，並判斷其位在胸部的哪個位置上。

二、 資料介紹與處理

本次的作業資料訓練資料包含 train 與 test 兩個資料夾，train 又細分成上述八種分別有照片與 mask，至於 test 裡共有 113 照片進行最終的測試。

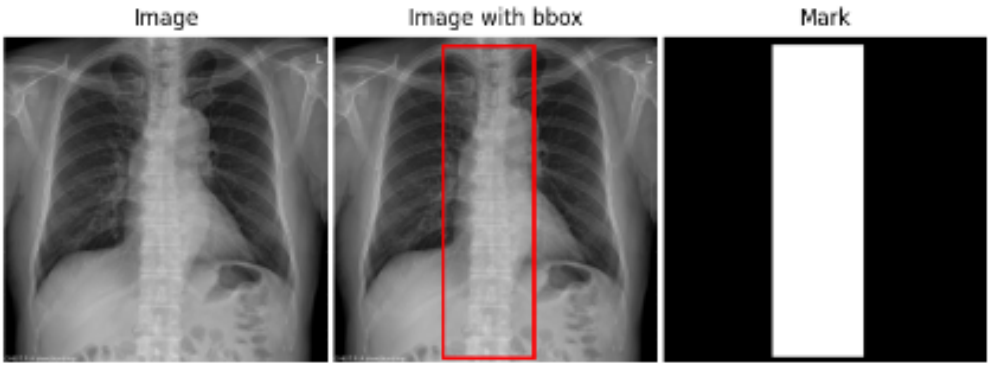
1. X-ray image normalization

首先，先對訓練資料進行轉換，將 csv 檔裡的類別中文轉成英文。接者對影像進行 X-ray image normalization，目的是為了要進行影像型態的轉換以及針對 L 與 R 的亮度進行調整，從指定的 DICOM 檔案中讀取影像，並取得原始的像素陣列，與獲取窗寬 (WW) 和窗位 (WC)，以及 BitsStored 的資訊。然後根據窗寬和窗位計算最大強度值 (iMax) 和最小強度值 (iMin)，對原始 BitsStored 進行強度的對數轉換 (log_img)，助於調整像素值的對比度，使用最簡單的色彩平衡算法，將強度正規化到指定的範圍 vmin 到 vmax。最後返回三個影像陣列：原始像素陣列 (origin)、經對數轉換處理後的像素陣列 (log_img)，以及正規化後的像素陣列 (normalize_img)。

2. 獲取所需的 bounding box

接下來將資料集中的 mask 轉換為丟入模型所需的 bounding box，並且寫

入 train data frame 裡，其將影像轉換為二值化遮罩，即將非零像素視為前景（物體），零像素視為背景。使用 scikit-image 的 label 函式，基於二值化遮罩創建連通區域標籤，形成連通區域標籤的陣列 (sk_mask)。使用 regionprops 函式從連通區域標籤中提取連通區域的屬性，包括邊界框（bbox），最後提取邊界框，並過濾掉面積小於 3000 的區域，最終返回第一個有效區域的邊界框座標（xmin、ymin、xmax、ymax）。



3. training set 跟 validation set

因應資料格式將上述八類轉為 0~7 的表示，分類 train 與 vaild 時，使用病人 ID 進行分類，因為一個病人在 data frame 可能有多筆的資料存在。

```
one_hot = binarizer.fit_transform(disease_id)
one_hot
```

Out[18]:

```
array([[1, 0, 0, ..., 0, 0, 0],
       [1, 0, 0, ..., 0, 0, 0],
       [1, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 1],
       [0, 0, 0, ..., 0, 0, 1],
       [0, 0, 0, ..., 0, 0, 1]])
```

4. COCO format 格式與 JPG 檔

最終將資料轉換為 COCO format 的形式來丟入物件偵測的模型裡，創建 images、annotations 和 categories 這三部分，然後將經過 normalization 影像存為 .jpg 檔，並將轉換為 COCO format 的資料存為 .json 檔，這樣完成我們資料目前的處理部分。

[23]:
```
train_coco = coco_format(training, categories)
print(train_coco)
```

```
{'images': [{'file_name': 'normal/220_94.jpg', 'height': 2560, 'width': 2472, 'id': 0}, {'file_n
ame': 'normal/220_93.jpg', 'height': 2496, 'width': 2312, 'id': 1}, {'file_name': 'normal/220_9
0.jpg', 'height': 2632, 'width': 2320, 'id': 2}, {'file_name': 'normal/220_88.jpg', 'height': 26
24, 'width': 2560, 'id': 3}, {'file_name': 'normal/220_86.jpg', 'height': 2632, 'width': 2544,
'id': 4}, {'file_name': 'normal/220_85.jpg', 'height': 2176, 'width': 2200, 'id': 5}, {'file_nam
e': 'normal/220_82.jpg', 'height': 2576, 'width': 2376, 'id': 6}, {'file_name': 'normal/220_81.j
pg', 'height': 2536, 'width': 2288, 'id': 7}, {'file_name': 'normal/220_79.jpg', 'height': 2504,
'width': 2408, 'id': 8}, {'file_name': 'normal/220_77.jpg', 'height': 2528, 'width': 2344, 'id':
9}, {'file_name': 'normal/220_74.jpg', 'height': 2464, 'width': 2416, 'id': 10}, {'file_name':
```

5. Dataset

回傳 image 與 target，target 含有 boxes：bounding box 的標註，labels：bounding box 對應的疾病類別，rea: bounding box 圍出的面積，Iscrowd，segmentation 時使用設為 0 ，imageID。

三、 模型介紹

本次訓練採用的模型為 Mask R-CNN。

Mask R-CNN 是一種先進的深度學習模型，屬於目標檢測領域的領先技術之一。相較於傳統的目標檢測方法，Mask R-CNN 不僅能夠準確地檢測圖像中的物體並定位其位置，還能進行實例分割，即在像素級別上區分不同物體實例，並為每個實例生成一個精確的二進制遮罩。這使得 Mask R-CNN 在需要更精細和細緻物體分割的應用場景中表現卓越。以下進一步探討 Mask R-CNN 的

關鍵特點：

1. 物體檢測（Object Detection）： Mask R-CNN 利用 Region Proposal Network（RPN）生成候選區域，並在這些區域上進行物體檢測。這能夠有效地定位圖像中的物體，並提供每個物體的類別標籤。
2. 實例分割（Instance Segmentation）： 除了物體檢測，Mask R-CNN 還能生成物體的精確像素級別的分割遮罩。這意味著模型能夠不僅檢測物體的位置，還能準確地區分不同物體實例，這在許多應用中是十分重要的，例如醫學影像中的細胞分割。

針對輸出類別數修改以下程式碼：用 PyTorch 的 torchvision 模組，初始化一個 Mask R-CNN 模型，採用預先訓練的 ResNet-50 FPN 架構。這是一種具有多尺度特徵的深度神經網路，有助於提升模型對不同尺寸物體的檢測性能。程式碼中進行一系列的修改，符合訓練所需預測的八類需求。

1. FastRCNNPredictor 的替換： 這部分確保了模型的物體檢測部分能夠適應特定的目標類別數量。透過替換分類預測器，使模型能夠根據應用需求進行目標檢測的調整。
2. MaskRCNNPredictor 的替換： 這裡替換了遮罩預測器，以調整模型的實例分割部分。這包括指定遮罩預測器的輸入特徵數、隱藏層的維度和目標類別的數量，進一步保證了模型的適應性。MaskRCNN 部分並無實際的進行訓練與結果，因為此次作業並無這塊，因此在 mask

部分給予預設空遮罩。

```
def maskrcnn(num_classes):
    model_ft = models.detection.maskrcnn_resnet50_fpn(pretrained=True)
    in_features = model_ft.roi_heads.box_predictor.cls_score.in_features
    model_ft.roi_heads.box_predictor = FastRCNNPredictor(in_features, num_classes)
    in_features_mask = model_ft.roi_heads.mask_predictor.conv5_mask.in_channels
    hidden_layer = 256
    model_ft.roi_heads.mask_predictor = MaskRCNNPredictor(in_features_mask, hidden_layer, num_classes)
    return model_ft

model =maskrcnn(num_classes =config.num_classes)
print(model)
```

四、 訓練結果評估

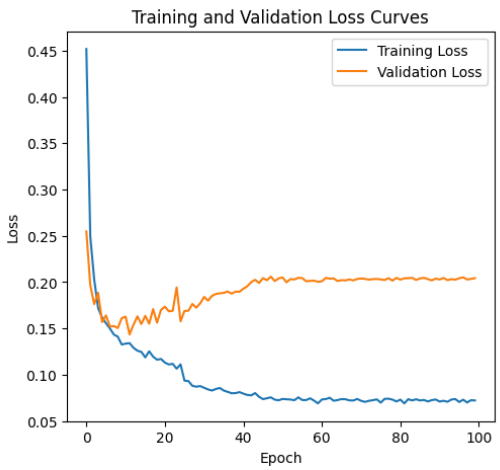
根據訓練其結果由以下說明，主要評估 IoU metric: bbox 在 IoU=0.50 的 Average Precision (AP) 的值，這裡也會針對 Loss, Classifier Loss, Box Reg Loss, RPN Box Reg Loss, Objectness Loss 進行簡單結論。

根據 validstion 的 Loss，會找到 Loss 最低值存下，因此用來預測模型的 best_ckpt 其相關訓練與驗證數值以及圖表如下所示：

```
*****Validation*****
Loss: 0.1434 | Classifier Loss: 0.0697 | Box Reg Loss: 0.0619 | RPN Box Reg Loss: 0.0047 | Objec
tness Loss: 0.0072
```

- Loss = 0.1434

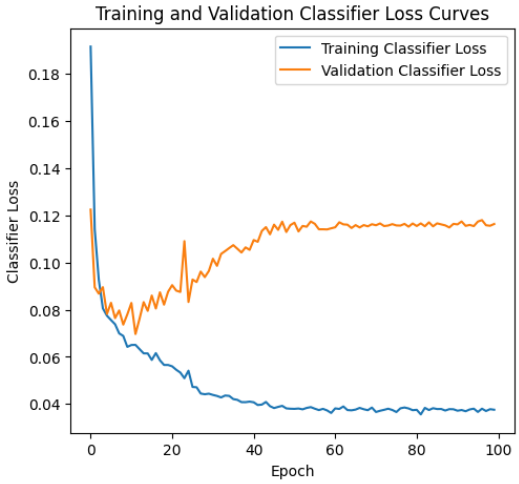
  我們在圖上可以看到 loss 在 epoch 12 時最低，在此之前是快速下降，在此之後往上攀升後維持在接近 0.2。



- Classifier loss =0.0697

一樣在圖上，我們可以看到 Classifier loss 在 epoch 12 時最低，

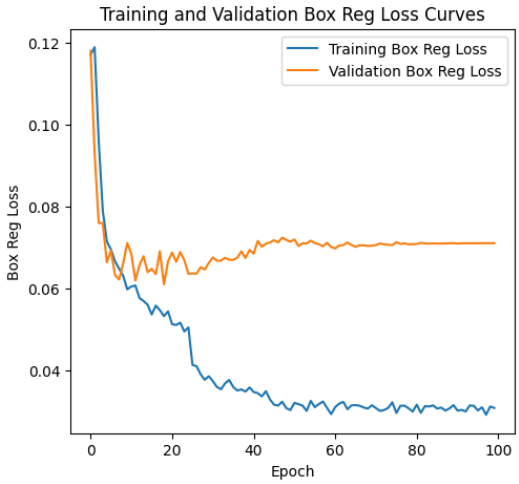
在此之前是快速下降，在此之後是往上攀升後維持靠近 0.12。



- Box Reg Loss =0.0619

在圖上，我們可以看到 Classifier loss 在 epoch 十幾時最低，在

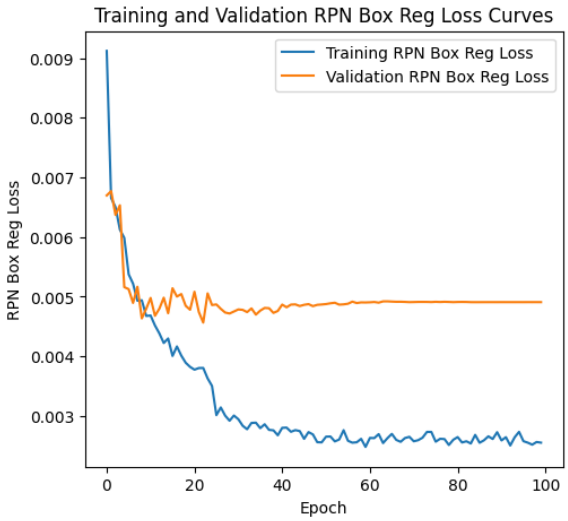
此之前是快速下降至此，在此之後是維持在 0.07 左右。



- PRN Box Reg Loss = 0.0047

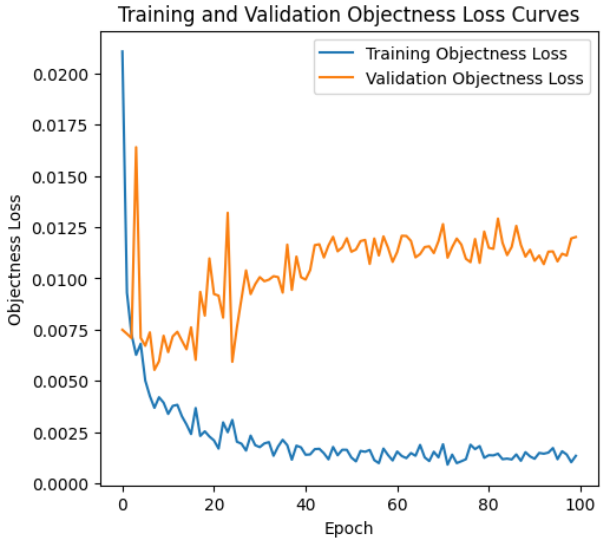
在圖上，我們可以看到 Classifier loss 下降至最底後，稍往上後至

0.005 並維持。



- Objectness Loss =0.0072

在圖上，我們可以看到 Classifier loss 在 epoch 12 時最低，但都沒有保持平穩狀態，但卻有往上走的趨勢。



根據 validation 最低的 loss 值，其訓練的 Average Precision (AP)值為 0.414，驗證的 Average Precision (AP)值為 0.403。

training

```
Epoch: 12/100 | LR: 0.005000
*****Training*****
Loss: 0.1340 | Classifier Loss: 0.0651 | Box Reg Loss: 0.0607 | RPN Box Reg Loss: 0.0045 | Objec
tness Loss: 0.0038
creating index...
index created!
Test:  [ 0/53]  eta: 0:00:38  model_time: 0.4102 (0.4102)  evaluator_time: 0.0724 (0.0724)  tim
e: 0.7350  data: 0.2453  max mem: 3228
Test:  [52/53]  eta: 0:00:00  model_time: 0.3733 (0.3739)  evaluator_time: 0.0697 (0.0768)  tim
e: 0.7128  data: 0.2530  max mem: 3228
Test: Total time: 0:00:37 (0.7106 s / it)
Averaged stats: model_time: 0.3733 (0.3739)  evaluator_time: 0.0697 (0.0768)
Accumulating evaluation results...
DONE (t=0.21s).
Accumulating evaluation results...
DONE (t=0.16s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.217
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.414
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.201
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.361
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.217
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.527
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.556
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.556
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.400
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.556
```

validation

```
*****Validation*****
Loss: 0.1434 | Classifier Loss: 0.0697 | Box Reg Loss: 0.0619 | RPN Box Reg Loss: 0.0047 | Objec
tness Loss: 0.0072
creating index...
index created!
Test:  [ 0/15]  eta: 0:00:11  model_time: 0.4478 (0.4478)  evaluator_time: 0.0783 (0.0783)  tim
e: 0.7850  data: 0.2519  max mem: 3228
Test:  [14/15]  eta: 0:00:00  model_time: 0.3736 (0.3592)  evaluator_time: 0.0716 (0.0686)  tim
e: 0.6687  data: 0.2340  max mem: 3228
Test: Total time: 0:00:10 (0.6688 s / it)
Averaged stats: model_time: 0.3736 (0.3592)  evaluator_time: 0.0716 (0.0686)
Accumulating evaluation results...
DONE (t=0.08s).
Accumulating evaluation results...
DONE (t=0.06s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.225
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.403
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.235
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.500
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.226
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.580
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.580
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.580
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.500
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.581
```

最後結果輸出只針對有疾病的七種類別寫入 csv 裡，一個病人可能包含多種疾病，因此一個病人可能含有多列的結果。

五、 結果討論

根據此次作業的模型，進行胸部 X 光影像的預測，在這裡我們可以看到訓練到後來，Average Precision (AP)值會大致固定在上下，沒有辦法再往上，或許增加訓練的資料量，可以幫助我們的模型訓練得更好，獲取更多特徵。但在真實的資料下，訓練模型本身就比我們所想的還要更難，除了增加資料量，我們可以再調整模型的結構或是一些參數來獲得更好的結果。