**Deep Learning 2024 Final exam\_B103012002\_林凡皓**

1. Implementing Backbone and Feature Pyramid Network

這次期末考是主要是實作FCOS(fully-convolutional one-stage object detection model)，此模型架構如下圖所示，

一張含有 文字, 圖表, 螢幕擷取畫面, 方案 的圖片

自動產生的描述

此架構主要可以分成三個部分，backbone、feature pyramid network (FPN)和head (prediction layers)。這邊主要先實作backbone和feature pyramid network的部分，這些實作內容在common.py檔案中。

* Backbone :

Backbone採用RegNetX-400MF，並載入pre-trained weight。這部分不需要自己實現。初始化backbone結果如下

一張含有 文字, 字型, 螢幕擷取畫面, 印刷術 的圖片

自動產生的描述

* FPN :

FPN為一種處理多尺度的網路架構，它的目的主要在增強對不同尺度的偵測能力，FPN會接收backbone的特徵c3、c4、c5並將它們轉換為新的特徵p3,、p4、p5。在架構途中會看到還有p6、p7的部分，這邊為了方便訓練與實作，將其拿掉。

1. 實現方法 :

要做到特徵的轉換可以透過convolutional layer來實現，這邊初始化三個convolutional layer分別用來處理p3、p4、p5。在forward path中，透過zip將c與p做配對，並利用for loop將每一組特徵都迭代過。每一次迭代過程中，先取得對應的convolutional layer，並將特徵c轉換為特徵p。對於p4、p5而言，我們可以將它們與上一級的特徵座內插，並將結果加到當前的特徵中。

1. 執行結果 :

一張含有 文字, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

1. Implementing FCOS prediction network (head)

這邊主要實現head prediction network的部分，用來在每個地方預測物件的類別、邊界。此網路定義在FCOSPredictionNetwork中。

1. 實現方式 :

因為要估測位置方框與類別，這邊會需要stem\_cls與stem\_box，每一個stem都包含一個convolutional layer與ReLU，而cls和box各自含有兩個stem。接著會需要初始化預測類別、邊界與中心的convolutional layer，這邊主要要注意output channel數量，預測類別的output channel為類別數量，預測邊界output channel為4(上下左右)，預測中心output channel為1。最後對每一層的convolutional layer作權重的初始化，權重的部分都初始化為

N ~ (0, 0.01)，bias都初始化為0。

1. 執行結果 :

一張含有 文字, 螢幕擷取畫面, 功能表, 字型 的圖片

自動產生的描述

1. Assigning a GT target to every model prediction

FCOS會在每個位置做物件類別、邊界與中心的預測，在訓練過程中我們需要為這三個預測設定一個ground truth。因為這三個預測都是在FPN (p3, p4, p5)上的一個位置，所以我們可以將這個問題看成是幫每個FPN特徵位置分配一個ground truth邊界與ground truth類別。

Ground truth邊界是從dataloader中取得的，為一個五維向量(x1, y1, x2, y2, C)，其中(x1, y1)為邊界的左上角，(x2, y2)為邊界的右下角，C是物件類別。我們會用(xc, yc)來表示FPN上的每一個位置，這些座標示圖片上的一個位置，也是特徵的receptive fields的中心。

這部分的實現在common.py中的get\_fpn\_locaton\_coords中。

1. 實現方式 :

先初始化一個dictionary，key為FPN層級的名稱，value都先初始化為None。接著透過for loop遍歷每一個FPN層級，並取得其shape和步伐。接著去計算縱向與橫向的位移矩陣，並可以利用torch.meshgrid的創建格子點。將縱向與橫向矩陣reshape成一維向量後堆疊起來，再加上level\_stride // 2來將座標調整到每個位置的中心。最後再將計算好的位置存入一開始初始化的dictionary中。

1. 執行結果 :

一張含有 文字, 螢幕擷取畫面, Rectangle, 正方形 的圖片

自動產生的描述

一張含有 螢幕擷取畫面, 室內, 相框, 牆 的圖片

自動產生的描述

一張含有 傢俱, 室內, 室內設計, 螢幕擷取畫面 的圖片

自動產生的描述

1. Matching feature map locations with GT boxes

接下來要將每個特徵位置與ground truth邊界匹配，在FCOS中有兩個規則來進行這種匹配 :

* 位置如果在一個box之中，則此位置與box匹配。如果該位置在兩個boxes之中，則該位置與較小的box匹配。如果位置不在任何box之中，該位置被歸類為背景。
* 對於特定的FPN層級，FCOS只考慮基於大小的boxes子集合，比較大的box被設定為p5，比較小的box被設定為p3。

透過這種方式匹配，每一個位置會收到一個ground truth的邊界與類別，即向量(x1, x2, y1, y2, C)，或是一個背景標籤(-1, -1, -1, -1, -1)，這部分不需要自己實現，執行結果如下

一張含有 車輛, 陸上交通工具, 輪, 跑車 的圖片

自動產生的描述

1. GT target for box regression

Box regression需要預測從特徵位置(上圖中黃色點)到邊界框邊緣(上圖中紅色框)的距離，這邊簡稱為LTRB。所有位置與ground truth邊界都是以絕對座標表示，範圍從(0, 224)，取決於圖像解析度。我們不能夠使用這種座標來訓練，因為這麼大的數值容易造成梯度爆炸，因此我們會將LTRB按照FPN層級的stride做normalization，normalize後的數值通常被稱為deltas。在進行inference時，由於模型會與測出normalize後的數值，因此我們需要對預測出來的數值進行逆轉換，將神經網路的輸出結果轉換為圖像中的預測框。這部分需要實現fcos\_get\_deltas\_from\_locations與fcos\_apply\_deltas\_to\_locations兩個函數。

* fcos\_get\_deltas\_from\_locations

1. 實現方式 :

先分別去計算特徵位置與ground truth box的距離，接著將計算出來的四個距離堆疊成一個矩陣deltas。最後處理背景的部分，將為背景的delta部分設定為(-1, -1, -1, -1)。

1. 執行結果 :



* fcos\_apply\_deltas\_to\_locations

1. 實現方式 :

此函數用來根據deltas的資訊來取得方框邊界。一開始要將normalize的deltas變回沒有normalize的樣子。由於模型預測出來的delta可能是負，但是這並不符合規範，因此將負的數值都clamp到0。最後再根據以下關係取得邊界座標

一張含有 文字, 筆跡, 行, 圖表 的圖片

自動產生的描述

1. 執行結果 :



1. GT targets for centerness regression

此數值在left = right且top = bottom時會最大，代表說特徵點位於ground truth框的中心。計算centerness的目的在於讓模型能夠專注在檢測方框中心的部分，進而減少在邊緣處的誤檢。

* 實現方法 :

透過deltas取得特徵位置到方框邊緣的距離，並根據centerness的公式計算出結果，最後再將背景的部分都設定為-1，讓模型忽略掉這些數值。

* 執行結果 :



1. Loss Functions

FCOS有三個預測輸出的layer，這三層分別用不同的loss function來優化。

* Object classification

對於FCOS來說，大多數位置都會被分配為背景，導致類別不平衡的問題，因此FCOS會使用Focal loss來處理這個問題。Focal loss為cross entropy的一種擴展，公式為

* Box regression

一般來說，FCOS會使用Generalized intersecton-over-union(GIoU)來做為優化box regression的loss function，但是由於要簡化問題，這邊直接使用L1 loss作為loss function，L1 loss公式為

* Centerness regression

centerness為[0, 1]之間的數值，因此採用binary cross entropy來做為Loss function，binary cross entropy公式為

* Total loss

每一個位置的loss會包含object classification loss、box regression loss和centerness regression loss，該位置的total loss為這三個部分相加後，以foreground locations的數量做平均。每個圖像中Foreground locations的數量高度可變，因此為了穩定訓練，total loss通常會以exponential moving average of foreground locations來做平均。

這部分不需要自己實現，執行結果如下

一張含有 文字, 字型, 螢幕擷取畫面 的圖片

自動產生的描述



1. Object detection model

這邊主要將剛才實現的所有東西應用到建立FCOS類別。

* \_\_init\_\_

初始化部分主要初始化backbone network與prediction network，這兩個網路都已經在前面實現好了，因此只需要呼叫它們並將channel size設定好即可。

* forward

forward path一開始將圖片送入backbone network中取得FPN feature，接著在送入prediction network中取得預測的類別、方框與中心。接著要取得每一個FPN層級的絕對座標，這部分可以透過common.py中實現的get\_fpn\_location\_coords來取得。透過先前定義的fcos\_match-location\_to\_gt來將ground truth box指派給特徵位置，並利用fcos\_get\_deltas\_from-locations取得boxes的ground truth deltas。再來要計算loss，這部分可以仿照Loss functions部分作者幫我們實現好的code。最後將計算好的classification loss、box regression loss與centerness loss回傳。

1. Overfit small data

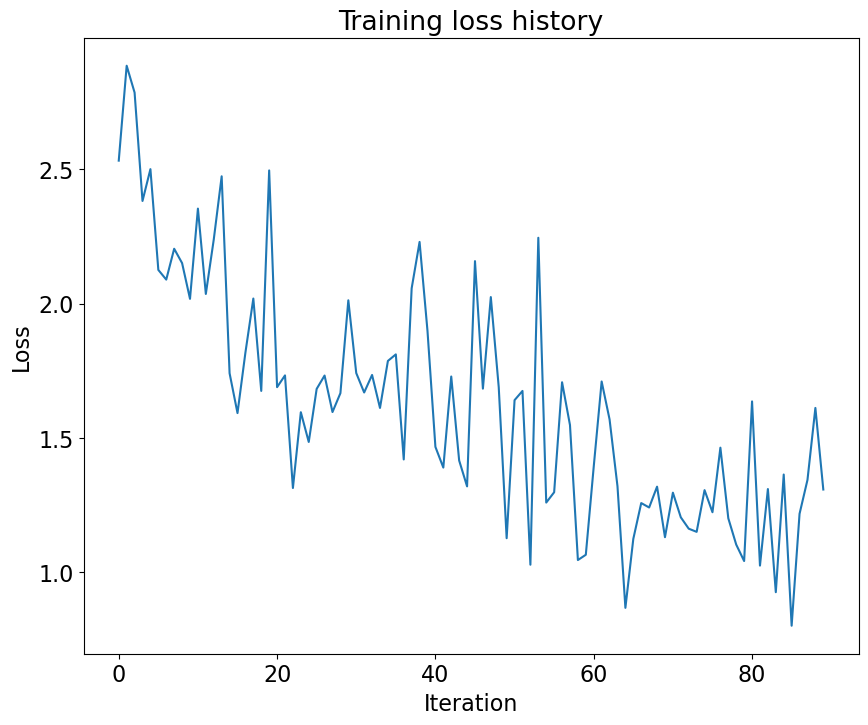
利用小資料來確認模型的實現是否正確，training loss curve如下

1. Train a net

確認好整個模型可以正常訓練後，用整個資料集做訓練，訓練參數配置都是期末考預設的參數，訓練9000 epochs後，loss與training loss curve如下

一張含有 文字, 字型, 螢幕擷取畫面, 黑色 的圖片

自動產生的描述



1. Non-Maximum Suppression (NMS)

為了等一下推論，這邊需要先實現nms。

* 實現方法 :

透過計算分類概率和中心度的幾何平均取的level\_pred\_scores，並取得之中最大值做為最有信心的類別與分數。接著去檢查信心程度是否高於threshold，保留那些信心程度高於threshold的類別與分數。透過fcos\_apply\_deltas\_to\_locations計算預測方框，並利用原始圖片修剪超出圖片的方框。最後匯集所有層級的預測結果，並執行NMS來過濾重疊的預測。

* 驗證結果 :

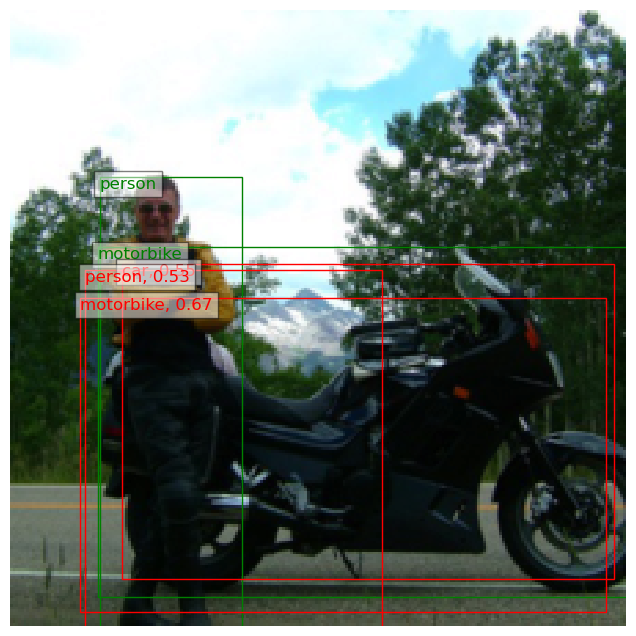
與torchvision的NMS比較結果如下

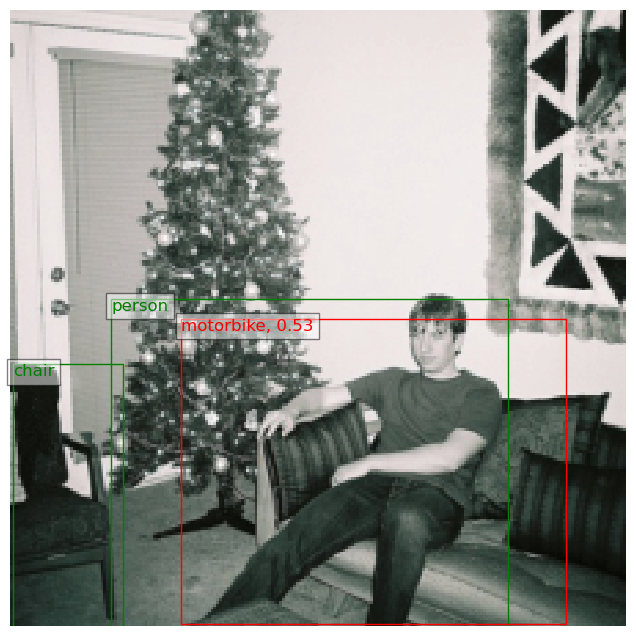
一張含有 文字, 字型, 螢幕擷取畫面 的圖片

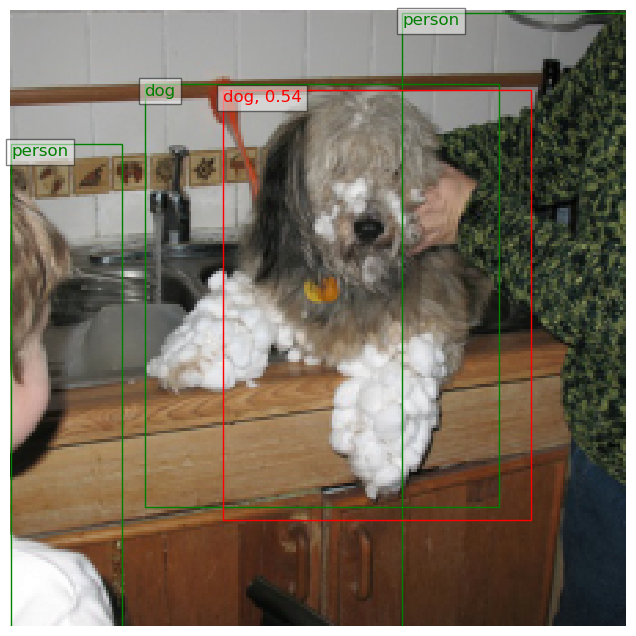
自動產生的描述

1. Inference

接著將剛才訓練好的模型用來做物件偵測，部分偵測結果如下









1. Evaluation

利用PASCAL VOC validation set計算mAP，計算結果如下

一張含有 文字, 螢幕擷取畫面, 行, 繪圖 的圖片

自動產生的描述

目前模型mAP為23.53 %，高於這次期末考標準的22 %。雖然說有達到期末考的標準，但是跟state of art的模型比較起來(mAP > 80%)相差很多，因此我嘗試一些方式試圖提升mAP。

1. Try to Improve mAP

* Change backbone to ResNet50

期末考預設是使用RegNetX-400MF作為backbone network，我嘗試將他修改為ResNet50。這部分主要使修改common.py的實現，我有將code貼到B103012002\_one\_stage\_detector.py最後，修改部分主要是在一開始建立\_cnn的地方，改為呼叫torchvision.model.resnet50，一樣去載入pre-train weight。由於自己電腦顯卡記憶體不足，因此這部分是在colab做訓練。訓練9000 epochs結果如下

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 字型, 繪圖 的圖片

自動產生的描述

這邊我有修改訓練所使用的函數，設定讓程式自動儲存loss最低的模型。將訓練好的模型計算mAP結果如下

一張含有 文字, 螢幕擷取畫面, 行, 圖表 的圖片

自動產生的描述

mAP提升到28.97 %，雖然有提升但是提升不多。從這樣的結果我猜測可能是訓練資料不足導致模型訓練效果不好，因此我嘗試做資料增生，這邊做了color jitter、Gaussian blur和gray scale，最後的訓練資料量為原先的3倍，這部分的code我有貼到B103012002\_one\_stage\_detector.py的最上面。增生後訓練結果如下

一張含有 文字, 螢幕擷取畫面, 行, 圖表 的圖片

自動產生的描述

可以看到mAP提升的很有限，這邊我猜測可能我做的資料增生方式會導致原先圖片的特徵遭到破壞，導致模型學習效果不好，因此我覺得做資料增生在這次考試的效果會很有限，因為能使用的增生方式不多(要能夠保持圖片特徵不被破壞且預測方框位置不可以更動到)，導致說訓練資料量會很有限，再加上one stage detector在準確度表現上本來就很有限，因此我將架構換成faster rcnn，並利用transfer learning 來解決資料量不足的問題。

* Faster RCNN transfer learning

在torchvision中有pre-train好的faster RCNN可以直接使用，因此transfer learning挑戰的地方並不是在如何建立架構以及如何訓練，而是在資料處理部分。這邊會針對模型、資料處理、訓練、推論與模型評估做詳細解釋。

1. 模型架構 :

模型架構部分我採用torchvision中的fasterrcnn\_resnet50\_fpn架構進行訓練，transfer learning pre-trained weight我採用在COCOv1資料集上預訓練好的權重。Fasterrcnn\_resnet50\_fpn的架構為參考論文”Faster R-CNN: Towards Real-Time Objection Detection with Region Proposal Network”所提出的架構，架構如下

一張含有 文字, 螢幕擷取畫面, 設計 的圖片

自動產生的描述

Backbone network採用ResNet50，且與期末考一樣採用FPN架構。由於我們採用的資料集與COCOv1所定義的類別數目不同，因此head (prediction network)的部分會需要做修改。我們採用的資料集一共有21個類別，所以需要將原先針對COCOv1的head換成是分21個類別的head，這部分可以用torchvision中的FastRCNNPredictor的做更換。

1. 資料處理 :

從torchvision的document中可以看到fasterrcnn\_resnet50\_fpn有預期要接收的輸入資料格式與輸出格式，規範如下

一張含有 文字, 螢幕擷取畫面, 字型, 文件 的圖片

自動產生的描述

期末考所提供的dataloader並不符合這樣的格式，因此所有的code都需要更改，這部分的code都在B103012002\_one\_stage\_detector.ipynb最後Faster RCNN Transfer Learning中，這邊會依序解釋所有自己寫的函數。

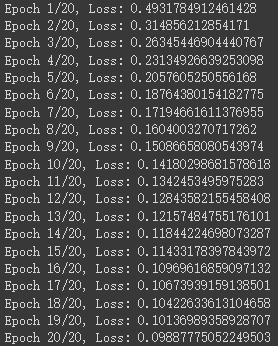
1. VOC\_CLASSES : 為一個encoder，將PASCAL VOC資料集編碼成ID。
2. get\_transform : 將圖片轉換成pytorch tensor，並做normalizaton。
3. collate\_fn : 使用pytorch的DataLoader來載入資料的時候，DataLoader會將資料打包成batch，一般情況下DataLoader會將每個batch中所以樣本組合成一個矩陣。如果資料的樣本形狀都相同的話，這樣的方法使可行的，但是對於物件偵測來說，每個圖片中的物體數量與尺寸都可能不同，因此需要自行定義一個函數來解決這個問題。此函數會接收一個含有tuple的list，並回傳一個包含兩個tuple，images和targets的tuple。
4. prepare\_targets : 此函數主要是將我們的資料變成符合fasterrcnn\_resnet50\_fpn的輸入格式。這部分會需要去解析一下PASCAL VOC資料集的資料結構，其資料結構如下一張含有 文字, 螢幕擷取畫面, 設計 的圖片

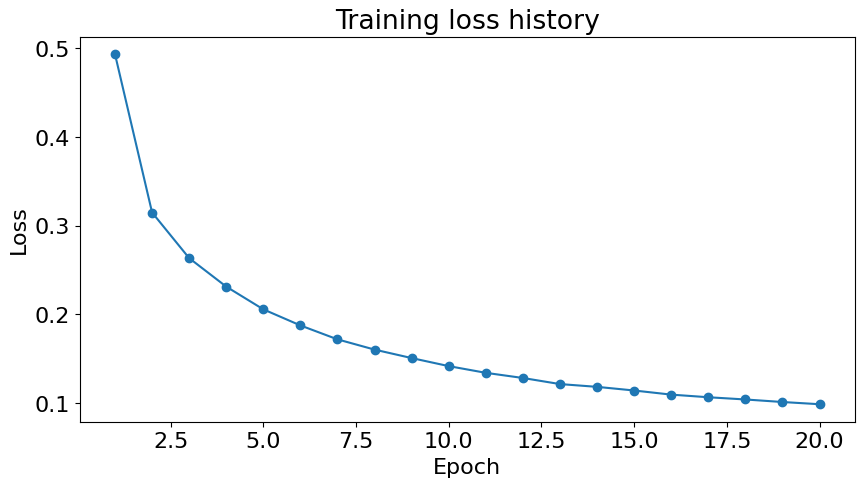
   自動產生的描述

target是圖像的標記數據，根據fasterrcnn\_resnet50\_fpn輸入資料規範，我們需要整理出boxes和label。Boxes所需要的資料都在’bndbox’中，因此可以透過dictionary key來取得資訊，並整理成0 <= x1 <= x2、0<= y1<= y2的格式。Label的部分也是透過dictionary key來取得，再透過一開始的encoder將類別轉換為ID。。

1. VOCDataset : 定義一個繼承torchvision.datasets.VOCDetection的類別，用來載入訓練資料。此類別初始化繼承torchvision.datasets.VOCDetection的方法，需要傳入資料路徑、資料年份(2007、2012)和資料類型(train、validation、test)。此外也需要初始化資料預處理的方法(transform)。Getter的部分先透過torchvision.datasets.VOCDetection的getter取得圖片和標記數據，並透過prepare\_targets將標記數據轉換為輸入格式，同時透過transform對圖片做預處理。最後將處理好的圖片與標記數據回傳。
2. train\_model : train\_model需要傳入待訓練模型、data\_loader、optimizer、device和epochs數量。一開始需要將model轉換為訓練模式，也就是開啟batch normalization和dropout。接著利用for loop迭代每一次epochs。每一次epochs都先從data\_loader中取的圖片與標記數據，接著去計算loss並根據optimizer做優化。計算當前epochs的average loss然後將這些資訊print出來並儲存到loss history中。最後再根據loss history繪製training loss curve。
3. main : 主要運行的程式碼。一開始要先載入資料，與期末考一樣採用PASCAL VOC2007 training set做訓練，optimizer採用SGD (lr = 0.001, momentum = 0.9, weight\_decay = 0.0005)，訓練20個epochs。
4. 訓練 :

訓練結果如下



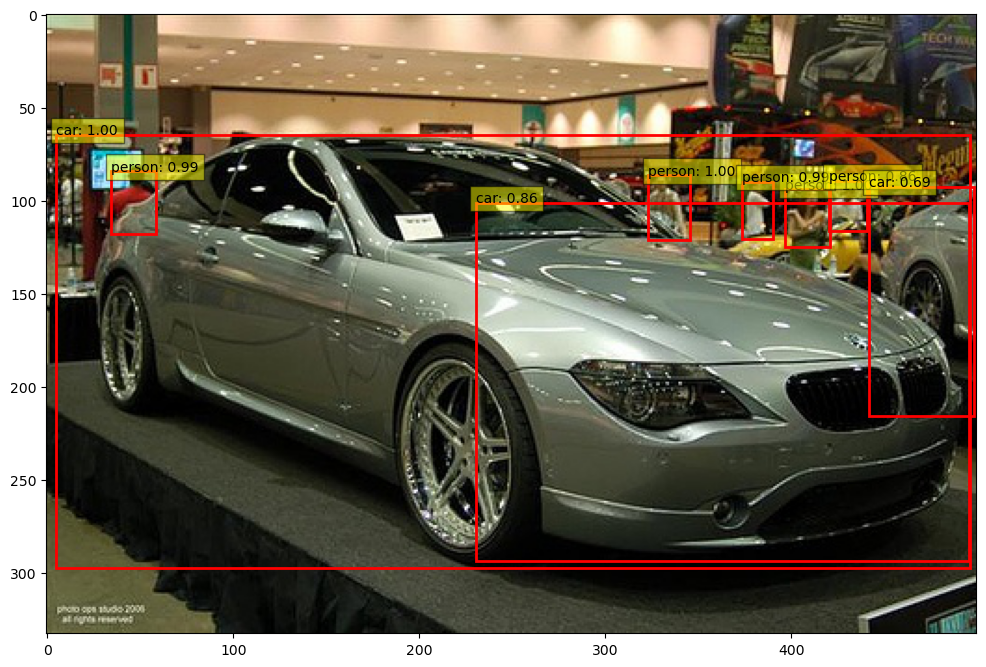


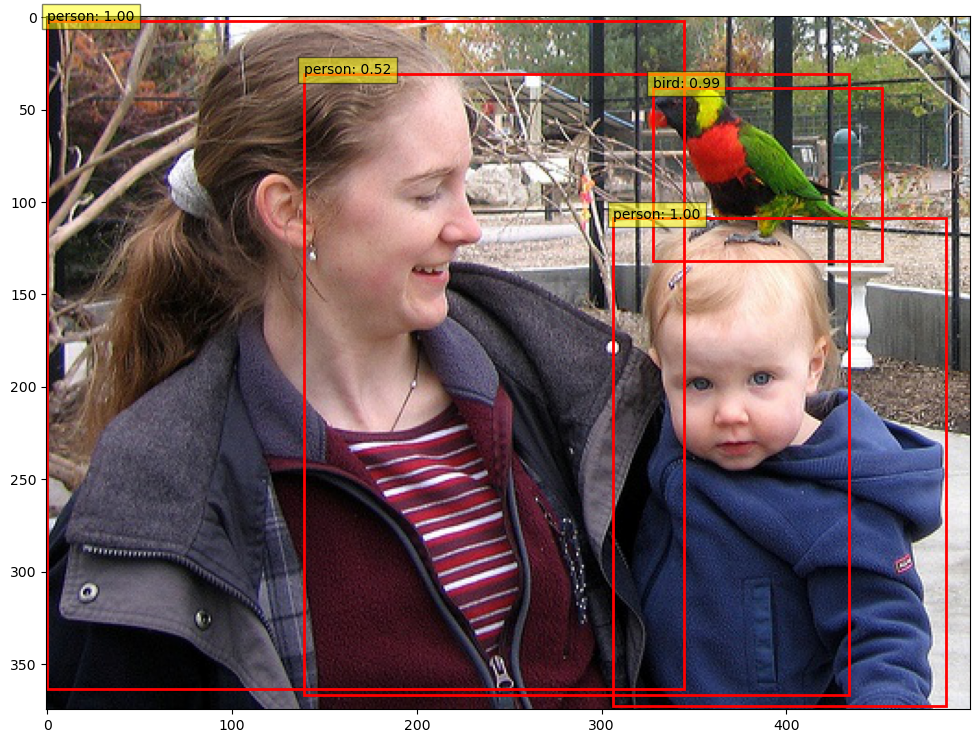
1. 推論與模型評估 :

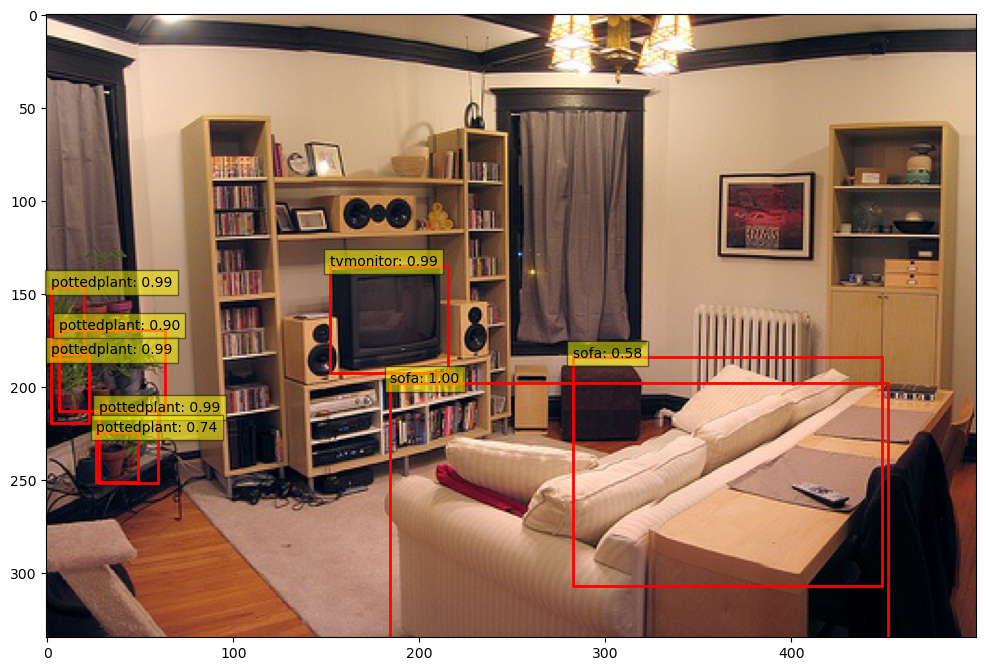
這部分也須根據fasterrcnn\_resenet50\_fpn的輸出格式定義inference with detector。

Inference with detector主要是用來產生計算mAP所需要的資料。首先須將預處理的圖片轉換回來，並設置好儲存ground truth與detection result的資料夾路徑。接著利用for loop迭代測試資料中每一張圖片，利用訓練好的模型做偵測。這邊使用的資料與期末考使用的資料相同，都是PASCAL VOC2007 validation set。接著透過fasterrcnn\_resnet50\_fpn的輸出資料結構取得boxes、scores和label，並過濾掉信心度過低的方框(threshold與期末考的數值同為0.4)。關於模型處理重複方框的方法，torchvision fasterrcnn\_resnet50\_fpn也是採用NMS，預設threshold為0.5，與期末考的設置都相同。

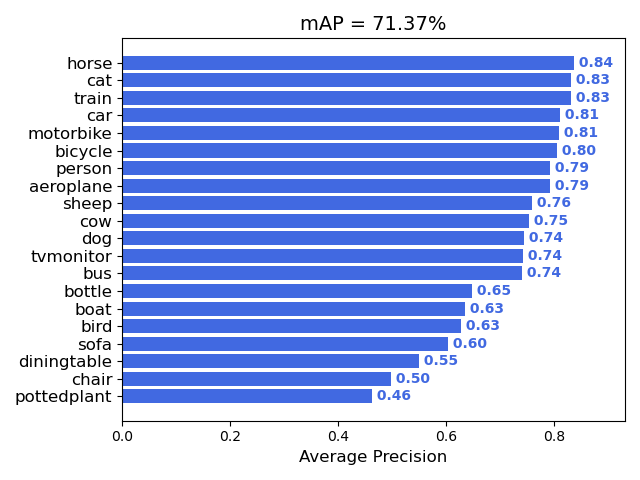
部分預測圖片結果如下







Faster RCNN transfer learning的mAP表現如下



只需要訓練20 epochs mAP就可以提升到71.37 %，與期末考實現的one stage detector的23.53 %相比提升了47.84 %。

1. Reference

[1] OpenAI. (2023). ChatGPT (Mar 14 version) [Large language model]. <https://chat.openai.com/>

[2] Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks” <https://arxiv.org/abs/1506.01497>

[3] PyTorch “fasterrcnn\_resnet50\_fpn” <https://pytorch.org/vision/main/models/generated/torchvision.models.detection.fasterrcnn_resnet50_fpn.html#torchvision.models.detection.fasterrcnn_resnet50_fpn>

[4] Lin et al, “Feature Pyramid Networks for Object Detection” <https://arxiv.org/abs/1612.03144>