Deep Learning Assignment 5

Person	品叡 池
Tags	DeepLearning

YOLO — You Only Look Once

Todo:

在 pre-trained model (ResNet50) 實現 Yolo v2 Loss function 、進行fine-tune，實作 object detection model。

Datasets

Pascal VOC 2007

Training Process

想找到最好的訓練結果(Low loss, High accuracy)，需要不停的調整訓練參數以達到最佳化。

Problem Shooting

這次作業的目標在於實作 YOLO v2 Loss function 前面花比較多時間閱讀論文原著，並理解如何正確計算loss。

$$
\begin{aligned}
&\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}
$$

YOLO v2 loss 由三大部分組成：

💡 *yolo v2 loss 計算方式會以每個cell最適合做為代表的 Bounding Box 去計算。*

- Regression loss — 前面兩行，計算〔 含有物件的 Bounding Box〕 的中心偏移、計算 Bounding Box 的大小差異。
- Confidence loss — 中間兩行，計算 〔含有物件的 Bounding box〕以及〔不含有物件的 Bounding Box〕的 loss。
- Classification loss — 最下面一行，計算〔含有物件的 cell〕預測出的 class 結果與 ground truth 的 loss。

Function Implementation

☀️ *以 MSELoss 為計算方式。*

📦Regression Loss

其中分為兩部分，第一要計算 Bounding Box 的中心位置與 ground truth 的偏移量；第二要計算 Bounding Box 的長寬與 ground truth 的差值。

```
def get_regression_loss(self, box_pred_response, box_target_response):
        """
        Parameters:
        box_pred_response : (tensor) size (-1, 4)
        box_target_response : (tensor) size (-1, 4)
        Note : -1 corresponds to ravels the tensor into the dimension specified
        See : https://pytorch.org/docs/stable/tensors.html#torch.Tensor.view_as

        Returns:
        reg_loss : scalar

        """
        # CODE
        # your code here
        # 計算 〔obj_boxes〕 與 〔ground truth〕 的 error, 利用 xywh 的差異來計算

        # x, y 的值都在 box_pred_resoonse, box_target_response 的前面兩個 column，直接以 indexing 的方式取出並計算MSE
        xy_loss = F.mse_loss(
            box_pred_response[:, :2], box_target_response[:, :2], reduction="sum")
        wh_loss = F.mse_loss(
            box_pred_response[:, 2:].sqrt(
            ), box_target_response[:, 2:].sqrt(), reduction="sum")
        reg_loss = xy_loss + wh_loss

        return reg_loss
```

→ 這個 loss 的計算就需要透過計算 predict box 與 ground truth box 之間的 iou 以判斷該 box 是否最適合代表該 cell 的 bounding box。

🎯Confidence Loss

分為兩個部分，計算含有物件的 cell 的 bboxes 與 ground truth 的差值；計算不含物件的 cell 的 bboxes 與 ground truth 的差值。

```
def get_contain_conf_loss(self, box_pred_conf, box_target_conf):
        """
        Parameters:
        box_pred_conf : (tensor) size (-1,1)
        box_target_conf: (tensor) size (-1,1)

        Returns:
        contain_loss : scalar

        Hints:
        The box_target_conf should be treated as ground truth, i.e., no gradient
        """
        # CODE
        # your code here

        # 〔box_pred_conf〕 參數為一個 [size(-1, 10)] 的 Tensor
        # 〔box_target_conf〕參數為一個 [size(-1, 5)] 的 Tensor
        pred_boxes = box_pred_conf.view(-1, 2, 5) # 摺疊 tensor 的維度以符合需求(B 個 Boxes)
        confidences = pred_boxes[:, :, -1].view(-1) # 特定出 confidence 的 column
        ground_truth = torch.ones_like(confidences)
        loss = F.mse_loss(confidences,
                          ground_truth, reduction='sum')
        # print("contain con loss: ", loss)
        return loss
```

```
def get_no_object_loss(self, pred_boxes_list, has_object_map):
        """
        Parameters:
        pred_boxes_list: (list) [(tensor) size (N, S, S, 5)  for B pred_boxes]
        has_object_map: (tensor) size (N, S, S)

        Returns:
```

```
        loss : scalar

        Hints:
        1) Only compute loss for cell which doesn't contain object
        2) compute loss for all predictions in the pred_boxes_list list
        3) You can assume the ground truth confidence of non-object cells is 0
        """
        ### CODE ###
        # Your code here

        no_obj_map = ~ has_object_map
        # 把每個 cell 的 bboxes 分割，以便後續的計算
        no_obj_boxes = pred_boxes_list[no_obj_map].view(-1, 2, 5)

        # 取出所有的 confidence 並 vectorize
        confidences = no_obj_boxes[:, :, -1].view(-1)

        # 有 obj 的 cell confidence 為 1
        ground_truth = torch.zeros_like(confidences)

        loss = F.mse_loss(confidences, ground_truth, reduction="sum")
        return loss
```

→ 這個部分其實是比較令人疑惑的，最初以為no object loss 的計算方式需要計算**所有no obj bboxes**的數值但是計算的結果並不如預期能夠讓模型提高準確率。重新檢視論文內容，似乎使要以 confidence 最大的一個 box 來進行計算才是正確的做法(has object loss 也是相同方法)。

🐕 Classification Loss

計算每個cell所含物件的機率值與ground_truth的loss。

```
def get_class_prediction_loss(self, classes_pred, classes_target, has_object_map):
        """
        Parameters:
        classes_pred : (tensor) size (batch_size, S, S, 20)
        classes_target : (tensor) size (batch_size, S, S, 20)
        has_object_map: (tensor) size (batch_size, S, S)

        Returns:
        class_loss : scalar

        https://pytorch.org/docs/stable/generated/torch.nn.MSELoss.html
        """
        ### CODE ###
        # Your code here
        # 只計算包含 obj 的 cell
        pred_probs = classes_pred[has_object_map]
        target_probs = classes_target[has_object_map]

        loss = F.mse_loss(
            pred_probs, target_probs, reduction='sum')

        return loss
```

Discussion

Initial Configs

```
learning_rate = 0.001 # lr 後續可以調整
num_epochs = 50
batch_size = 16 # 如果調到 32 GPU 資源不夠

# Yolo loss component coefficients (as given in Yolo v1 paper)
lambda_coord = 5
lambda_noobj = 0.5
```

```
criterion = YoloLoss(S, B, lambda_coord, lambda_noobj)
optimizer = torch.optim.SGD(net.parameters(), lr=learning_rate, momentum=0.9, weight_decay=5e-4)
```

After 5 epoch:

```
Epoch [5/5], Iter [100/235], Loss: total=6.571, reg=1.840, containing_obj=2.721, no_obj=0.680, cls=1.330
Epoch [5/5], Iter [150/235], Loss: total=6.661, reg=1.854, containing_obj=2.796, no_obj=0.680, cls=1.331
Epoch [5/5], Iter [200/235], Loss: total=6.621, reg=1.840, containing_obj=2.778, no_obj=0.681, cls=1.323
---Evaluate model on test samples---
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
100%|██████████| 1255/1255 [00:27<00:00, 45.24it/s]
---class aeroplane ap 0.2591731726812007---
---class bicycle ap 0.24969250950447186---
---class bird ap 0.1415560726407039---
---class boat ap 0.051440329218107---
---class bottle ap 0.0--- (no predictions for this class)
---class bus ap 0.0--- (no predictions for this class)
---class car ap 0.28141680607583186---
---class cat ap 0.13408602150537635---
---class chair ap 0.1402756671232296---
---class cow ap 0.0--- (no predictions for this class)
---class diningtable ap 0.0--- (no predictions for this class)
---class dog ap 0.17069986259165018---
---class horse ap 0.17565398201893534---
---class motorbike ap 0.021739130434782608---
---class person ap 0.2112025751021171---
---class pottedplant ap 0.044140300674645---
---class sheep ap 0.05203900513747712---
---class sofa ap 0.0--- (no predictions for this class)
---class train ap 0.3256431668357761---
---class tvmonitor ap 0.288423377490321---
---map 0.12735909895l7313---
4 [0.2591731726812007, 0.24969250950447186, 0.1415560726407039, 0.051440329218107, 0.0, 0.0, 0.28141680607583186, 0.13408602150537635,
Updating best val loss: 7.01748
```

在前五個 epoch 使用 SGD 更新權重就能夠讓模型有 map = 0.127… 的表現。

- 變更 optimizer: SGD → Adam:

After 5 epochs:

```
Starting epoch 5 / 5
Learning Rate for this epoch: 0.001
Epoch [5/5], Iter [50/235], Loss: total=24.508, reg=2.767, containing_obj=2.220, no_obj=16.709, cls=2.812
Epoch [5/5], Iter [100/235], Loss: total=24.012, reg=2.764, containing_obj=2.253, no_obj=16.283, cls=2.712
Epoch [5/5], Iter [150/235], Loss: total=23.820, reg=2.763, containing_obj=2.255, no_obj=16.072, cls=2.730
Epoch [5/5], Iter [200/235], Loss: total=23.695, reg=2.808, containing_obj=2.289, no_obj=15.777, cls=2.820
---Evaluate model on test samples---
100%|██████████| 1255/1255 [02:02<00:00, 10.23it/s]
---class aeroplane ap 0.0006049785759792214---
---class bicycle ap 0.0---
---class bird ap 0.0---
---class boat ap 2.4207213749697410e-05---
---class bottle ap 0.0---
---class bus ap 0.0---
---class car ap 1.577050757378626e-05---
---class cat ap 4.1436177926948024e-05---
---class chair ap 5.841994194739619e-06---
---class cow ap 0.0---
---class diningtable ap 0.0---
---class dog ap 0.0---
---class horse ap 0.0---
---class motorbike ap 0.0---
---class person ap 0.004105723315232621---
---class pottedplant ap 0.0---
---class sheep ap 0.0---
---class sofa ap 2.149982800137599e-05---
---class train ap 0.0---
---class tvmonitor ap 0.0---
---map 0.00024097288063291947---
4 [0.0006049785759792214, 0.0, 0.0, 2.420721374969741e-05, 0.0, 0.0, 1.577050757378626e-05, 4.1436177926948024e-05, 5.841994194739619e
```

Question: 為何 SGD 的表現明顯高過於 Adam？因為不是很清楚造成的原因所以後續仍使用SGD 作為 optimizer

經過 50 epochs (約 30 mins) 的訓練之後所得到結果 map = 0.55…， 希望能獲得更好的表現繼續讓模型訓練 100 epochs (約 1 hr)，map = 0.61…。

- 使用 learning rate scheduler

```
scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer, mode='min', factor=0.1, patience=5, verbose=True)
```

觀察 Loss 的變化動態調整學習率。

→ 表現並無太大的差異，使用助教提供固定學習率下降的方式就夠了。

Final Result on validation

```
---Evaluate model on test samples---
100%|██████████| 1255/1255 [00:24<00:00, 52.25it/s]
---class aeroplane ap 0.6495342035167719---
---class bicycle ap 0.5995116022900857---
---class bird ap 0.49538010766478285---
---class boat ap 0.43376342861206746---
---class bottle ap 0.23963014250389275---
---class bus ap 0.6864565598482422---
---class car ap 0.703436601353923---
---class cat ap 0.7709911751265506---
---class chair ap 0.33809238820662135---
---class cow ap 0.5387295349235165---
---class diningtable ap 0.3550931479004569---
---class dog ap 0.655615804496382---
---class horse ap 0.7098227984733275---
---class motorbike ap 0.5467164959273587---
---class person ap 0.5673300320180734---
---class pottedplant ap 0.29261978116521264---
---class sheep ap 0.43321949727408404---
---class sofa ap 0.4609778601352961---
---class train ap 0.8143345408584856---
---class tvmonitor ap 0.6189387201053136---
---map 0.5455097211200223---
```

Extra Credit

辨識影片中的物件

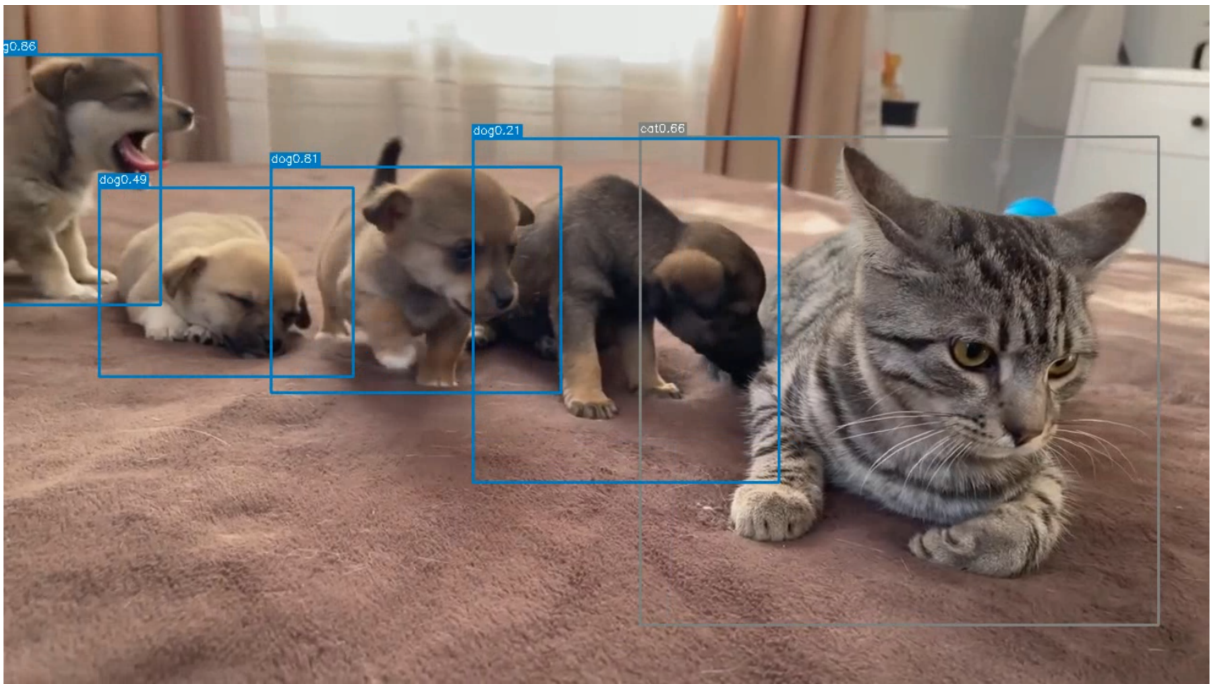
Original Video Link —

Funny Cat Reaction to Puppies [Kitty sees them for the First Time] - YouTube

Demo Video Link —

https://youtu.be/wh_yVOostq0

影片描述：包含兩種動物(cat, dog)如下圖所示：



Implementation

- 利用 cv2 套件將讀取影片並依照設定的禎數(fps)轉換為圖片，

```
from os import curdir
from torchvision import transforms
from torch.autograd import Variable


cap = cv2.VideoCapture('data/demo2.mp4')
colors = [tuple(255 * np.random.rand(3))for i in range(20)]

# 設置影片的寬度和高度
width = int(cap.get(3))
height = int(cap.get(4))

# 設置FPS（每秒幾幀）
fps = 2

# 設置影片的保存路徑
output_path = os.path.join(curdir, "data", "output_frames_2")

# 計數器
frame_count = 0

while(cap.isOpened()):
    ret, frame = cap.read()
    #print(len(frame))
    #for img in frame:
    #    print(img.shape)
    #assert 0
    #if  ret:
    #    #cv2.imshow('frame', frame)
    #    if cv2.waitKey(1) & 0xFF == ord('q'):
    #        break
    if not ret:
        break

    # 根據禎樹保存圖片
    if frame_count % int(fps) == 0:
        # 保存圖片
        img_name = "frame_" + str(frame_count) + ".jpg"
        filename = os.path.join(output_path, img_name)
        cv2.imwrite(filename, frame)
    frame_count += 1

cap.release()
cv2.destroyAllWindows()
```

- 所有圖片使用模型進行預測加上預測結果
- 最後將所有經過預測的圖片轉為影片檔

```
import os
import glob
net.eval()

# path
image_folder_path = os.path.join(os.curdir, "data", "output_frames_2")
output_video_path = os.path.join(os.curdir, "data", "video_puppy.avi")


# select files
image_files = glob.glob(os.path.join(image_folder_path, '*.jpg'))

# sorting files
sorted_image_files = sorted(image_files, key=lambda x: int(os.path.basename(x).split('_')[1].split('.')[0]))

# set configs
img = cv2.imread(sorted_image_files[0])
height, width, _ = img.shape
fourcc = cv2.VideoWriter_fourcc(*'XVID')
fps = 10
video = cv2.VideoWriter(output_video_path, fourcc, fps, (width, height))

# predict all images
for i, image_file in enumerate(sorted_image_files):
    image_name = (os.path.basename(image_file))
    image = cv2.imread(os.path.join(image_folder_path, image_name))
    print(image_name)
    result = predict_image(net, image_name, root_img_directory=image_folder_path)
    for left_up, right_bottom, class_name, _, prob in result:
        color = COLORS[VOC_CLASSES.index(class_name)]
        cv2.rectangle(image, left_up, right_bottom, color, 2)
        label = class_name + str(round(prob, 2))
        text_size, baseline = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.4, 1)
        p1 = (left_up[0], left_up[1] - text_size[1])
        cv2.rectangle(image, (p1[0] - 2 // 2, p1[1] - 2 - baseline), (p1[0] + text_size[0], p1[1] + text_size[1]),
                      color, -1)
        cv2.putText(image, label, (p1[0], p1[1] + baseline), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (255, 255, 255), 1, 8)
    # write images into video
    video.write(image)

# close writing mode
video.release()
```