XI’AN JIAOTONG-LIVERPOOL UNIVERSITY
西 交 利 物 浦 大 学

FINAL ASSIGNMENT
ANSWER SUBMISSION
COVER SHEET

Name	Mengfan (Surname)	Li (Given Name)
Student ID Number	2032048	
Programme	Msc MMT	
Module Title	Deep learning in computer vision	
Module Code	INT408	
Module Examiner	Jimin Xiao	

By uploading or submitting the answers of this FINAL ASSIGNMENT, I certify the following:

- I will act fairly to my classmates and teachers by completing all of my academic work with integrity. This means that I will respect the standards and instructions set by the Module Leader and the University, be responsible for the consequences of my choices, honestly represent my knowledge and abilities, and be a community member that others can trust to do the right thing even when no one is watching. I will always put learning before grades, and integrity before performance.
- I have read and understood the definitions of collusion, copying, plagiarism, and dishonest use of data as outlined in the Academic Integrity Policy, and cheating behaviors in the Regulations for the Conduct of Examinations of Xi’an Jiaotong-Liverpool University.
- This work is produced all on my own and can effectively represent my own knowledge and abilities.
- I understood the penalty rule for late or non-submission according to Xi’an Jiaotong-Liverpool University regulations. (Late Submission Policy: 5% of the total marks available for the assessment shall be deducted from the assessment mark for each working day after the submission date, up to a maximum of five working days.)

I understand collusion, plagiarism, dishonest use of data, submission of procured work, submission of work produced and/or contributed by others are serious academic misconducts. By uploading or submitting the answers with this statement, I acknowledge that I will be subject to disciplinary action if I am found to have committed such acts.

Signature ....李梦凡........ Date ....*20/12/2020*............................

Abstract

This assessment aims at evaluating students' ability to exploit the deep learning knowledge, which is accumulated during lectures, and after-class study, to analyze, design, implement, develop, test and document the pedestrian detection algorithm using Mask R-CNN framework. The assessment will be based on the Pytorch software.

Content

1 Introduction

This assessment aims at evaluating students' ability to exploit the deep learning knowledge, which is accumulated during lectures, and after-class study, to analyze, design, implement, develop, test and document the pedestrian detection algorithm using Mask R-CNN framework [1]. The assessment will be based on the Pytorch software.

2 Materials

PyCharm

Xshell

Pytorch

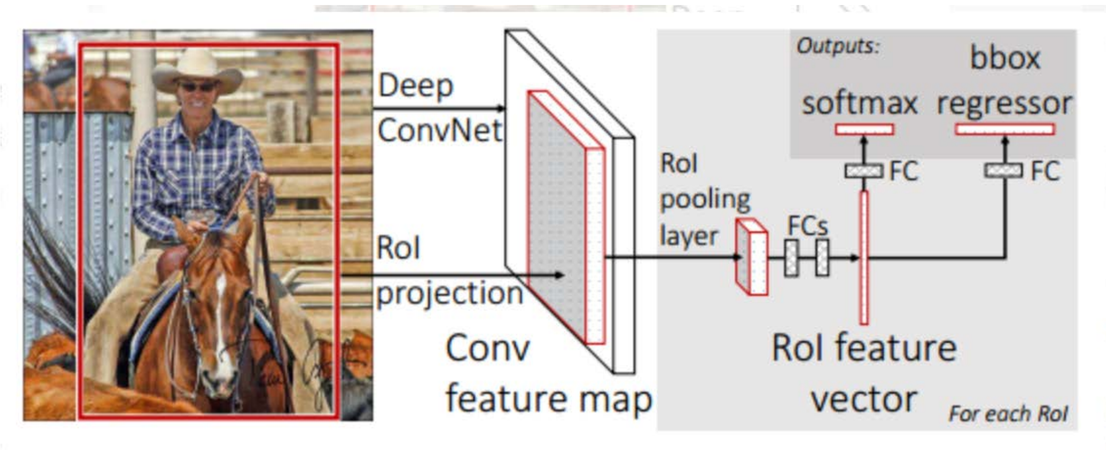
3 Methods, result and discussion

3.1

Please describe the 2 key components in the Mask R-CNN framework: the RoI Pooling layer and the loss functions in the framework.

RoI Pooling

Let's first look at the overall structure of fast R-CNN



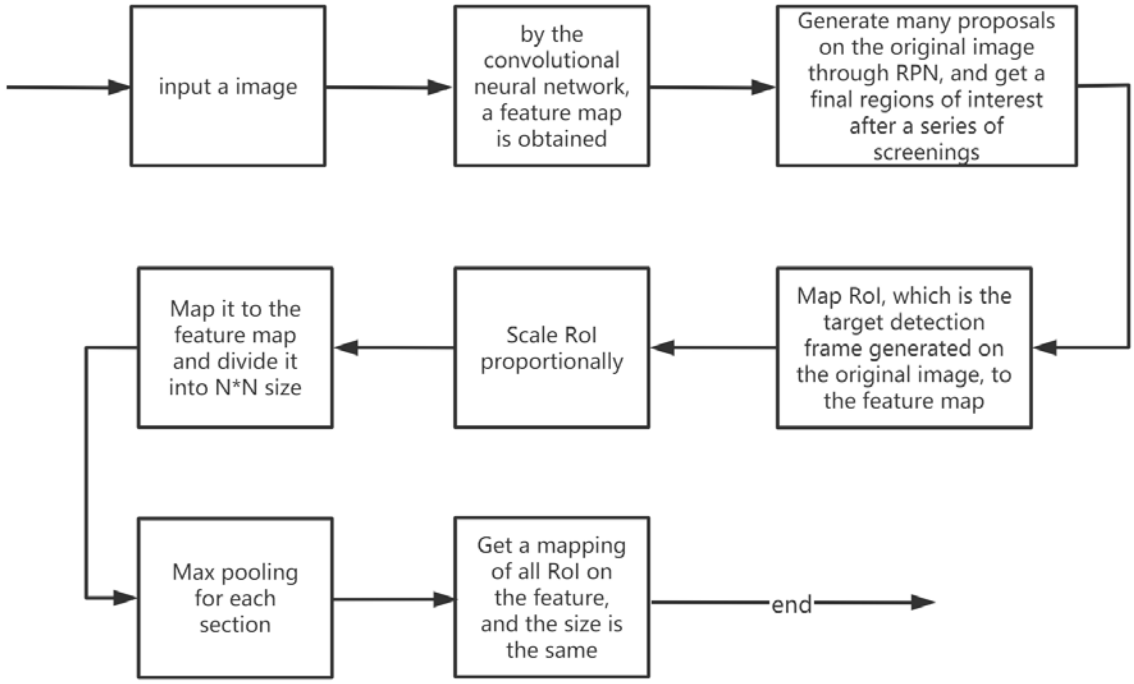
This layer draws on the SPP layer of SPP net, inputs feature map and object proposal, and then extracts the corresponding feature on the feature map for each object proposal, and makes the output have the same size.

The ROI pooling layer can achieve significant acceleration of training and testing, and improve detection accuracy.

This layer has two inputs:

1. Feature maps of fixed size obtained from deep networks with multiple convolution kernel pooling;
2. A matrix of N5 representing all ROIs, where N represents the number of ROIs. The first column represents the image index, and the remaining four columns represent the remaining upper left corner and lower right corner coordinates;

The workflow of RoI pooling is as follows



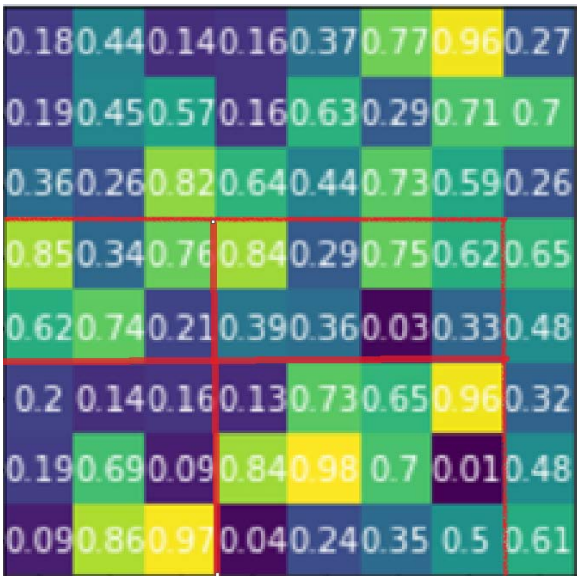
Step1: An 8*8 feature map is obtained through the convolutional neural network

Step2: Project the region of interest onto the feature map to get the coordinates (2,0) of the upper left corner and the coordinates (7,6) of the lower right corner

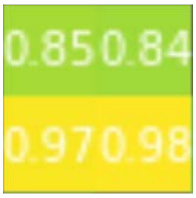
Step3: Divide the area into 2*2 size(because the output size is 2*2)

7/2=3.5, round up to 3;

5/2, round up to 2;



Step4: Finally, max pooling is performed on each section to get a 2*2 feature



Back-propagation through RoI pooling layers.

Back-propagation routes derivatives through the RoI pooling layer. For clarity, we assume only one image per mini-batch($N = 1$), though the extension to $N > 1$ is straightforward because the forward pass treats all images independently.

Let $x_i \in R$ be the i-th activation input into the RoI pooling layer and let y_{rj} be the layer's j-th output from the r-th RoI. The RoI pooling layer computes $y_{rj} = x_{i*(r,j)}$, in which $i*(r,j) = argmax_{i' \in R(r,j)} x_{i'}$. R(r,j) is the index set of inputs in the sub-window over which the output unit y_{rj} max pools. A single x_i may be assigned to several different outputs y_{rj}.

The RoI pooling layer's backwards function computes partial derivative of the loss function with respect to each input variable x_i by following the argmax switches:

$$\frac{\partial L}{\partial x_i} = \sum_r \sum_j [i = i^*(r,j)] \frac{\partial L}{\partial y_{rj}}$$

In words, for each mini-batch RoI r and for each pooling output unit y_{rj}, the partial derivative $\frac{\partial L}{\partial y_{rj}}$ is accumulated if i is the argmax selected for y_{rj} by max pooling. In back-propagation, the partial derivatives $\frac{\partial L}{\partial y_{rj}}$ are already computed by the backwards function of the layer on top of the RoI pooling layer.

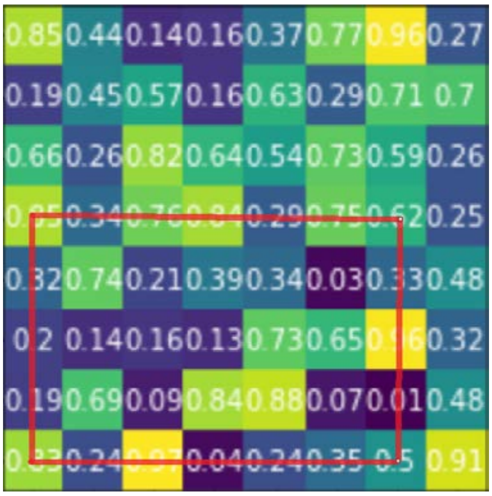
RoI Align

Since the error generated by RoI pooling after two quantizations has a very large impact on small targets, it is a point that needs to be paid attention to during instance segmentation. Therefore, the mask R-CNN proposed the RoI Align method, we can see From the comparison of the data in the paper, we found that the hint is still relatively large through observation, so the final result of the instance segmentation is obviously also very improved. The initial steps of RoI Align and RoI pooling are basically the same, but the mapping session is different.

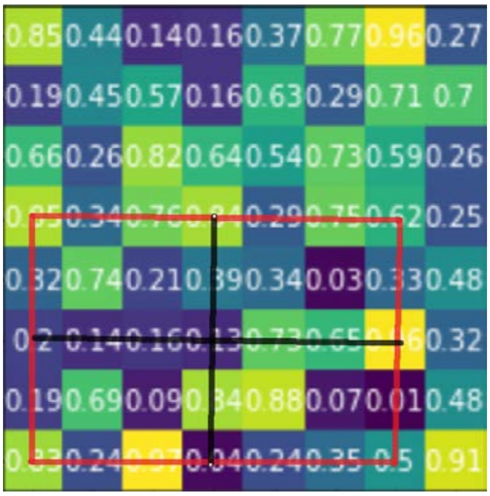
	AP	AP_{50}	AP_{75}	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}
RoIPool	23.6	46.5	21.6	28.2	52.7	26.9
RoIAlign	**30.9**	**51.8**	**32.1**	**34.0**	**55.3**	**36.4**
	+7.3	*+ 5.3*	*+10.5*	*+5.8*	*+2.6*	*+9.5*

Step1: Obtain 8*8 feature map through convolutional neural network

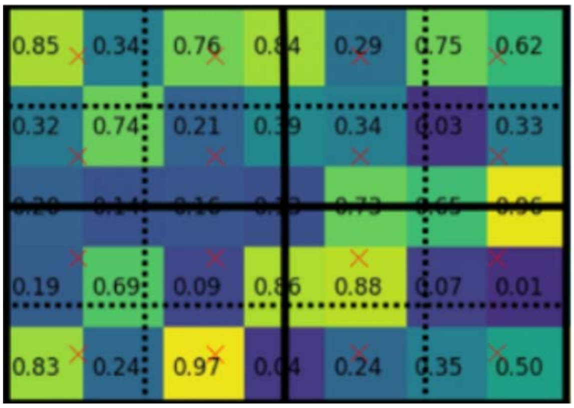
Step2: Map the region of proposal to the original image without rounding up



Step3: Divide the mapped feature map into 2*2 sections

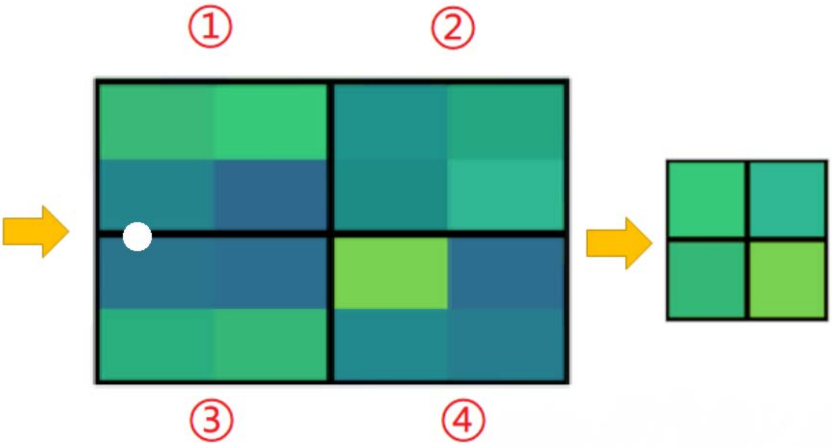


Step4: Then select 4 sampling points in each section, and then perform bilinear interpolation on each point in each section through bilinear interpolation, 4 points will get 4 values, and then perform max pooling , Get the value of each section



Step5: Finally get a 2*2 feature map,

for each small area (①, ②, ③, ④), there will be 4 44 such values. Take the maximum of these 4 44 values as the value of each small area (①, ②, ③, ④) . In this way, 4 44 values of 4 44 small areas can be obtained as the final feature map output result



Bilinear interpolation

Bilinear interpolation is also called bilinear interpolation. Mathematically, bilinear interpolation is a linear interpolation extension of an interpolation function with two variables. Its core idea is to perform linear interpolation in two directions.

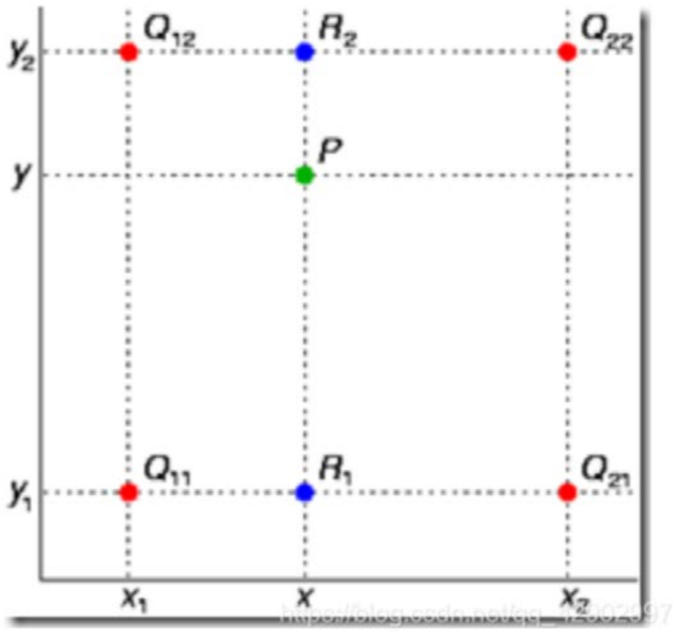
Four pixels: $Q_{11}, Q_{12}, Q_{21}, Q_{22}$

The coordinates of the four pixels:$(x_1, y_1), (x_1, y_2), (x_2, y_1)$, (x_2, y_2)

Pixel value of four pixels: $f(Q_{11}), f(Q_{12}), f(Q_{21}), f(Q_{22})$

Two points inserted by horizontal interpolation R_1, R_2 , the coordinates are$(x, y_1), (x, y_2)$

A point P of the longitudinal interpolation pair is inserted, and its coordinates are(x, y)



Interpolation method:

- Insert horizontally first, then vertically
- Insert vertically first, then horizontally

calculation process

First calculate the horizontal interpolation, the process of obtaining R_2 from Q_{12} Q_{22}

$$\frac{f(Q_{22})-f(Q_{12})}{x_2-x_1} \approx \frac{f(Q_{22})-f(R_2)}{x_2-x}$$

Cross multiply:

$$\big(f(Q_{22})-\ f(Q_{12})\big)\times(x_2-x)\approx(f(Q_{22})-f(R_2))\times(x_2-x_1)$$

Simplification:

$$f(R_2)\approx\frac{x_2-x}{x_2-x_1}f(Q_{12})+\frac{x-x_1}{x_2-x_1}f(Q_{22})$$

Similarly, the longitudinal difference R_1 can be obtained

$$f(R_1)\approx\frac{x_2-x}{x_2-x_1}f(Q_{11})+\frac{x-x_1}{x_2-x_1}f(Q_{21})$$

Combine R_1 R_2 to interpolate P in the y direction

$$\frac{f(R_2)-f(R_1)}{y_2-y_1} \approx \frac{f(R_2)-f(R_1)}{y_2-\mathrm{y}}$$

Simplification:

$$f(P) \approx \frac{y_2-\mathrm{y}}{y_2-y_1} f(R_1) + \frac{\mathrm{y}-y_1}{y_2-y_1} f(R_2)$$

Bring the obtained simplification result of $f(R_1)\,f(R_2)$ into the formula of $f(P)$:

$$\begin{aligned} f(P) \approx & \frac{f(Q_{11})}{(x_2-x_1)(y_2-y_1)}(x_2-x)(y_2-\mathrm{y}) \\ & + \frac{f(Q_{21})}{(x_2-x_1)(y_2-y_1)}(x-x_1)(y_2-\mathrm{y}) \\ & + \frac{f(Q_{12})}{(x_2-x_1)(y_2-y_1)}(x_2-x)(\mathrm{y}-y_1) \\ & + \frac{f(Q_{22})}{(x_2-x_1)(y_2-y_1)}(x-x_1)(\mathrm{y}-y_1) \end{aligned}$$

Therefore, the pixel value of point P can be obtained in the end, no matter whether the horizontal interpolation or vertical interpolation is performed first, the pixel value of the last point P is the same

Loss function（FAST）

$$L(\{p_i\},\{u_i\}) = \frac{1}{N_{cls}} \sum_t L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_t p_i^* L_{reg}(t_i, t_i^*)$$

Following the definition of multi-task loss and minimizing the objective function, Fast R-CNN unifies the category output task and the candidate box regression task. There are two loss functions: classification loss and regression loss.

p_i: Anchor[i] predicted classification probability

When Anchor[i] is a positive sample, $p_i^* = 1$; When Anchor[i] is a negative sample, $p_i^* = 0$

t_i: The parameterized coordinates of the Bounding Box predicted by Anchor[i]

t_i^*: Parameterized coordinates of Bounding Box of Ground Truth of Anchor[i]

Parameterized coordinates of Bounding box:

$$t_x = \frac{x - x_a}{\omega_a}, t_y = \frac{y - y_a}{h_a}$$
$$t_\omega = \log\left(\frac{\omega}{\omega_a}\right), t_h = \log\left(\frac{h}{h_a}\right)$$
$$t_x^* = \frac{x^* - x_a}{\omega_a}, t_y^* = \frac{y^* - y_a}{h_a}$$
$$t_\omega^* = \log\left(\frac{\omega^*}{\omega_a}\right), t_h^* = \log\left(\frac{h^*}{h_a}\right)$$

N_{cls}:mini-batch size

N_{reg}:number of Anchor Location

L_{reg}: $L_{reg}(t_i, t_i^*) = R(t_i - t_i^*)$, R is Smooth L1 founction

$p_i^* L_{reg}(t_i, t_i^*)$ means to return to the Bounding Box only when the sample is positive

Smooth L1 loss:

$$Smooth_{L1}(x) = \begin{cases} 0.5 * x^2 & |x| < 1 \\ |x| - 0.5 & other \end{cases}$$

$$\frac{dSmooth_{L1}(x)}{dx} = \begin{cases} x & |x| < 1 \\ \pm 1 & other \end{cases}$$

SmoothL1 avoids the defects of L1 and L2 loss. When X is small, the gradient to X will also become smaller; and when X is large, the absolute value of the gradient to X reaches the upper limit 1, which will not be due to the gradient of the predicted value. It is very large and causes training instability.

L_{cls}: Is the log loss of the two columns

$$L_{cls}(p_i, p_i^*) = -\log\left[p_i p_i^* + (1 - p_i^*)(1 - p_i)\right]$$

λ: Weight balance parameter

Loss function（MASK）

Since this report only discusses FAST R-CNN, MASK R-CNN loss function will only be introduced here and will not be described in detail.

There are a total of five loss functions in Mask RCNN, which are the two losses of the rpn network, the two losses of mrcnn, and the loss function of the mask branch. The first four loss functions are the same as those of fasterrcnn, and the final mask loss function uses the mask branch to have K*m^2 output for each RoI. K (number of categories) binary masks with a resolution of m * m. Lmask is the average binary cross-entropy loss. For a RoI belonging to the kth category, Lmask only considers the kth mask (other mask inputs will not contribute to the loss function). Such a definition would allow a mask to be generated for each category, and there would be no inter-class competition.

- rpn classification loss

The cross entropy rpn_match is related to GT, the foreground is 1 and the background is 0; rpn_class_logits is generated in rpn_graph, which is the feature map Reshape to [batch,anchors, 2] but there is no value activated by softmax. rpn regression loss SmoothL1 target_bbox is GT rpn_match is related to GT, foreground is 1 and background is 0; rpn_bbox is the value of feature map Reshape to [batch, anchors, 4] generated in rpn_graph.

$$\mathrm{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

- MASK R-CNN classification loss cross entropy

target_class_ids: target category ID GT

pred_class_logits: the feature map is obtained by the head network connected to the fully connected layer, the predicted category ID

active_class_ids : actual category ids 80 categories

actual use target_class_ids and pred_class_logits to calculate the cross entropy loss active_class_ids is used to eliminate predictions that are not in the image The predicted loss of the category in the category.

- MASK R-CNN regression loss SmoothL1

target_bbox: is the GT box

target_class_ids: the category ID corresponding to the GT box

pred_bbox :is the prediction box obtained from the feature map through the head network convolution

- MASK R-CNN loss mask binary cross entropy

Lmask is the loss function on the mask branch, the output size is K*m*m, and its encoding resolution is K binary masks of m*m, that is, each of the K categories corresponds to a binary mask. For each image the prime uses the sigmoid function, and Lmask is the average binary cross-entropy loss. RoI's groundtruth the category is k, Lmask is only defined on the kth Mask, and the remaining masks have no effect on it (That is to say, during training, although each point will have K binary masks, there is only one A k-type mask contributes to the loss, and this k value is predicted by the classification branch). Mask-RCNN has no inter-class competition, because other classes do not contribute to the loss. mask branch for each all categories are predicted, depending on the classification layer to select the output mask (at this time the size should be m*m, which is predicted when a category comes out, you only need to output the mask corresponding to that category), use the general method of FCN. It uses softmax and multiple cross-entropy losses for each pixel, and there will be inter-class competition. Binary crossover. Entropy will make each type of mask not compete with each other, rather than comparing with other types of masks.

target_mask: GT mask

target_class_ids: the category ID corresponding to the GT box

mrcnn_mask :The mask trained by the image

3.2

Please describe the object detection performance metric, mAP (Mean Average Precision), and explain why it can well reflect the object detection accuracy.

mAP

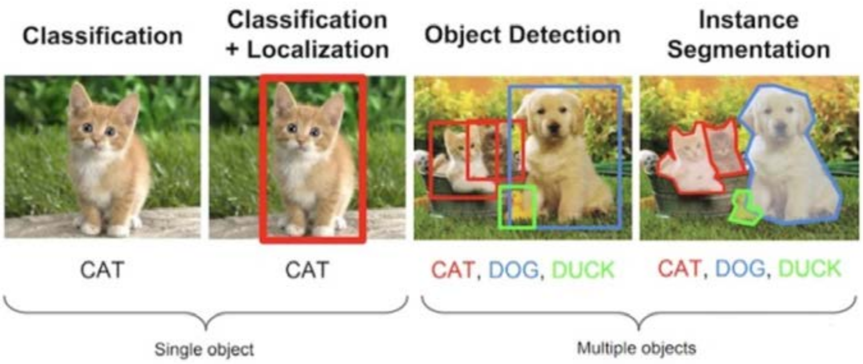
For most common problems solved using machine learning, there are usually multiple models available. Each model has its own uniqueness and performance varies with factors.

The performance of each model is evaluated on the "verification/test" data set. Performance measurement uses various statistics such as accuracy, precision, recall, etc. The selected statistics are usually specific to specific application scenarios and use cases. For each application scenario, it is very important to choose a metric that can objectively compare models.

Before explaining mAP, we first define the target detection problem.

In the object detection problem, given an image, find the objects it contains, find their positions and classify them. The target detection model is usually trained on a specific set of classes, so the model will only locate and classify those classes in the image. In addition, the position of the object is usually represented by a rectangular bounding box.

Therefore, target detection involves the location and classification of objects in the image.



The Mean Average Precision described below is particularly suitable for algorithms that simultaneously predict the location and category of objects. Therefore, it can be seen from this figure that it is very useful for evaluating positioning models, target detection models and segmentation models.

Evaluate the target detection model

Why is mAP?

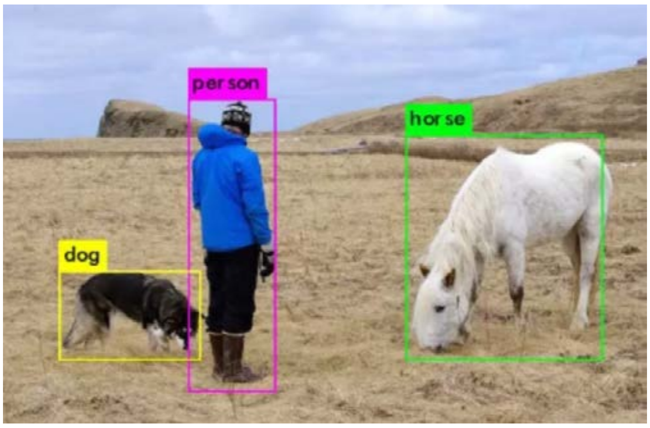
Each picture in the target detection problem may contain some different types of objects. As mentioned earlier, the object classification and localization performance of the model needs to be evaluated. Therefore, the standard index precision used for image classification problems cannot be directly applied here. This is why mAP is needed.

About Ground Truth

For any algorithm, the evaluation index needs to know the ground truth (true label) data. We only know the ground truth of the training, validation, and test data sets. For target detection, ground truth includes the category of objects in the image and the true bounding box of each object in the image.

mAP meaning and calculation

The trained target detection model will give a large number of prediction results, but most of the prediction values will have a very low confidence score, so we only consider those prediction results with a confidence score higher than a certain threshold. The original image is sent to the trained model. After the confidence threshold is screened, the target detection algorithm gives the prediction result with bounding box:

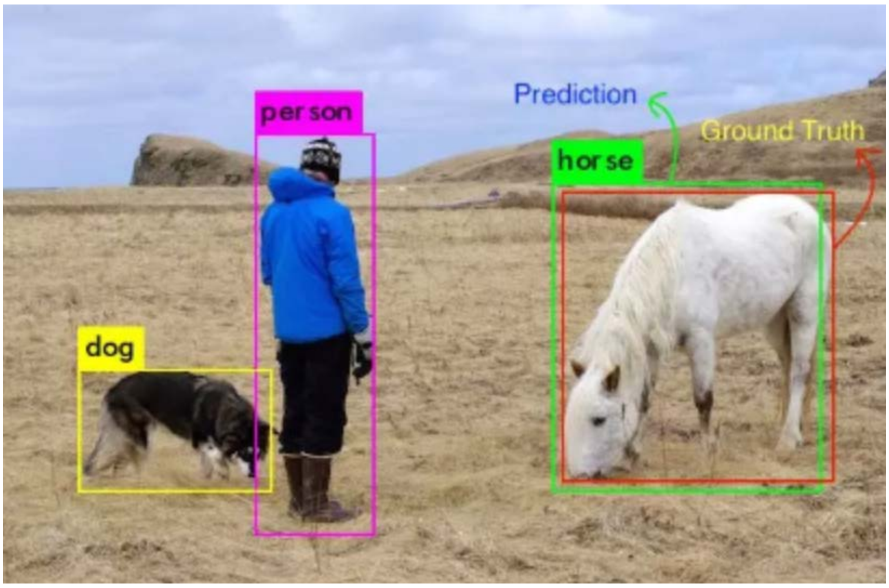


Now, since we humans are target detection experts, we can know that these detection results are roughly correct. But how do we quantify it? We first need to judge the correctness of each test. IoU (Intersection over Union) is used here, which can be used as a metric to evaluate the correctness of the bounding box.

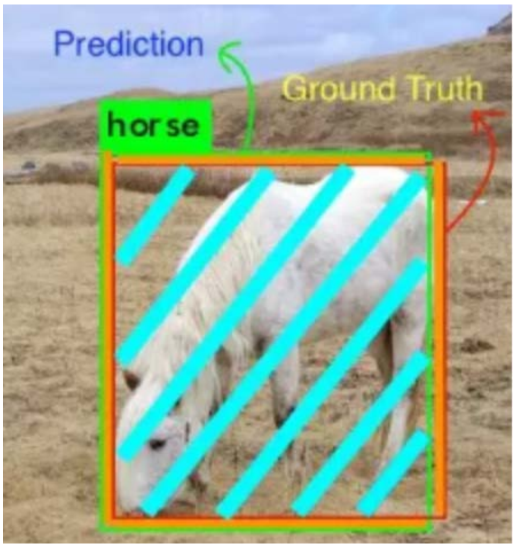
IoU

The intersection ratio is the ratio between the intersection and union of the predicted bounding box and the reference bounding box. To obtain the values of intersection and union, we first overlay the predicted bounding box on the reference bounding box.

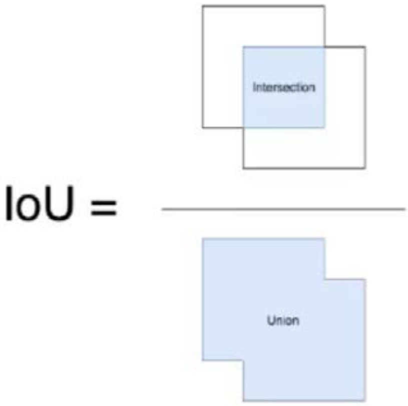
Now for each category, the overlapping part of the predicted bounding box and the reference bounding box is called the intersection, and all areas spanned by the two bounding boxes are called the union.



For each class, the area where the prediction box and the ground truth overlap is the intersection, and the total area across is the union. The intersection and union of the horse class are shown in the following figure (this example has a large intersection):



The blue-green part is the intersection, and the union also includes the orange part. Then, IoU can be calculated as follows:



Identify the correct test results and calculate precision and recall

In order to calculate precision and recall, as with all machine learning problems, we must identify True Positives, False Positives, True Negatives, and False Negatives (false negatives).

In order to get True Positives and False Positives, we need to use IoU. By calculating IoU, we can determine whether a detection result (Positive) is correct (True) or wrong (False). The most commonly used threshold is 0.5, that is, if IoU>0.5, it is considered True Positive, otherwise it is considered False Positive.

To calculate Recall, we need the number of Negatives. Since we did not predict that every part of the object in the picture is considered as negative, it is difficult to calculate True Negatives. But we can only calculate False Negatives, which are objects that our model missed. Another factor that needs to be considered is the confidence of each test result given by the model. By changing the confidence threshold, we can change whether a prediction box is Positive or Negative, that is, change the positive and negative of the predicted value (not the true positive and negative of the box, but the predicted positive and negative).

Basically, all predictions (Box + Class) above the threshold are considered Positives, and those below this value are Negatives. For each picture, the ground truth data will give the actual number of objects in each category in the picture. We can calculate the IoU value of each Positive prediction box and ground truth, and take the largest IoU value, and consider that the prediction box has detected the ground truth with the largest IoU. Then according to the IoU threshold, we can calculate the number of correct detection values (True Positives, TP) and the number of false positives (False Positives, FP) for each category in a picture. Based on this, the precision of each category can be calculated:

$$precision = \frac{TP}{TP + FP}$$

Now that we have got the number of correct predictions (True Positives), it is easy to calculate the number of missing objects (False Negatives, FN). Based on this, Recall can be calculated (in fact, the denominator can be the total number of ground truth):

$$recall = \frac{TP}{TP + FN}$$

PR curve

We certainly hope that the higher the test result P, the better, and the higher R the better, but in fact the two are contradictory in some cases. For example, in extreme cases, we only detect one result and it is accurate, then Precision is 100%, but Recall is very low; and if we return all results, Recall must be very large, but Precision is very low .

Therefore, you need to judge whether you want P to be higher or R to be higher in different situations. If you are doing experimental research, you can draw a Precision-

Recall curve to help the analysis.

Here we give a simple example. Assume that there are five objects to be detected in our data set. Our model gives 10 candidate boxes. We sort the candidate boxes according to the confidence level given by the model from high to low.

Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0	0.4
3	False	0.67	0.4
4	False	0.5	0.4
5	False	0.4	0.4
6	True	0.5	0.6
7	True	0.57	0.8
8	False	0.5	0.8
9	False	0.44	0.8
10	True	0.5	1.0

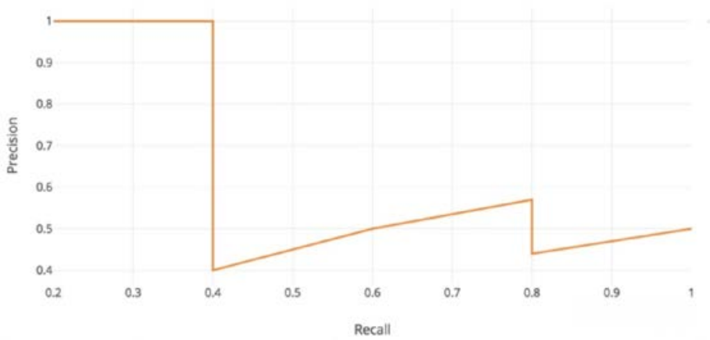
The second column of the table indicates whether the prediction of the candidate frame is correct (that is, whether there is an object to be detected and the iou value of the candidate frame is greater than 0.5) The third and fourth columns indicate when the confidence of the candidate frame of the row is the threshold , Precision and Recall values. We use the third line of the table to calculate:

$$TP = 2 \quad FP = 1 \quad TN = 3$$

$$Precision = \frac{2}{2+1} = 0.67$$

$$Recall = \frac{2}{2+3} = 0.4$$

From the above table, taking Recall as the horizontal axis and Precision as the vertical axis, we can get the PR curve. We will find that the value of Precision and Recall is negatively correlated and fluctuates up and down in a local area.



AP(Average Precision)

As the name suggests, AP is the average accuracy. Simply put, it is the average of the Precision value on the PR curve. For the pr curve, we use integral to calculate.

$$AP = \int_0^1 p(r)dr$$

In practical applications, we do not directly calculate the PR curve, but smooth the PR curve. That is, for each point on the PR curve, the value of Precision takes the value of the largest Precision on the right side of the point.

mAP

$$MeanAveragePrecision_{IoU=threshold} = \frac{\sum AveragePrecision_C}{N(Classes)}$$

$$MeanAveragePrecision = \frac{\sum MeanAveragePrecision_{IoU=threshold}}{N_{(Ten\ IoU)}}$$

Average precision mean = the sum of the average precision values of all categories/the number of all categories

Therefore, the mean average precision is the mean of the average precision of all categories in the data set.

Why choose mAP?

In target detection, each picture may contain multiple targets in multiple categories. Therefore, the evaluation of the target detection model needs to evaluate the positioning and classification effects of the model at the same time.

Therefore, the precision index often used in image classification problems cannot be directly used for target detection, and mAP is more effective.

There are at least two variables that affect Precision and Recall, namely IoU and confidence threshold. IoU is a simple geometric metric that can be easily standardized. For example, the IoU threshold used in the PASCAL VOC competition is 0.5, while the calculation of mAP in the COCO competition is more complicated. It calculates a series of IoU thresholds (0.05 to 0.95). mAP. However, the confidence level varies greatly in different models. It is possible that using 0.5 confidence level in model A is equivalent to using 0.8 confidence level in model B, which will cause the precision-recall curve to change..

3.3

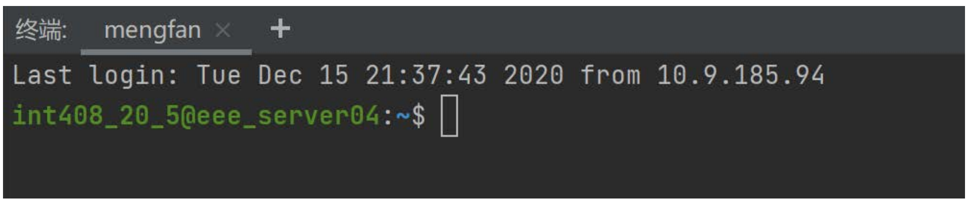
Please train (or fine-tune) and test the framework on one of the existing pedestrian detection datasets, and report the final AP performance that you have achieved. The dataset in this lab is PennFudanPed [3]. Please also report some pedestrian detection examples by including the images and bounding boxes.

Train

Step1: Use Pycharm to link the server

All configuration information has been configured in PyCharm's new SSH in advance

Then link the serve



Step2: Switch to your own folder directory,

The directory address is: /Data_HDD/INT408_20/INT408_5/mengfan

```
终端:   mengfan ×  +
Last login: Tue Dec 15 21:37:43 2020 from 10.9.185.94
int408_20_5@eee_server04:~$ cd /Data_HDD/INT408_20/INT408_5/mengfan
int408_20_5@eee_server04:/Data_HDD/INT408_20/INT408_5/mengfan$
```

Step3: Set up the environment

I have copied the environment configuration file to:

/home/int408_20_5/.conda/envs/mengfan

```
int408_20_5@eee_server04:/Data_HDD/INT408_20/INT408_5/mengfan$ cd /home/int408_20_5/.conda/envs/mengfan
int408_20_5@eee_server04:~/.conda/envs/mengfan$ dir
bin  compiler_compat  conda-meta  include  info  lib  maskrcnn_resnet50_fpn_coco-bf2d0c1e.pth  share  ssl  x86_64-conda_cos6-linux-gnu
int408_20_5@eee_server04:~/.conda/envs/mengfan$
```

View environment: conda info –envs

```
int408_20_5@eee_server04:/Data_HDD/INT408_20/INT408_5/mengfan$ conda info --envs
# conda environments:
#
int408ass1               /home/int408_20_5/.conda/envs/int408ass1
int408ass2               /home/int408_20_5/.conda/envs/int408ass2
mengfan                  /home/int408_20_5/.conda/envs/mengfan
zhicai                   /home/int408_20_5/.conda/envs/zhicai
base                  *  /usr/local/anaconda3
```

Activate the environmenr

```
int408_20_5@eee_server04:/Data_HDD/INT408_20/INT408_5/mengfan$ source activate mengfan
(mengfan) int408_20_5@eee_server04:/Data_HDD/INT408_20/INT408_5/mengfan$
```

Step4:train

I have copied the train code to my own folder

```
(mengfan) int408_20_5@eee_server04:/Data_HDD/INT408_20/INT408_5/mengfan/INT408_a2$ dir
PennFudanPed  \\model_mengfan.pkl  bounding.py   coco_utils.py  train_frcnn.py  utils.py
\\model.pkl   __pycache__          coco_eval.py  engine.py      transforms.py   vis.py
(mengfan) int408_20_5@eee_server04:/Data_HDD/INT408_20/INT408_5/mengfan/INT408_a2$
```

Run the train_frcnn.py

```
(mengfan) int408_20_5@eee_server04:/Data_HDD/INT408_20/INT408_5/mengfan/INT408_a2$ python train_frcnn.py
```

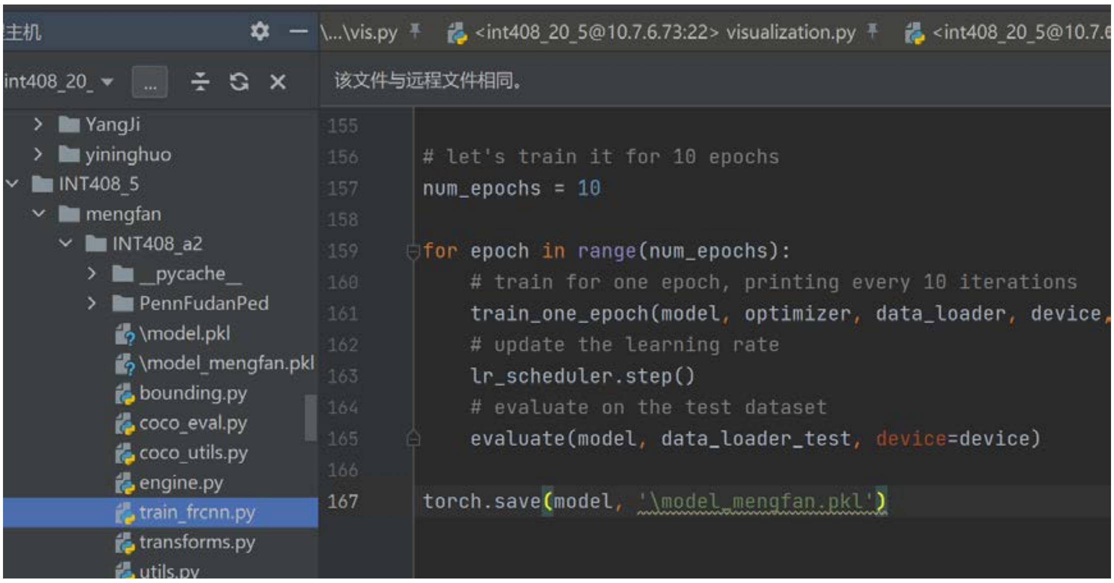
Result:

```
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.991
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.956
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 ] = 0.590
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.834
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.379
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.868
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.868
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 ] = 0.812
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.872
oU metric: segm
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.772
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.991
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.920
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 ] = 0.491
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.780
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.350
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.809
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.809
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 ] = 0.750
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.814
mengfan) int408_20_5@eee_server04:/Data_HDD/INT408_20/INT408_5/mengfan/INT408_a2$
```

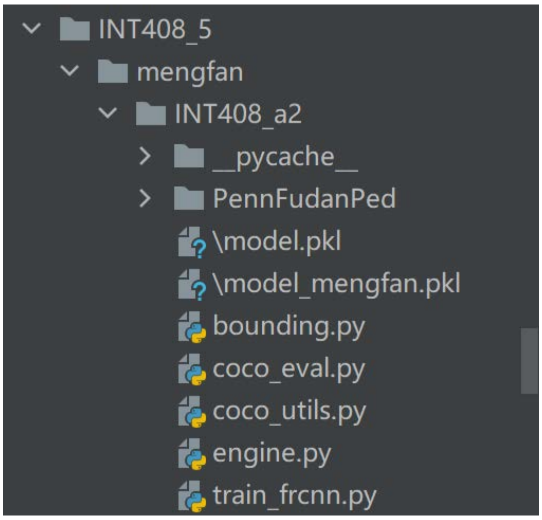
Test

Step1: Detection target

First save the model, save the tested model. named: \model_mengfan.pkl



After execution, a saved model will appear under the root directory, so you can directly use this model in the test.



Follow the experimental instructions to write vis.py

```
from PIL import Image
......
dataset_test = PennFudanDataset('PennFudanPed', get_transform(train=False))
img, _ = dataset_test[0]
model = torch.load('\model.pkl')
model.eval()
with torch.no_grad():
    prediction = model([img.to(device)])
```

import some libraries

Definition PennFudanDataset

<int408_20_5@10.7.6.73:22> \...\vis.py   <int408_20_5@10.7.6.73:22> \...\visual.py

文件与远程文件相同。

0 项结果

```
class PennFudanDataset(torch.utils.data.Dataset):
    def __init__(self, root, transforms=None):
        self.root = root
        self.transforms = transforms
        # load all image files, sorting them to
        # ensure that they are aligned
        self.imgs = list(sorted(os.listdir(os.path.join(root, "PNGImages"))))
        self.masks = list(sorted(os.listdir(os.path.join(root, "PedMasks"))))

    def __getitem__(self, idx):
        # load images ad masks
        img_path = os.path.join(self.root, "PNGImages", self.imgs[idx])
        print(img_path)
        mask_path = os.path.join(self.root, "PedMasks", self.masks[idx])
        img = Image.open(img_path).convert("RGB")
        # note that we haven't converted the mask to RGB,
        # because each color corresponds to a different instance
        # with 0 being background
        mask = Image.open(mask_path)

        mask = np.array(mask)
        # instances are encoded as different colors
        obj_ids = np.unique(mask)
        # first id is the background, so remove it
```

Use the data set, because it is test, so select the label train=False

```
dataset_test = PennFudanDataset('PennFudanPed', get_transform(train=False))
```

Choose a picture to test

```
img, _ = dataset_test[8]
```

Quote the trained model

```
model = torch.load('\model_mengfan.pkl')
model.eval()
```

Model.eval() is very important. Be sure to specify train/eval for the instantiated model. When eval(), the framework will automatically fix BN and DropOut instead of taking the average, but use the trained value, otherwise If the batch_size of the test is too small, it is easy to be caused by the BN layer to cause great color distortion in the generated image.

print

```
with torch.no_grad():
    prediction = model([img.to(device)])
    print(prediction)
```

torch.no_grad() is a context manager, the part wrapped by this statement will not track the gradient.

Result:

```
(mengfan) int408_20_5@eee_server04:/Data_HDD/INT408_20/INT408_5/mengfan/INT408_a2$ python vis.py
PennFudanPed/PNGImages/FudanPed00001.png
[{'boxes': tensor([[159.5585, 174.8557, 300.9803, 429.1591],
        [417.8943, 171.5250, 532.7275, 490.3836]], device='cuda:0'), 'labels': tensor([1, 1], device='cuda:0'), 'scores': tensor([0.9993, 0.9988], devic
e='cuda:0'), 'masks': tensor([[[[0., 0., 0.,  ..., 0., 0., 0.],
          [0., 0., 0.,  ..., 0., 0., 0.],
          [0., 0., 0.,  ..., 0., 0., 0.],
          ...,
          [0., 0., 0.,  ..., 0., 0., 0.],
          [0., 0., 0.,  ..., 0., 0., 0.],
          [0., 0., 0.,  ..., 0., 0., 0.]]],


        [[[0., 0., 0.,  ..., 0., 0., 0.],
          [0., 0., 0.,  ..., 0., 0., 0.],
          [0., 0., 0.,  ..., 0., 0., 0.],
          ...,
          [0., 0., 0.,  ..., 0., 0., 0.],
          [0., 0., 0.,  ..., 0., 0., 0.],
          [0., 0., 0.,  ..., 0., 0., 0.]]]], device='cuda:0')}]
```

There are two targets identified here, let’s compare the original image FudanPed00001.png to see if the recognition is accurate.

```
(mengfan) int408_20_5@eee_server04:/Data_HDD/INT408_20/INT408_5/mengfan/INT408_a2$ python vis.py
PennFudanPed/PNGImages/FudanPed00002.png
[{'boxes': tensor([[ 69.6159,  95.8164, 189.2390, 380.0176],
        [158.6682,  88.0225, 228.9424, 362.6456],
        [130.2889,  99.3562, 207.5127, 368.2436]], device='cuda:0'), 'labels': tensor([1, 1, 1], device='cuda:0'), 'scores': tensor([0.9990, 0.9461, 0.4
457], device='cuda:0'), 'masks': tensor([[[[0., 0., 0.,  ..., 0., 0., 0.],
          [0., 0., 0.,  ..., 0., 0., 0.],
          [0., 0., 0.,  ..., 0., 0., 0.],
          ...,
          [0., 0., 0.,  ..., 0., 0., 0.],
          [0., 0., 0.,  ..., 0., 0., 0.],
          [0., 0., 0.,  ..., 0., 0., 0.]]],


        [[[0., 0., 0.,  ..., 0., 0., 0.],
          [0., 0., 0.,  ..., 0., 0., 0.],
          [0., 0., 0.,  ..., 0., 0., 0.],
          ...,
          [0., 0., 0.,  ..., 0., 0., 0.],
          [0., 0., 0.,  ..., 0., 0., 0.],
          [0., 0., 0.,  ..., 0., 0., 0.]]],
```

There are two targets identified here, let's compare the original image FudanPed00002.png to see if the recognition is accurate

```
(mengfan) int408_20_5@eee_server04:/Data_HDD/INT408_20/INT408_5/mengfan/INT408_a2$ python vis.p
PennFudanPed/PNGImages/FudanPed00006.png
[{'boxes': tensor([[205.6853, 102.3433, 347.4590, 382.6847],
        [  1.2773, 107.5019,  91.4248, 385.9730],
        [ 31.7128,  95.6408,  87.0316, 282.5471]], device='cuda:0'), 'labels': tensor([1, 1, 1]
624], device='cuda:0'), 'masks': tensor([[[[0., 0., 0.,  ..., 0., 0., 0.],
          [0., 0., 0.,  ..., 0., 0., 0.],
          [0., 0., 0.,  ..., 0., 0., 0.],
          ...,
```

There are two targets identified here, let's compare the original image FudanPed00006.png to see if the recognition is accurate

Step2: Draw border

Read the data Boxes and Scores from the Step1

```
bounding_boxs=prediction[0]['boxes']
scores=prediction[0]['scores']
```

Use function to draw box

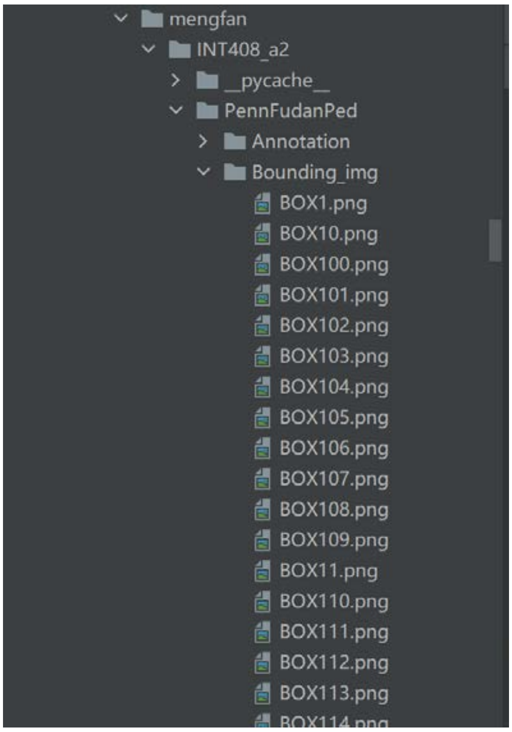
```
draw=ImageDraw.Draw(img)
```

Set threshold 0.9,

```
if score>0.90:
    draw.rectangle(xy=(int(bounding_box[0]),int(bounding_box[1]),int(bounding_box[2]),int(bounding_box[3])),fill=None,outline='
    draw.text((int(bounding_box[0])+7,int(bounding_box[1])+7),str(round(float(score),4)),fill=(0,255,0))
else:
    draw.rectangle(xy=(int(bounding_box[0]), int(bounding_box[1]), int(bounding_box[2]), int(bounding_box[3])),fill=None, outli
    draw.text((int(bounding_box[0]) + 7, int(bounding_box[1]) + 7), str(round(float(score), 4)),fill=(255, 0, 0))
```

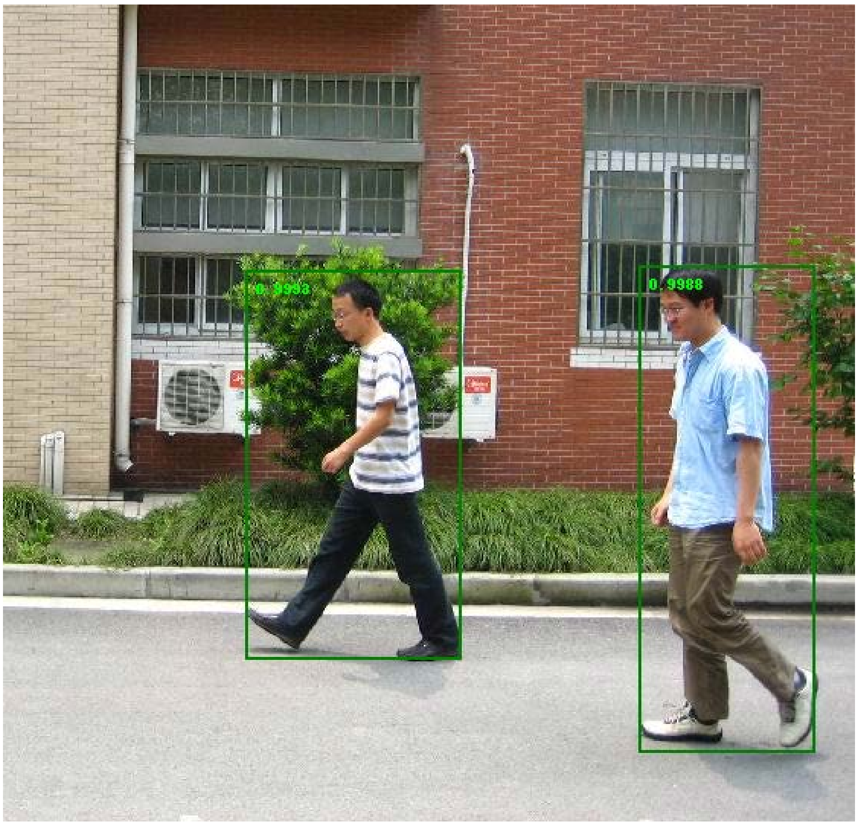
Result:

```
(mengfan) int408_20_5@eee_server04:/Data_HDD/INT408_20/INT408_5/mengfan/INT408_a2$ python visual.py
```



Step3: Detection accuracy

We randomly select five for verification



FudanPed000001

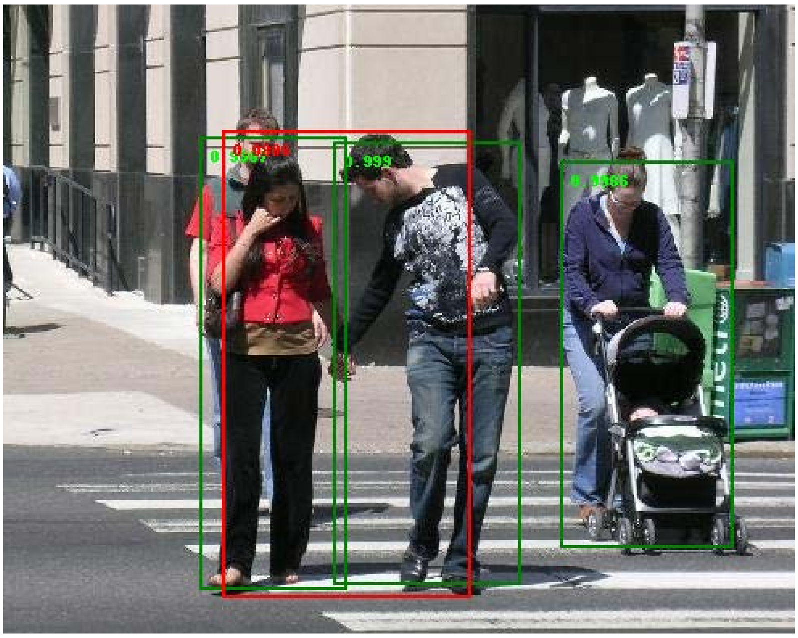
There are two targets in the picture, two are identified, Score respectively:0.9993 and 0.9988

FudanPed00010

However, there is only one goal left after IoU screening, one are identified, IoU respectively:0.9986

FudanPed00100

There are two targets in the picture, two are identified, Score respectively:0.9994 and 0.9988

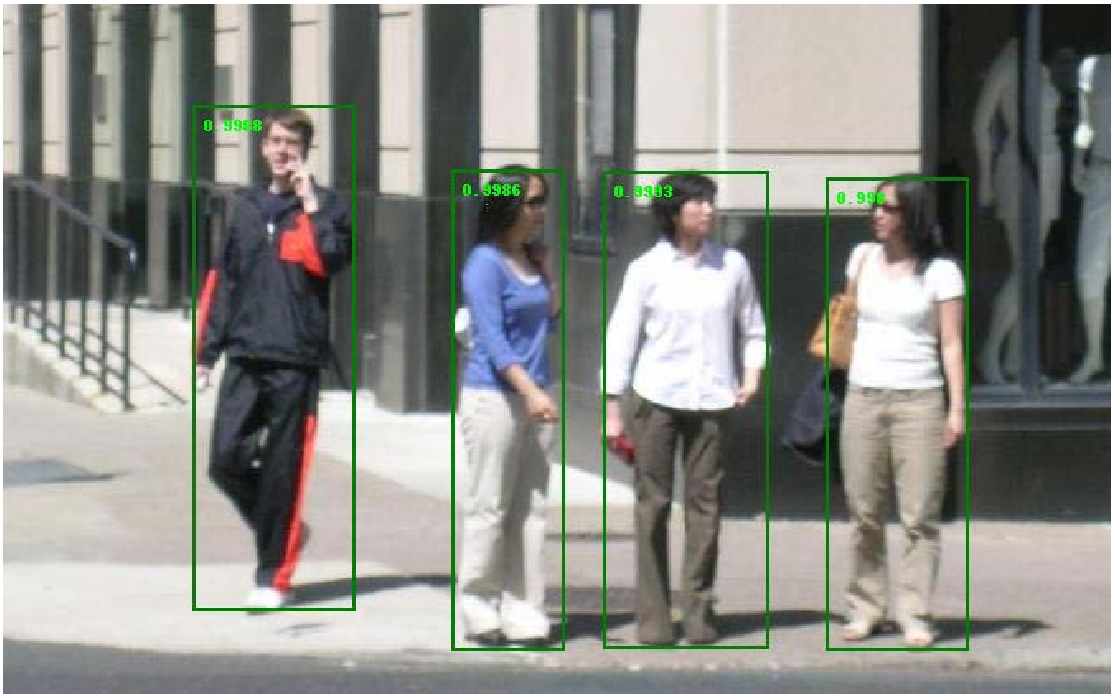


FudanPed00101

The picture has four labels, but because of the overlap, after IoU screening, only three targets are left. Score is:0.9987,0.999,0.9966 ,but there is a wrong label, Score is:0.0886

FudanPed00102

The picture has two labels, but because of the overlap, after IoU screening, only one targets are left. Score is:0.9984, but there is a wrong label, Score is:0.05



FudanPed00103

There are four targets in the picture, all are identified, Score respectively:0.9988,0.9986,0.9993,0.998.

Calculate IoU

For the calculation of IoU, I have made some attempts, but the effect is not very good. I will continue to improve in the future research life.

Code:

```
def calculate_iou(box1, box2):
    xmin1, ymin1, xmax1, ymax1 = box1
    xmin2, ymin2, xmax2, ymax2 = box2
    s1 = (xmax1 - xmin1) * (ymax1 - ymin1)
    s2 = (xmax2 - xmin2) * (ymax2 - ymin2)
    xmin = max(xmin1, xmin2)
```

```
    ymin = max(ymin1, ymin2)
    xmax = min(xmax1, xmax2)
    ymax = min(ymax1, ymax2)
    w = max(0, xmax - xmin)
    h = max(0, ymax - ymin)
    area = w * h
    iou = area / (s1 + s2 - area)
    return iou
```

3.4

Propose your own method to further improve the pedestrian detection performance or reduce the model size based on the Mask R-CNN framework, and compare different methods with the performance you obtained and explain why.

Case1: change some scales

I will modify some scales to see different results

a) Hidden layer number

Epoch 10

Hidden layer number	*256(original)*	*64*	*128*	*512*
IoU metric segm: Average Recall medium	*0.700*	*0.750*	*0.688*	*0.725*

Hidden layer number	*256(original)*	*64*	*128*	*512*
FudanPed00103 object1	*Score:0.9987*	*Score:0.9991*	*Score:0.9988*	*Score:0.9984*
FudanPed00103 object2	*Score:0.9983*	*Score:0.9989*	*Score:0.9985*	*Score:0.9988*
FudanPed00103 object3	*Score:0.9993*	*Score:0.9995*	*Score:0.999*	*Score:0.9992*
FudanPed00103 object4	*Score:0.9985*	*Score:0.999*	*Score:0.9983*	*Score:0.9987*

It can be seen that when hidden layer number=64, the effect is improved

b) Learning rate

Epoch 10

Learning rate	*0.005(original)*	*0.001*	*0.003*	*0.007*
IoU metric segm: Average Recall medium	*0.700*	*0.725*	*0.700*	*0.750*

Learning rate	*0.005(original)*	*0.001*	*0.003*	*0.0005*
FudanPed00103 object1	*Score:0.9987*	*Score:0.9984*	*Score:0.9984*	*Score:0.9977*
FudanPed00103 object2	*Score:0.9983*	*Score:0.9983*	*Score:0.9983*	*Score:0.9976*
FudanPed00103 object3	*Score:0.9993*	*Score:0.9986*	*Score:0.9986*	*Score:0.998*
FudanPed00103 object4	*Score:0.9985*	*Score:0.9979*	*Score:0.9979*	*Score:0.9974*

Case2: change the backbone

Change the maskrcnn_resnet50_fpn to mobilenet_v2

Method one

Code:

```
import torchvision
from torchvision.models.detection import FasterRCNN
from torchvision.models.detection.rpn import AnchorGenerator

# load a pre-trained model for classification and return
```

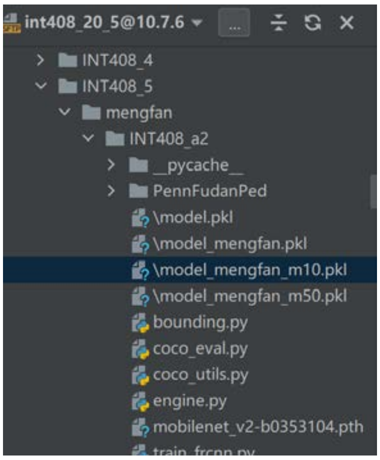
```python
# only the features
backbone = torchvision.models.mobilenet_v2(pretrained=True).features
# FasterRCNN needs to know the number of
# output channels in a backbone. For mobilenet_v2, it's 1280
# so we need to add it here
backbone.out_channels = 1280

# let's make the RPN generate 5 x 3 anchors per spatial
# location, with 5 different sizes and 3 different aspect
# ratios. We have a Tuple[Tuple[int]] because each feature
# map could potentially have different sizes and
# aspect ratios
anchor_generator = AnchorGenerator(sizes=((32, 64, 128, 256, 512),),
                                   aspect_ratios=((0.5, 1.0, 2.0),))

# let's define what are the feature maps that we will
# use to perform the region of interest cropping, as well as
# the size of the crop after rescaling.
# if your backbone returns a Tensor, featmap_names is expected to
# be [0]. More generally, the backbone should return an
# OrderedDict[Tensor], and in featmap_names you can choose which
# feature maps to use.
roi_pooler = torchvision.ops.MultiScaleRoIAlign(featmap_names=[0],
                                                output_size=7,
                                                sampling_ratio=2)

# put the pieces together inside a FasterRCNN model
model = FasterRCNN(backbone,
                   num_classes=2,
                   rpn_anchor_generator=anchor_generator,
                   box_roi_pool=roi_pooler)
```

and a support file: mobilenet_v2-b0353104.pth



Result:

Train effect

resnet50_fpn

Epoch	*10*	*50*	*100*
IoU metric segm: Average Recall medium	*0.700*	*0.700*	*0.700*

mobilenet_v2

Epoch	*10*	*50*	*100*
IoU metric segm: Average Recall medium	*0.188*	*0.100*	*0.163*

Test effect

Epoch 10

	resnet50_fpn	*mobilenet_v2*
FudanPed00103 object2	*Score:0.9987*	*Score:0.9269*
FudanPed00103 object2	*Score:0.9983*	*Score:0.9734*
FudanPed00103 object3	*Score:0.9993*	*Score:0.9754*
FudanPed00103 object4	*Score:0.9985*	*Score:0.9635*

Epoch 50

	resnet50_fpn	*mobilenet_v2*
FudanPed00103 object2	*Score:0.9987*	*Score:0.9641*
FudanPed00103 object2	*Score:0.9984*	*Score:0.9823*
FudanPed00103 object3	*Score:0.9991*	*Score:0.9834*
FudanPed00103 object4	*Score:0.9986*	*Score:0.977*

Epoch 100

	resnet50_fpn	*mobilenet_v2*
FudanPed00103 object2	*Score:0.9985*	*Score:0.9765*
FudanPed00103 object2	*Score:0.9983*	*Score:0.9713*
FudanPed00103 object3	*Score:0.9993*	*Score:0.9868*
FudanPed00103 object4	*Score:0.9985*	*Score:0.9846*

Method two

Code:

```python
def get_instance_segmentation_model(num_classes):
    # load a pre-trained model for classification and return
    # only the features
    backbone = torchvision.models.mobilenet_v2(pretrained=True).features
    # FasterRCNN needs to know the number of
    # output channels in a backbone. For mobilenet_v2, it's 1280
    # so we need to add it here
    backbone.out_channels = 256

    # let's make the RPN generate 5 x 3 anchors per spatial
    # location, with 5 different sizes and 3 different aspect
    # ratios. We have a Tuple[Tuple[int]] because each feature
    # map could potentially have different sizes and
    # aspect ratios
    anchor_generator = AnchorGenerator(sizes=((32, 64, 128, 256, 512),),
                                       aspect_ratios=((0.5, 1.0, 2.0),))

    # let's define what are the feature maps that we will
    # use to perform the region of interest cropping, as well as
    # the size of the crop after rescaling.
    # if your backbone returns a Tensor, featmap_names is expected to
    # be [0]. More generally, the backbone should return an
    # OrderedDict[Tensor], and in featmap_names you can choose which
    # feature maps to use.
```

```
    roi_pooler = torchvision.ops.MultiScaleRoIAlign(featmap_names=['0'],
                                                    output_size=7,
                                                    sampling_ratio=2)

    backbone.add_module("conv3", torch.nn.Conv2d(1280, 256, kernel_size=(1, 1), stride=(1, 1), bias=False))
    # put the pieces together inside a FasterRCNN model
    model = MaskRCNN(backbone,
                     num_classes=2,
                     rpn_anchor_generator=anchor_generator,
                     box_roi_pool=roi_pooler)
    #model.add_module("conv3", torch.nn.Conv2d(1, 1, 256))
    return model
```

Result:

Averaged stats: model_time

	resnet50_fpn	*mobilenet_v2*
Epoch[9]	*0.0651*	*0.0294*

Conclusion

It can be seen that after resnet50_fpn changing to mobilenet_v2, the effect is not good, so in future research, we will continue to discuss the improvement of MASK R-CNN.

In addition to mobilenet_v2, we can also change to resnet18, resnext101_32x8d which will be further discussed in future research.

Reference

[1] He K, Gkioxari G, Dollár P, et al. Mask RCNN. arXiv e-prints, Article[J]. arXiv preprint arXiv:1703.06870, 2017.

[2] Tong Xiao, Shuang Li, Bochao Wang, Liang Lin, and Xiaogang Wang. Joint detection and identification feature learning for person search. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jul 2017.

[3] https://www.cis.upenn.edu/~jshi/ped_html/

[4] Sandler M, Howard A, Zhu M, et al. Mobilenetv2: Inverted residuals and linear bottlenecks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 4510-4520.

[5] CS231n: Convolutional Neural Networks for Visual Recognition Spring 2020 http://cs231n.stanford.edu/

[6] Girshick, R. (2015) 'Fast R-CNN'. http://arxiv.org/abs/1504.08083

[7] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4510-4520).

[8] mask rcnn-benchmark https://github.com/facebookresearch/maskrcnn-benchmark/blob/master/MODEL_ZOO.md