# INT408 Deep Learning in Computer Vision

Lab 2

Pedestrian detection using Mask R-CNN

# Overall

## 1 Introduction

This lab is to use the Pytorch software and Mask R-CNN framework [1] (Faster-RCNN + Instance Segmentation) for pedestrian detection. Pedestrian detection aims to predict bounding boxes of all the pedestrian instances in an image.

## 2 Report

Due : 20-Dev-2020

Q&A: 9-Dec-2020

# Report - Formal

Introduction

Method (q1,q2)
Mathmatic

Experience (q3,q4)
Screenshot/tables/Visulization

Conclution

(optional) Appendix

1.Please describe the 2 key components in the Fast R-CNN framework: the RoIPooling layer and the loss functions in the framework. (20%)

2.Please describe the object detection performance metric, mAP (Mean Average Precision), and explain why it can well reflect the object detection accuracy. (20%)

3.Please train(finetune) 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. (40%).

4.Propose your own method to further improve the pedestrian detection performance. (20%)

# Demo

## 1 Mask R-CNN on PennFudanPed

- **Environment**

conda info --envs

scp -r /Data_HDD/INT408_20/INT408_public/env_a2/int408ass2/
/home/your_account/.conda/envs

source activate int408ass2

- **Codes**

scp -r /Data_HDD/INT408_20/INT408_public/INT408_a2/
/Data_HDD/INT408_20/INT408_1/xiaoming/

cd /Data_HDD/INT408_20/INT408_1/xiaoming/

# Demo

- **Run**

export CUDA_VISIBLE_DEVICES=X

python train_frcnn.py

- **Visulization**

python vis.py

ifconfig(ipconfig for windows) - find your ID address

scp example.png lihui@10.8.203.189:./Desktop

# Things To Do

- **Model save**

**torch.save**(model, '\model.pkl')

- **Inference & load**

**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)])

```
prediction

[{'boxes': tensor([[ 61.7920,  35.8468, 196.2695, 328.1466],
         [276.3983,  21.7483, 291.1403,  73.4649],
         [ 79.1629,  42.9354, 201.3314, 207.8434]], device='cuda:0'),
  'labels': tensor([1, 1, 1], 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.]]],


         [[[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'),
  'scores': tensor([0.9994, 0.8378, 0.0524], device='cuda:0')}]
```

# Reference

**2 More Reference**

https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html

https://colab.research.google.com/github/pytorch/vision/blob/temp-tutorial/tutorials/torchvision_finetuning_instance_segmentation.ipynb