

Video Streaming and tracking

Homework 2 - Object Detection

Deadline: 2021/11/08 23:55

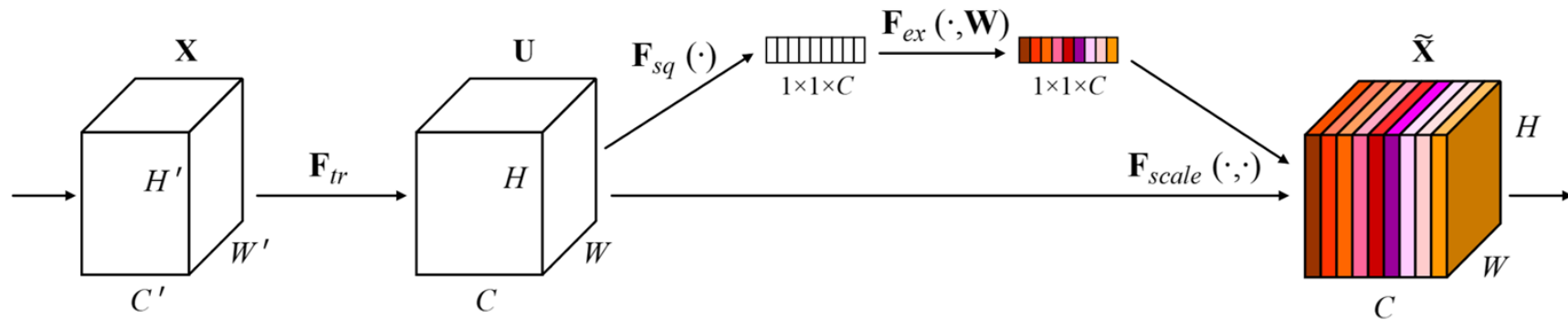
Outline

- Introduction
- Dataset
- Evaluation Metrics
- Grading Policy
- Hand in Rules

Introduction

- Train a neural network to do detection on our own dataset
- Model : object detection algorithms (Choose one)
 - YOLOv4, SSD, Retina-Net, Faster RCNN.
- Add SE module to your network
- Recommended framework : pytorch

Squeeze-and-Excitation Networks



$F_{tr}()$: convolution operation

$F_{sq}()$: avg_pool2d

$F_{ex}()$: Linear \rightarrow ReLU \rightarrow Linear \rightarrow Sigmoid

Analysis

1. The difference between adding to shallow and deep layers
2. The different amount of the SE modules
3. Loss training curve, etc.

Sample code

- Conv2d \rightarrow SELayer \rightarrow Conv2d

```
from torch import nn

class SELayer(nn.Module):
    def __init__(self, channel, reduction=16):
        super(SELayer, self).__init__()
        self.avg_pool = nn.AdaptiveAvgPool2d(1)
        self.fc = nn.Sequential(
            nn.Linear(channel, channel // reduction),
            nn.ReLU(inplace=True),
            nn.Linear(channel // reduction, channel),
            nn.Sigmoid()
        )

    def forward(self, x):
        b, c, _, _ = x.size()
        y = self.avg_pool(x).view(b, c)
        y = self.fc(y).view(b, c, 1, 1)
        return x * y
```

Dataset

- GTA video dataset
- You only need to detect car.

(Only one class)

- 2039 training images, labels
- 240 validation images, labels
- 720 testing images

- [Dataset link](#)




Labels

./HW2_ObjectDetection_2021/labels/

Each row is **[class x_center y_center width height](0~1 range)** format.(use 0 to represent car)

e.g. 0_000014 labels => image with bounding boxes

```
Open ▾  0_000014.txt  
~/Desktop/HW2_ObjectDetection_2020/labels  
0 0.1739583333333333 0.4569444444444443 0.1135416666666667 0.067592592  
0 0.21484375 0.40324074074074073 0.09010416666666667 0.05092592592592592  
0 0.7651041666666667 0.3930555555555555 0.0885416666666667 0.0787037037  
0 0.88984375 0.3921296296296296 0.0921875 0.07314814814814814  
0 0.07578125 0.3555555555555557 0.06927083333333334 0.053703703703703705  
0 0.03854166666666667 0.45231481481481484 0.07604166666666666 0.060185185  
0 0.44114583333333335 0.24675925925925926 0.01666666666666666 0.03240740  
0 0.6049479166666667 0.6824074074074075 0.0828125 0.17777777777777778
```



class	x_center	y_center	width	height
-------	----------	----------	-------	--------

Evaluation Metrics: mAP (mean Average Precision)

- Most common metric for object detection.
- In this HW, mAP defined in the **PASCAL VOC 2012** competition is used.
- We will use the following github repo to calculate your score.

<https://github.com/Cartucho/mAP>

- It also contains some explanations about how to calculate it.

Grading Policy

Model implementation - **70 points**

- implement on your own or clone from Github then run on our dataset and pass the baseline($mAP=0.4$) **on the validation dataset.** - **50/70 points**
- add the **SE** module to your model. - **20/70 points**

Model performance - **15 points**

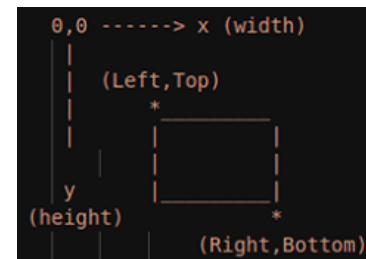
- the points will determined by the rank with your classmates.
- ranking the mAP **on testing dataset.**
- you will get **15/10/5 points** base on your rank in the class.
- You can add what you want to improve your performance.

Grading Policy

Report - **15 points**

- Experiment Setup (details of your model)
 - Method you try (Data pre-process, Model architecture, Hyperparameters,...)
- Brief explain your code(contain SE module part)
 - if you used code from GitHub, provide reference.
- Your validation results on your two model
 - with/without your modification(ex: with/without SE module)
- Discussion
 - Problems you encountered
 - which layer you add SE modules to and compare the corresponding results
 - What else you add to your model

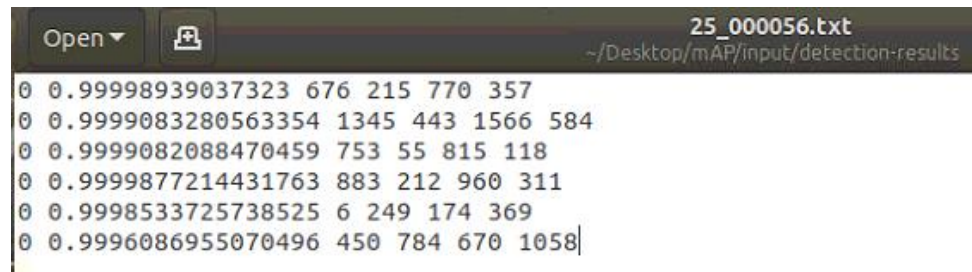
Hand in Rules(1/3)



You should hand in your result by detecting the **testing data** through your model.

format **[class confidence left top right bottom]** in pixel wise(1920x1080).

e.g. test data 25_000094 raw image -> detection result



Hand in Rules(2/3)

Store each detection result in `[image_name].txt`

Right now, you should have two models: with and without your modification.

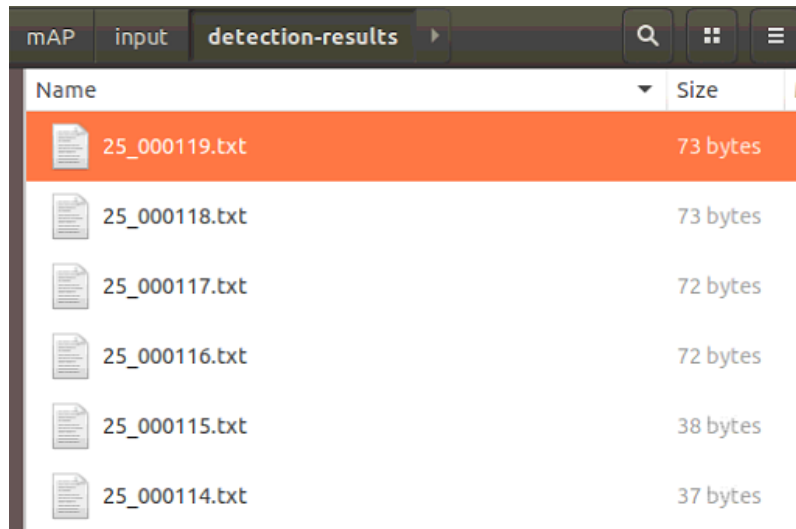
Use these two models to detect on the testing dataset.







Submit two results in different folders.

You should hand in

Two result folder contain testing 720 results

Two folder should be named: Original and Modified



mAP input detection-results			Q	⋮	≡
Name		Size			
 25_000119.txt		73 bytes			
 25_000118.txt		73 bytes			
 25_000117.txt		72 bytes			
 25_000116.txt		72 bytes			
 25_000115.txt		38 bytes			
 25_000114.txt		37 bytes			

Hand in Rules(3/3)

Your submission should contain:

- **Two result folder contain testing 720 results (Original and Modified)**
- **Report (in pdf)**
- **Code (Do not contain checkpoint and dataset)**

Compress them into **one zip** file name **HW2_[studentid].zip**.

Penalty

Format penalty - **10 points**

- Submit the result in the wrong name, format, etc.
- Submit the report not in pdf format

Late penalty - **20% per day**

- 1 day => 80%, 2 day => 60%...

You can use any code from Github, but don't copy from your classmate!

References

<https://github.com/Cartucho/mAP>

<https://arxiv.org/pdf/1709.01507.pdf>