Video Streaming and Tracking

Homework 2 - Object Detection

Deadline: 2022/10/31 23:55

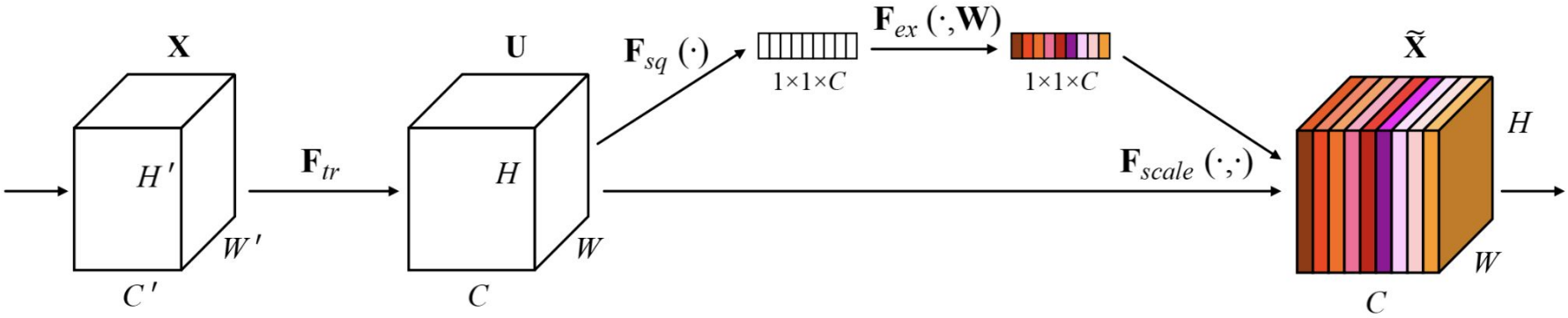
Outline

- 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
  - YOLOX (we use the official code to set the baseline)
- Add SE module to your network
- Framework : PyTorch

Squeeze-and-Excitation Networks



$F_{tr}()$: convolution operation

$F_{sq}()$: avg_pool2d

$F_{ex}()$: Linear → ReLU → Linear → Sigmoid

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

Sample code

- Conv2d → SELayer → Conv2d

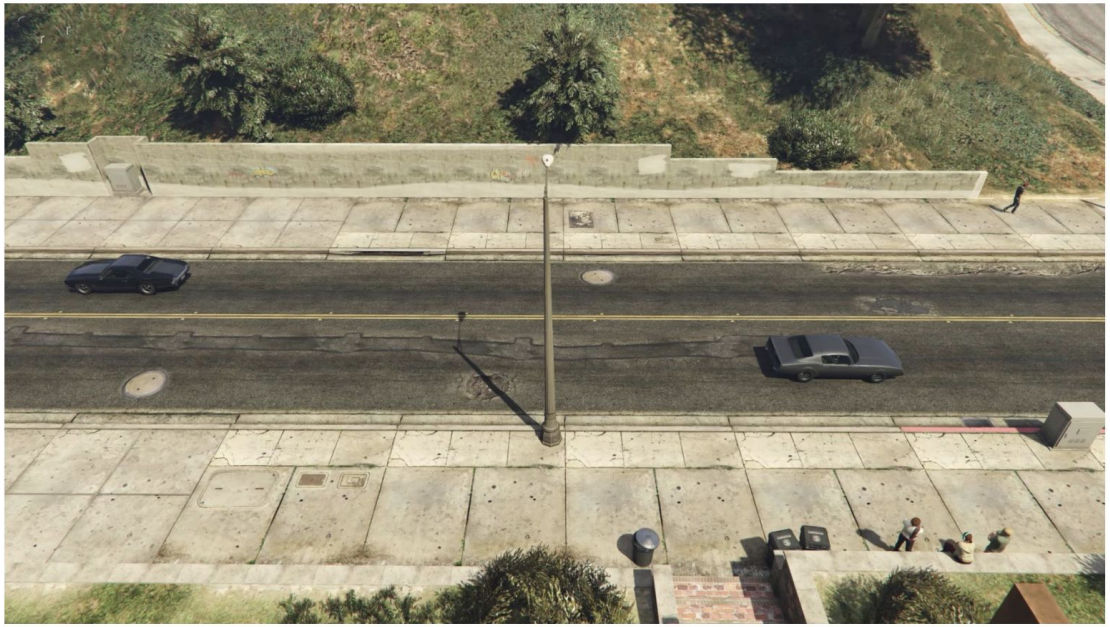
```
from torch import nn

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)
- 1596 training images, labels
- 227 validation images, labels
- 456 testing images



Dataset link

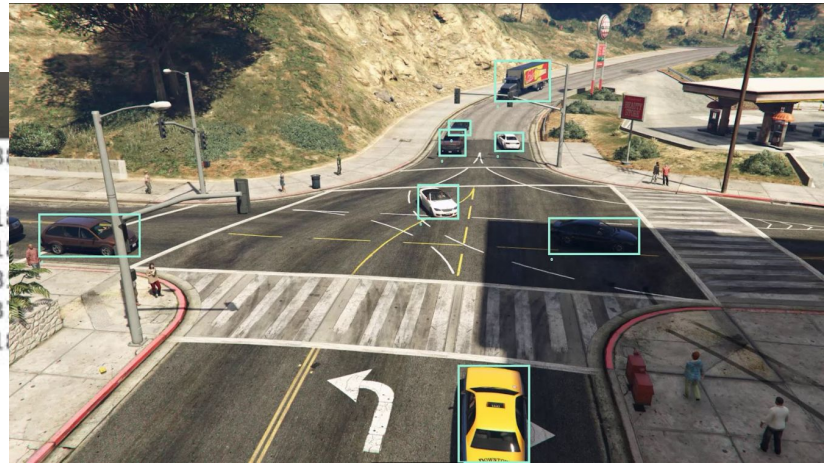
Labels

- ./HW2_ObjectDetection_2022/{train, val}_labels/
- Each row is [class x_center y_center width height] (0~1 range) format (use 0 to represent car)

Open ▾ 8.txt
~/Desktop/HW2_ObjectDetection_2022/train_labels

```
0 0.59453125 0.8930555555555556 0.08802083333333334 0.213888888888
0 0.7171875 0.5055555555555555 0.1125 0.08148148148148149
0 0.5263020833333333 0.4310185185185185 0.0515625 0.076851851851851
0 0.54375 0.3023148148148148 0.03333333333333333 0.060185185185185
0 0.55234375 0.27037037037037037 0.028645833333333332 0.035185185185
0 0.09869791666666666 0.5041666666666667 0.12447916666666667 0.0953
0 0.6130208333333333 0.300462962962963 0.036458333333333336 0.04351
0 0.6296875 0.16805555555555557 0.06875 0.09166666666666666
```

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/rafaelpadilla/Object-Detection-Metrics

- It also contains some explanations about how to calculate it
- We will set IoU threshold greater than 0.85 to calculate the testing score

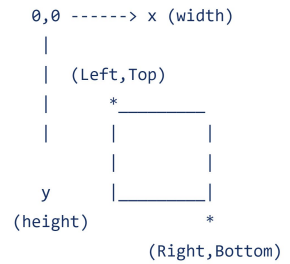
  E.g. python pascalvoc.py -t 0.85 -gtformat xyrb -detformat xyrb -np

Grading Policy (1/2)

- Model implementation - 65 points
  - Implement on your own or clone from Github then run on our dataset and **pass the baseline** (mAP = **0.8**) by **using the code we provide (set IoU threshold to 0.85) to evaluate on the validation set** - 50/65 points
  - Add the **SE** module to your model - 15/65 points
- Model performance - 15 points
  - The points will determined by the rank with your classmates
  - Ranking the **averange** mAP (**with/without SE module) on testing set** - you will get **15 / 10 / 5 / 0 points** base on your rank in the class
  - You can use **SE module + other module** to improve your performance

Grading Policy (2/2)

- Report - 20 points (15 basic + 5 bouns points)
  - Experiment Setup
    - Data pre-process, Model architecture, Hyperparameters,...
  - Brief explain your code
    - **include SE module or other modules and training / inference command line**
    - If you used code from GitHub, provide reference
  - Screenshot your **validation results** on your two models (with / without SE module)
  - Discussion
    - Problems you encountered
    - Which layer you add SE modules to and compare the corresponding results
  - (Optional) Analysis - 5 bouns points
    - The difference between adding to shallow and deep layers
    - The different amount of the SE modules in one layer
    - Loss training curve, etc.



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)

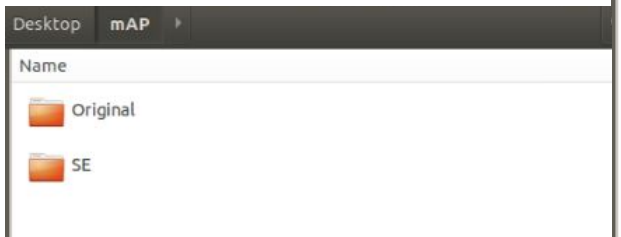
Open

14.txt
~/Desktop/mAP/result

```
0 0.222222 676 215 770 357
0 0.111111 1345 443 1556 584
0 0.999999 753 55 815 118
```

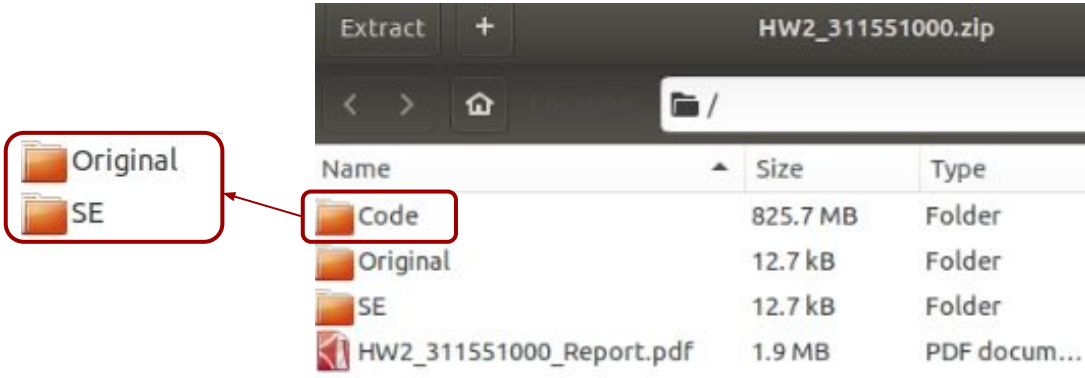
Hand in Rules (2/3)

- Store each detection result in **[image_name].txt**
- Right now, you should have two models: with and without the SE module
- Use these two models to detect on the testing set
- Submit two results in different folders
- You should hand in
  - **Two result folder contain testing 456 results**
  - **Two folder should be named: Original and SE**



Hand in Rules (3/3)

- Your submission should contain
  - **Two result folder contain testing 456 results (with / without SE module)**
  - **Report (in pdf)**
  - **Code (include your environment and two checkpoints). Do not contain dataset.**
    - Please submit the code that can generate the prediction results in the **two folders**.
- 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
- No validation results are shown in the report - 10 points
- 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://arxiv.org/pdf/1709.01507.pdf
- https://github.com/Megvii-BaseDetection/YOLOX
- https://github.com/rafaelpadilla/Object-Detection-Metrics