Video Streaming and Tracking

Homework 2 - Object Detection

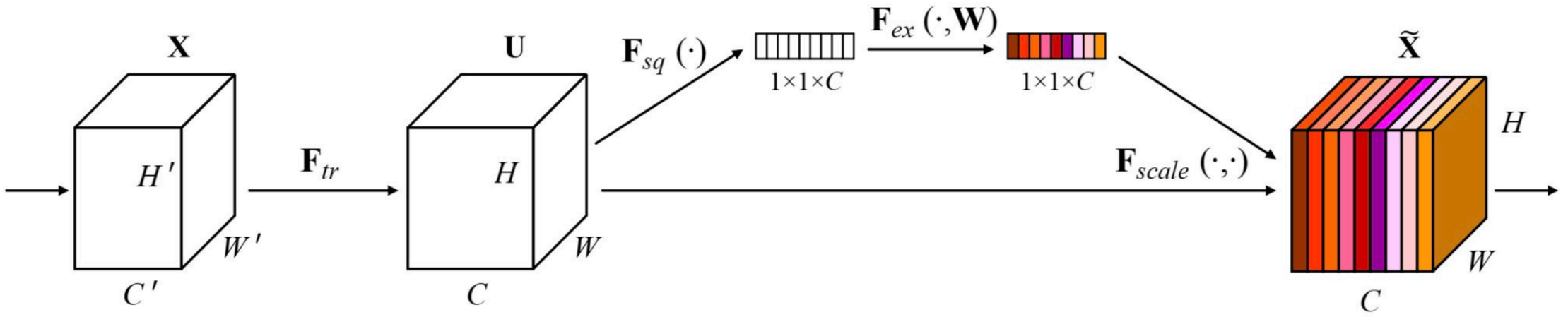
Outline

- 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



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

$F_{tr}()$: Convolution operation

$F_{sq}()$: avg_pool2d

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

Sample Code

- Conv2d → SELayer → Conv2d

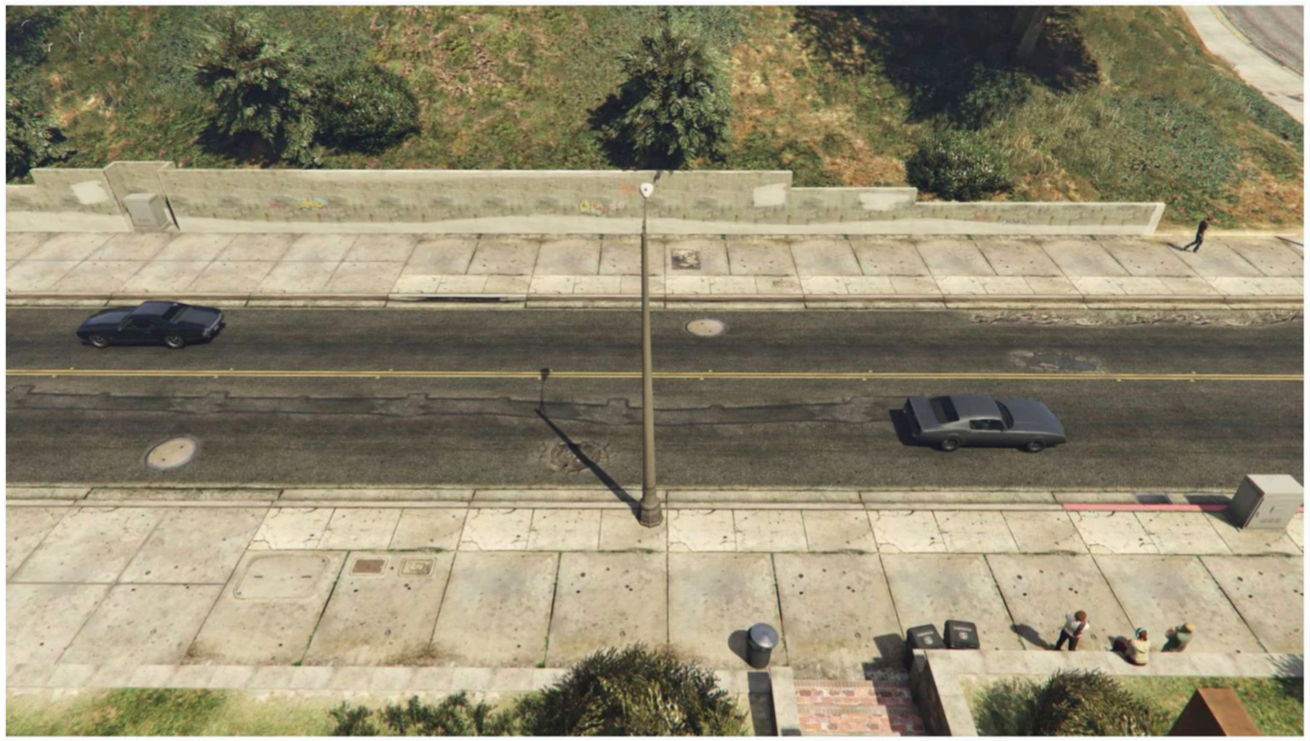
```python
from torch import nn

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



Labels

- ./HW2_ObjectDetection_2023/{train, val}_labels/
- Each row is [class x_center y_center width height] (0~1 range) format (use 0 to represent car)
- Converting the dataset to the YOLOX-compatible format(e.g. COCO)

Open ▾ 0_000014.txt
~/Desktop/HW2_ObjectDetection_2020/labels

```
0 0.17395833333333333 0.45694444444444443 0.11354166666666667 0.067592592
0 0.21484375 0.40324074074074073 0.09010416666666667 0.05092592592592592
0 0.7651041666666667 0.39305555555555555 0.08854166666666667 0.0787037037
0 0.88984375 0.3921296296296296 0.0921875 0.07314814814814814
0 0.07578125 0.35555555555555557 0.06927083333333334 0.05370370370370705
0 0.03854166666666667 0.45231481481481484 0.07604166666666666 0.060185185
0 0.44114583333333335 0.24675925925925926 0.016666666666666666 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/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

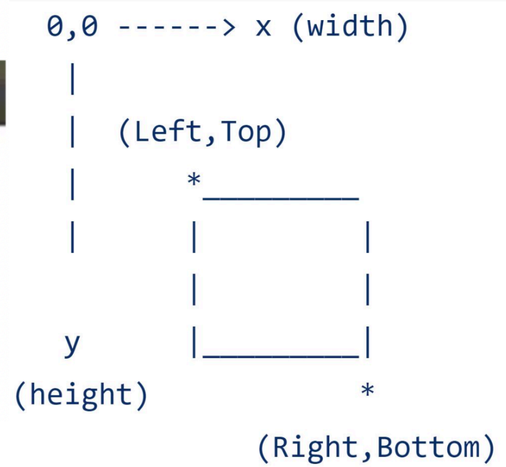
  E.g. python pascalvoc.py -t 0.85 -gtformat xyrb -detformat xyrb -np

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 ▾ 25_000056.txt
~/Desktop/mAP/input/detection-results

```
0 0.99998939037323 676 215 770 357
0 0.9999083280563354 1345 443 1566 584
0 0.9999082088470459 753 55 815 118
0 0.9999877214431763 883 212 960 311
0 0.9998533725738525 6 249 174 369
0 0.9996086955070496 450 784 670 1058
```

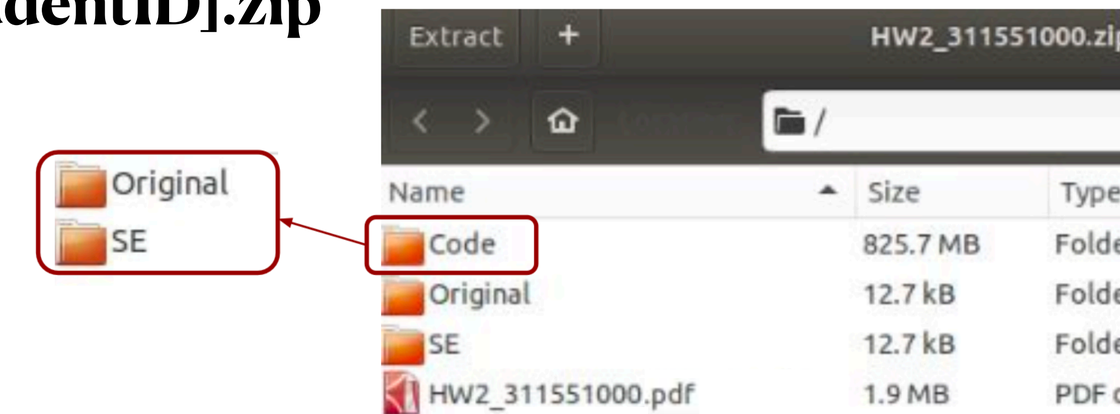


Hand in Rules (2/3)

- Store each detection result in **[image_name].txt**
- 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 720 results
  - Two folder should be named: Original and SE

Hand in Rules (3/3)

- Your submission should contain
  - Two result folder contain testing 720 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**



Grading (1/2)

- Model implementation - 70 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 points
  - Add the **SE** module to your model - 20 points
- Model performance - 15 points
  - The points will determined by the rank with your classmates
  - Ranking the average 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 (2/2)

- Report - 15 points
  - Experiment Setup (Data pre-process, Hyperparameters,...) - 5 points
  - Explain which layer you add SE modules to and compare the corresponding results - 5 points
  - Screenshot your **validation results** on your two models (with / without SE module) - 5 points
- If you used code from GitHub, provide reference

Penalty

- Format penalty - 5 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
- Deadline: 2023/11/8 23:59
- 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