Master's degree in
Computer Engineering for Robotics and Smart Industry

UNIVERSITÀ di VERONA
Dipartimento di INFORMATICA

Deep Learning Project

ACTIVE VISION RECOGNITION:

The use of Faster R-CNN for object detection on a custom dataset

Prof. Marco Cristani, Francesco Giuliari

Fabio Castellini

Academic Year 2020/2021

Objectives

- Understand the nature of the Active Vision Dataset
- Create a script to adapt AVD's format to the standard MSCOCO format
- Train a Faster-RCNN *(Region Based Convolutional Neural Network)* on the custom dataset
- Validate after each training epoch to check if any progress is achieved
- Final test on the three subsets of the testing scenes (ordered by difficulty)
- Qualitative results of object detection (classification + bounding box regression) on some images

The model

- After some experimentations the popular Faster-RCNN architecture has been chosen
- The training phase started from a pretrained model on *COCO train2017* and 2 (out of 5) trainable backbone layers, available inside the torchvision library
- Learning rate was set at 0,001; additional warmup (at the beginning) and decay (towards the 20th epoch) were performed
- Batch size, using only one GPU, was set at 8 images

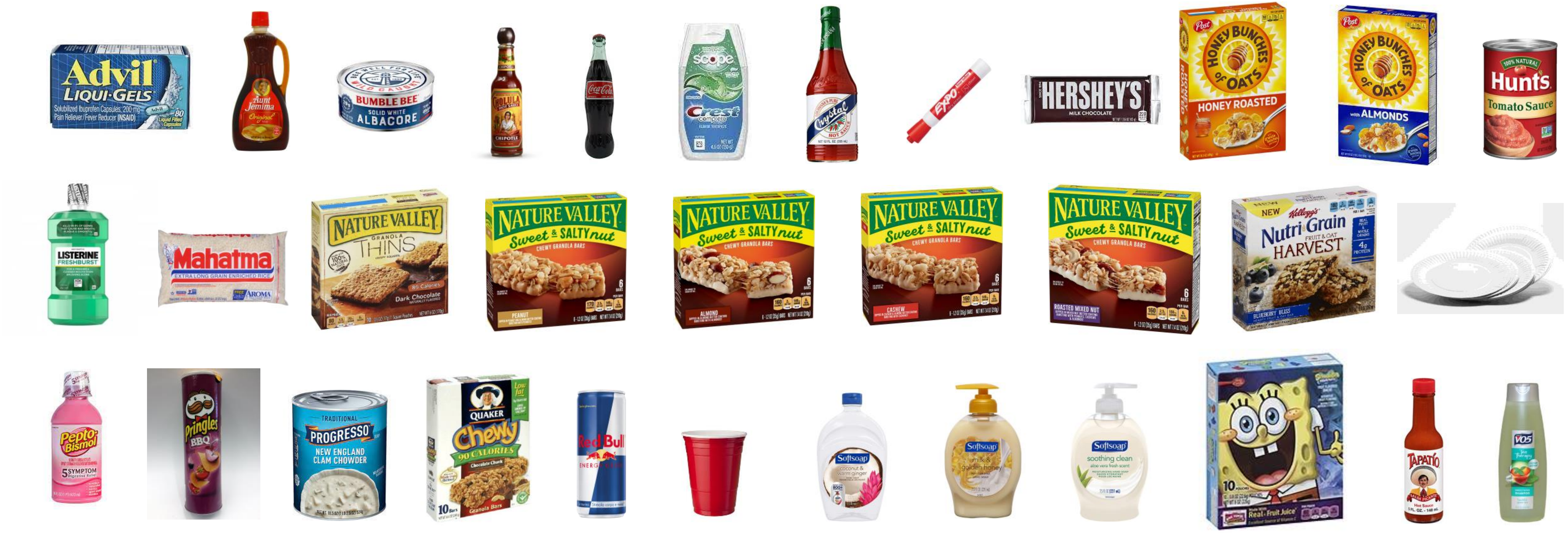
The project's structure

```
DL_Project/
│
├─ projectFASTER-RCNN_main.ipynb
│
├─ objectDetectionCode/
│  ├─ AVD_to_COCO.py
│  ├─ coco_eval.py
│  ├─ coco_utils.py
│  ├─ detect.py
│  ├─ engine.py
│  ├─ evaluatePerDifficulty.py
│  ├─ group_by_aspect_ratio.py
│  ├─ presets.py
│  ├─ README.md
│  ├─ train.py
│  ├─ transforms_AVD.py
│  ├─ transforms.py
│  ├─ utils.py│
│
├─ annotations/
│  ├─ instances_easy_valAVD.json
│  ├─ instances_medium_valAVD.json
│  ├─ instances_hard_valAVD.json
│  ├─ instances_trainAVD.json
│  ├─ instancesMapping.txt
│
├─ annotations_AVD_structure/
│  ├─ Home_001_1/
│  │  ├─ annotations.json
│  ├─ Home_001_2/
│  │  ├─ annotations.json
│  ├─ .../
│
├─ checkpoints/
│  ├─ model_24.pth
│
│
├─ trainAVD/
│  ├─ 000110000010101.jpg
│  ├─ 000110000020101.jpg
│  ├─ ...
│
├─ valAVD/
│  ├─ easy/
│  │  ├─ 000520000010101.jpg
│  │  ├─ ...
│  ├─ medium/
│  │  ├─ 000120000010101.jpg
│  │  ├─ ...
│  ├─ hard/
│  │  ├─ 000320000010101.jpg
│  │  ├─ ...
```

AVD: Active Vision Dataset

- The Active Vision Dataset aims to allow simulation of *robotic motion* through an environment for *object detection*. Images were collected from **many scenes**, which may be one or more rooms in a **home** or **office**.
- *Collection Procedure (from the AVD website):* for most scenes we conduct a full scan, and then move instances around the scene and collect either a full second scan or a partial second scan. A partial scan has much less images and coverage of the scene than a full scan. A number of object instances from the BigBIRD dataset are placed in every scene. The second scan usually has a different subset of BigBIRD instances than the first scan. We then sample 12 images 30 degrees apart at various locations in the scene

AVD: Active Vision Dataset



The 33 instances inside AVD

AVD: how it was splitted

- **Training scenes (16188 images):**

Home_001_1, Home_002_1, Home_003_1, Home_004_1, Home_005_1, Home_006_1, Home_007_1, Home_008_1, Home_010_1, Home_011_1, Home_014_1, Office_001_1

- **Validation/testing scenes (9708 images):**

Easy (1728): Home005_2, Home015_1

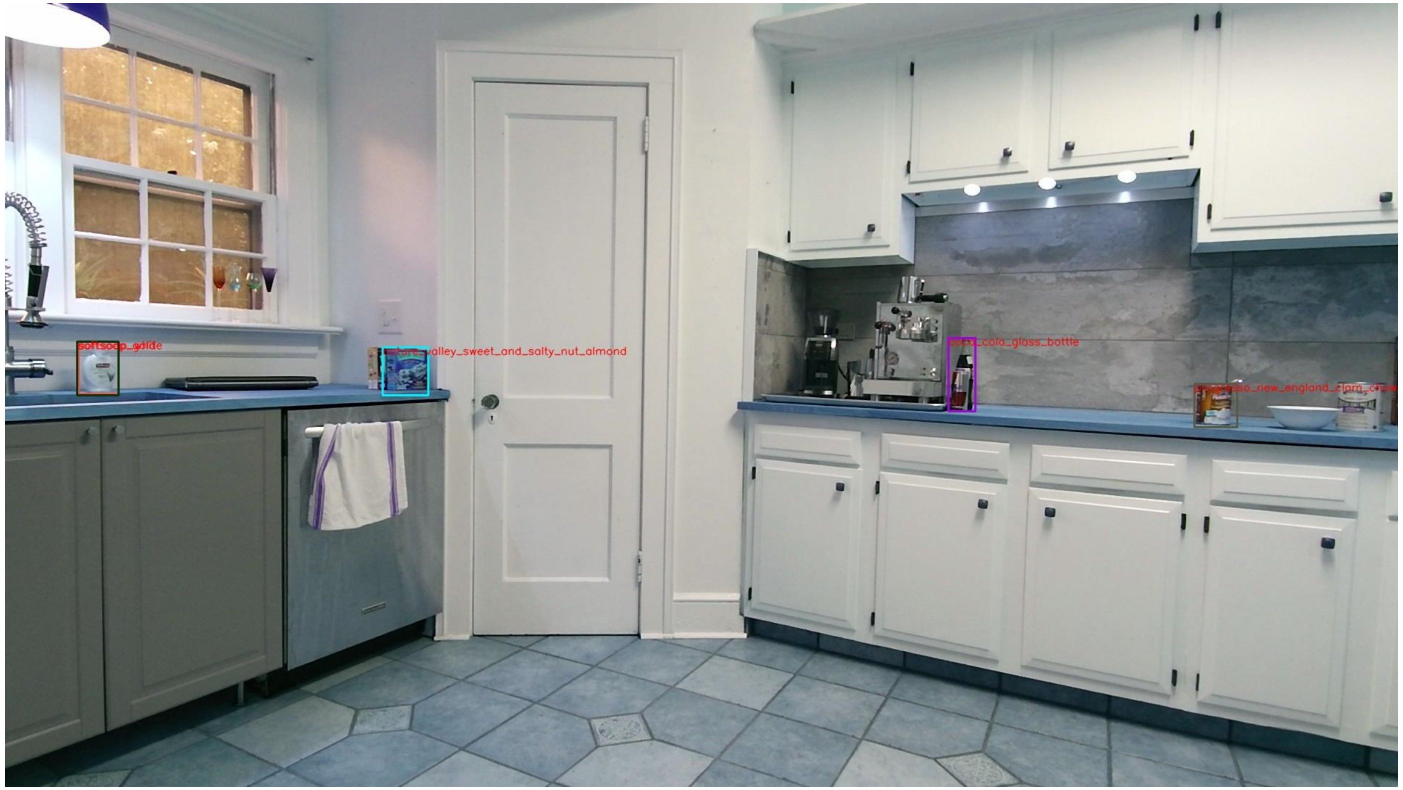
Medium (3276): Home001_2, Home016_1, Home014_2

Hard (4704): Home003_2, Home004_2, Home013_1

COCO evaluation metrics on the medium difficulty testest

```
IoU metric: bbox
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.393
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.614
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.463
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 ] = 0.278
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.509
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.512
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.523
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.523
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 ] = 0.417
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.643
```

Qualitative result on the easy testset

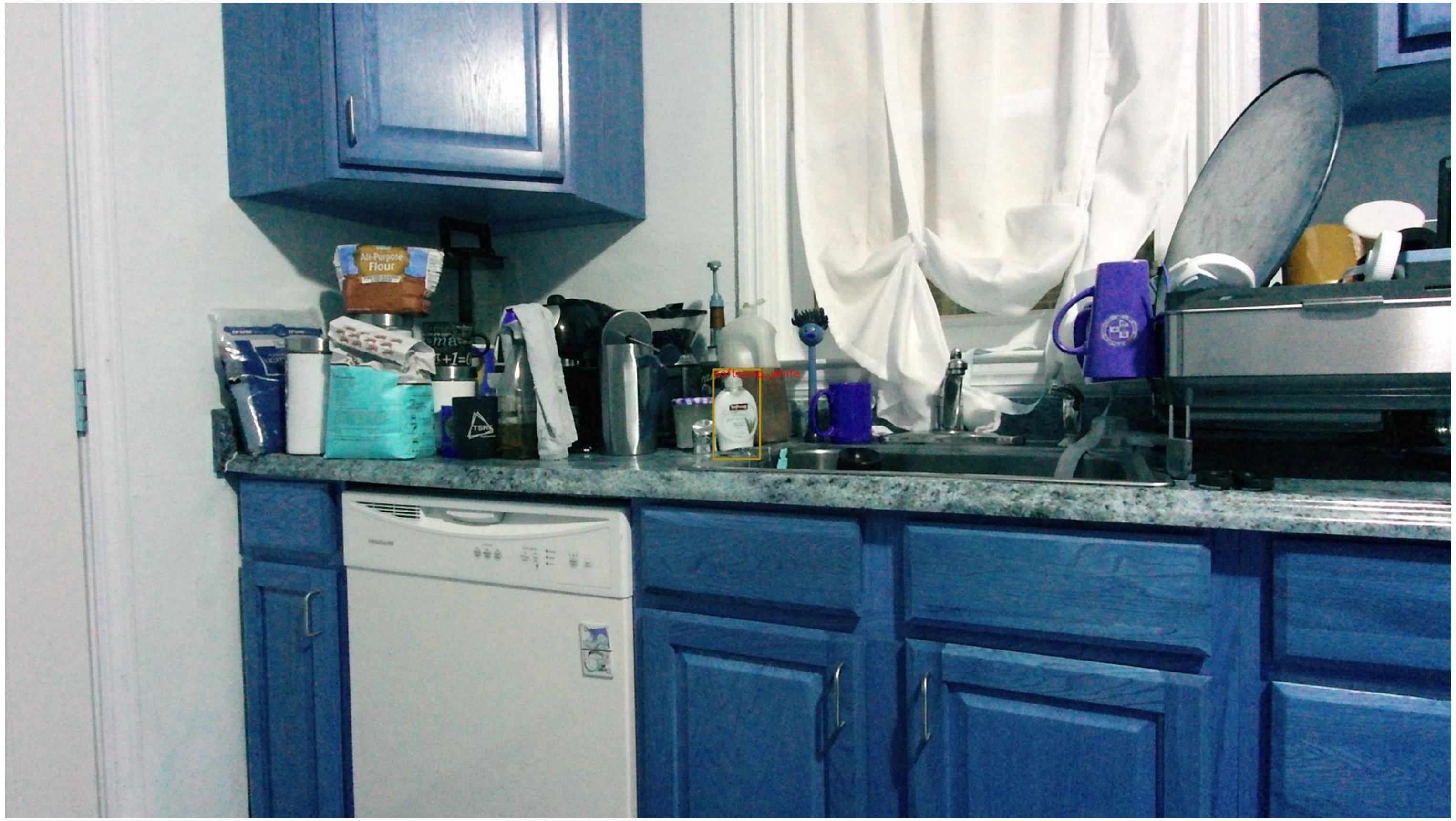


Selected image: 000520003850101.jpg

Scores of the detected objects:

Object: progresso_new_england_clam_chowder with score: 0.99480283
Object: coca_cola_glass_bottle with score: 0.992034
Object: nature_valley_sweet_and_salty_nut_almond with score: 0.7632372
Object: softsoap_white with score: 0.6416099
Object : softsoap_gold with score: 0.63568956
Object: nature_valley_sweet_and_salty_nut_cashew with score: 0.257843
Object: nature_valley_sweet_and_salty_nut_roasted_mix_nut with score: 0.2547
Object : nutrigrain_harvest_blueberry_bliss with score: 0.23512077
Object : honey_bunches_of_oats_with_almonds with score: 0.1307449
Object : nutrigrain_harvest_blueberry_bliss with score: 0.121005155
Object : quaker_chewy_low_fat_chocolate_chunk with score: 0.09978826
Object : spongebob_squarepants_fruit_snaks with score: 0.09103382
Object : bumblebee_albacore with score: 0.059846364
Object : spongebob_squarepants_fruit_snaks with score: 0.05562656

Qualitative result on the hard testset



Selected image: 000420001720101.jpg

Scores of the detected objects:

Object : softsoap_white with score: 0.9773382
Object: softsoap_gold with score: 0.050204426

Future development

- The main drawback in the final results was the mean Average Precision, that, according to COCO standard metrics is around 35%, depending on the testset. As suggested by the teachers, *Data Augmentation* could be performed on top of the trained model.
- The albumentations Python library has been explored but code implementation did not lead to functional results yet.
- Just to mention, some basic cropping and flipping were performed during training.
- The whole network could be retrained using different hyperparameters and settings. Even though the loss (composed by many factors in object detection) kept dicreasing over time, model accuracy (tested every epoch on the easy subset) seemed to converge pretty early.

UNIVERSITÀ di VERONA
Dipartimento di INFORMATICA

Resources

- https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html
- *https://arxiv.org/pdf/1506.01497.pdf*
- *https://pytorch.org/vision/stable/models.html*
- *https://www.cs.unc.edu/~ammirato/active_vision_dataset_website/index.html*
- *https://github.com/ammirato/AVDB_evaluation*
- *https://github.com/ammirato/active_vision_dataset_processing*