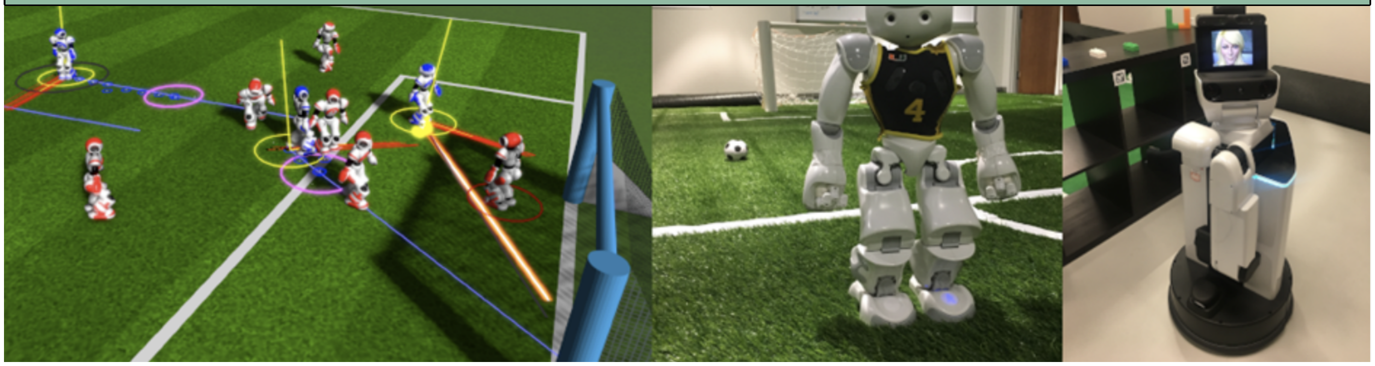
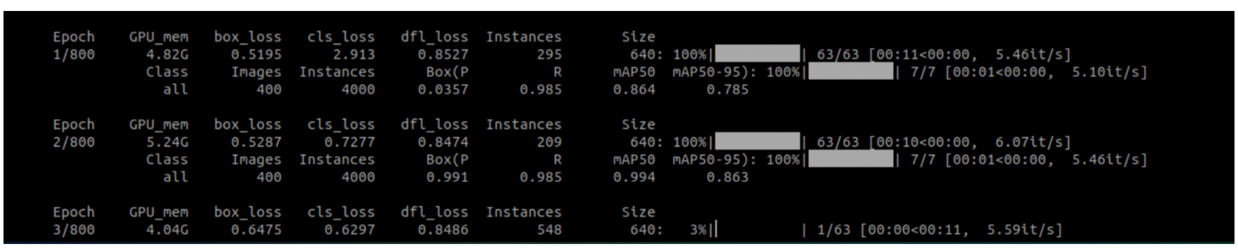
Fall 2024 - CSC398 Autonomous Robots - Assignment 7



Due date: 11/21/2024, 11:59pm. This assignment is worth 20 points.

This assignment is worth 20 points and it is due on or before November 21st, 2024 before midnight. The goal of this exercise is familiarize yourself with a typical computer vision pipeline. You will generate and augment a dataset for a state-of-the-art deep learning architecture. You learn a vision model and use it for our HSR.

1. We made some small modifications to the simulation world. `git pull` (from your catkin workspace) to update.
2. cd into the `src/` directory off your catkin workspace. Clone the new assignment: `git clone` https://classroom.github.com/a/CFkm745J.You will find a starter code there.
3. cd into your assignment-7 repo.
4. Open https://www.cs.miami.edu/home/visser/csc398-files/ in your browser and copy the zip-file `generated_crop_data.zip` into your assignment-7 repo. Unzip it. Remove the zip-file.
5. Once the directory and files are copied, go to the `scripts` directory and run the `init_package.py` script. This will adjust your package naming. `catkin_make` the workspace and source.
6. You need to install a few more packages: `pip install -r requirements.txt`.
7. Run `datagen.py`. When prompted answer 'Y' to the questions. This will take a few minutes. The script creates 5,610 files in the `datasets`-folder by default. Cross-check the created dataset.
8. Once the dataset for the training process is generated, run the `train.py` script. The terminal should look like in figure 1. The training process will take just under 2 hrs. You can change the parameters in `train.py` to speed-up the process. The heigher the batch size the faster the training. The default is 48 but 64 would be better. The memory of your desktop is the limit. Also, we have 300 epochs as a default. However, this might not be enough.



```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      1/800      4.82G     0.5195      2.913     0.8527        295        640: 100%|██████████| 63/63 [00:11<00:00,  5.46it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 7/7 [00:01<00:00,  5.10it/s]
                   all        400       4000     0.0357      0.985      0.864      0.785

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      2/800      5.24G     0.5287     0.7277     0.8474        209        640: 100%|██████████| 63/63 [00:10<00:00,  6.07it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 7/7 [00:01<00:00,  5.46it/s]
                   all        400       4000      0.991      0.985      0.994      0.863

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      3/800      4.04G     0.6475     0.6297     0.8486        548        640:   3%||          | 1/63 [00:00<00:11,  5.59it/s]
```

Figure 1: Typical output of the training process.

9. After training is complete, review the results. You can find a confusion matrix and AUC figures under the newly created `runs/detect` directory. There will be one folder for each started training run. Make a note of the accuracy of the trained data on the test data.
10. To test the data on the robot, make use your assignment 6.
    - Edit the `vision.launch` file in your package and **adjust the path** in the line with the `yolo_weights` info to your latest run. Also: **add a node** that runs the `pc_bb_provider.py` script that will create the 3D convex hulls of the perceived objects in RViz. Make sure you add a MarkerArray to your RViz with the topic `yolo_detect/boxes`.
    - Run the simulation with `./isaaac_sim_hsr_start.sh`.
    - Use your code from assignment 6 to navigate to position (1.5,2.3) with orientation 0°.
    - Once the HSR reaches its destination, run `hsr_simple_mover.py` to ensure proper orientation of the robot, i.e., the robot should look at the spawned objects.

- Launch the `vision.launch` file from your assignment-7 repo. Your see a popup window with the camera view and the detected objects.
- Measure the accuracy of the detected objects and compare this accuracy with the training results. Elaborate on your findings. Hint: you can make a table that shows the comparison. A good start for the measurement is the `yolo_detect` script in the `robocup2023` package under the scripts/vision folder.
- Take a screenshot showing the detection in simulation and another one in rviz. Add them to your repository.

Submission:

1. Add and commit the modifications and findings to the provided package to GitHub classroom.