Exercise 1, part 1: 2D Object Detection

1 Introduction

For this first homework, you will implement and train a deep neural network for 2D object detection. The network performance will be evaluated on a custom detection dataset created with our FreiCar simulator. As you will be exclusively detecting cars in the dataset, the only class present in the dataset is the class freicar.

A code repository is provided that you have to complete in order to solve the task. We will not ask you to implement a network architecture from scratch. We would rather like to see that you are able to integrate existing modules into your code. Thus you are free to use existing implementations available online as you see them fit.

A brief overview for setup, training, and inference is contained in the repository README.md. Please read the code comments and notes carefully to avoid unnecessary bugs.

2 Tasks

You will train and evaluate the EfficientDet [1] object detection architecture - its D-0 variant in specific, which is the smallest variant and thus provides the highest inference speed. To solve this task, we prepared a repository here (which is already cloned into your freicar_ws/src folder):

https://aisgit.informatik.uni-freiburg.de/vertensj/freicar_base/-/tree/master/freicar_exercises/01-01-object-detection-exercise

2.1 Complete codebase

You have to implement 4 code segments in order to successfully train and evaluate your model:

- Dataloader in dataloader/freicar_dataloader.py: We ask you to implement the dataloader __getitem__() function to support the indexing such that dataset[i] can be used to get i-th sample. Also, implement the __len__() function so that len(dataset) returns the size of the dataset. If you are not familiar with PyTorch dataloaders, visit https://pytorch.org/tutorials/beginner/data_loading_tutorial.html.

- Train loop in train.py: In the train loop, you have to access the training samples (image and annotations) that you obtain from the dataloader, zero all previous gradients, do the model forward-pass, calculate the loss (classification loss and regression loss), backpropagate the gradients and perform one optimizer step.

- Eval loop in train.py: Very similar code as the train loop. We need this to log the evaluation loss to detect overfitting.
• Evaluation script in `evaluate.py`: Here you will implement the calculation of the mIoU (mean Intersection over Union) metric and the AP (Average Precision) metric over the full dataset and print out the result.

2.2 Evaluation

Please write a one-page report on your findings. This report includes a short explanation of data augmentation techniques used (see Bonus) and an evaluation of the overall model performance on the validation split of the dataset. The dataset contains both images from the real world and images from the simulator. You should provide evaluation metrics for the real, synthetic and real + synthetic dataset split and briefly interpret the results. You should evaluate mean Intersection over Union (mIoU) and mean Average Precision (mAP). The mIoU value should be the mean of all single-image IoU values, where you should only analyze images that contain at least 1 ground truth bounding box. The mAP value calculation should analyze all images in the evaluation split including images with no ground truth bounding box. Please use an IoU threshold of 0.5 for your mAP calculations.

Also report the FPS of the model for doing inference with a batch size of 1. Code for FPS benchmarking can be found in `inference.py`.

2.3 Write ROS node

Write a ROS node that subscribes to the image topic and which publishes the predicted bounding boxes for each image.

2.4 Bonus

Bonus points will be awarded for successful integration of data augmentation techniques (at least 2 different types). Since data augmentation helps avoiding over-fitting the model to the training dataset, this should slightly improve the evaluation metrics. You should therefore report results with and without augmentation enabled if you want to be awarded bonus points.

3 Submission

The submission is due on Dec 10, 2020, 23:59.
Submit your report as a PDF file and include a link to your repository. Your repository should include information of how to run your code!

Good luck!

References