FREICAR 2020: SEMANTIC SEGMENTATION

FELIX HANSER, SAKSHI GOEL, DAVIDE REZZOLI, MOHAMMAD QANADILO, NAYA BASLAN

1 ABSTRACT

In this report, we describe our approach to solve the second exercise task of FreiCAR 2020. First, we train the Fast-SCNN model to perform semantic segmentation and lane regression. The input image of the model is obtained from the front rgb camera of the FreiCAR. The task of semantic segmentation is to associate each pixel in the image to a corresponding class. There are 4 defined classes with labels from 0 to 3. The model is then extended to perform lane regression. Images from both segmentation and regression are then published as ROS topics and visualized in *rviz* . The final task is to randomly sample from pixels that belong to lane centers and publish them as *MarkerArrays* .

2 TRAINING FAST-SCNN FOR SEMANTIC SEGMENTATION

We use Fast-SCNN model for semantic segmentation of images loaded into data-loader. We do the forward pass through our model to segment images into 4 classes(Lanes, Junctions, Car, Background) and calculate the loss in form of the **torch.nn.CrossEntropyLoss()**. We get the loss values of less than 1 for this training which we try to minimise with backward propagation.

3 TRAINING FAST-SCNN FOR SEMANTIC SEGMENTATION AND LANE REGRESSION

For training of the lane regression , we added an additional classifier *Classifier_lanereg* to the fast_scnn_model which predicts the value of lane_regression during forward pass. The loss for this is calculated using **torch.nn.L1Loss()**. The loss starts from approximately 17 and is seen fluctuating between values of 17 to 6 as it tries to converge for a given number of epochs.The classification loss and the regression loss have different magnitudes and are therefore given weights before being used to minimise the total loss in backward propagation. This weighing factor acts as a normalization factor, which loss is regarded more. Currently we only weight the **torch.nn.L1Loss()** Loss and therefor have one hyperparameter which is chosen manually on the basis of evaluation results.

4 Model Evaluation and Plots

The evaluation is called every 3 Epochs and it calculates the Mean intersection over union and the regression loss l1 over the validation split. Our research shows that the mIoU is very consistent and approximately we can see the values of 0.90 to 0.95 after only a few epochs of evaluation. The l1 loss also decrease till 6 epochs. The plot is represented as a line for the training split and dots for the evaluation split. We represent the evaluation split as dots instead of line as the values are available every 3 epochs.

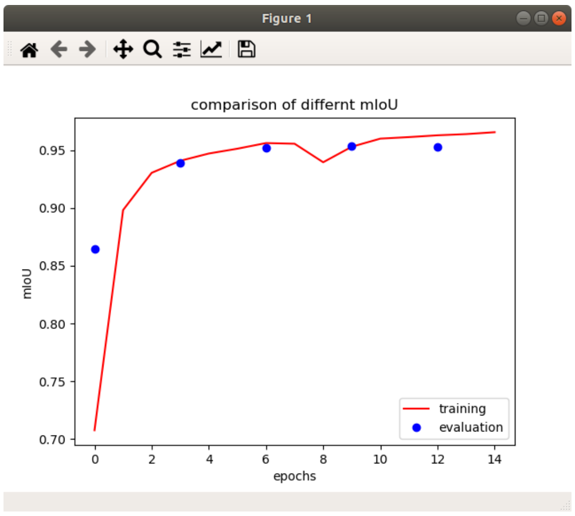


Figure 1: mIoU for 15 epochs, weighing factor 0.05

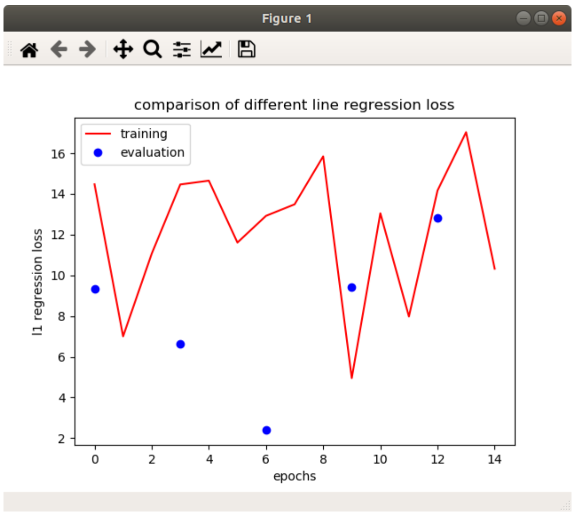
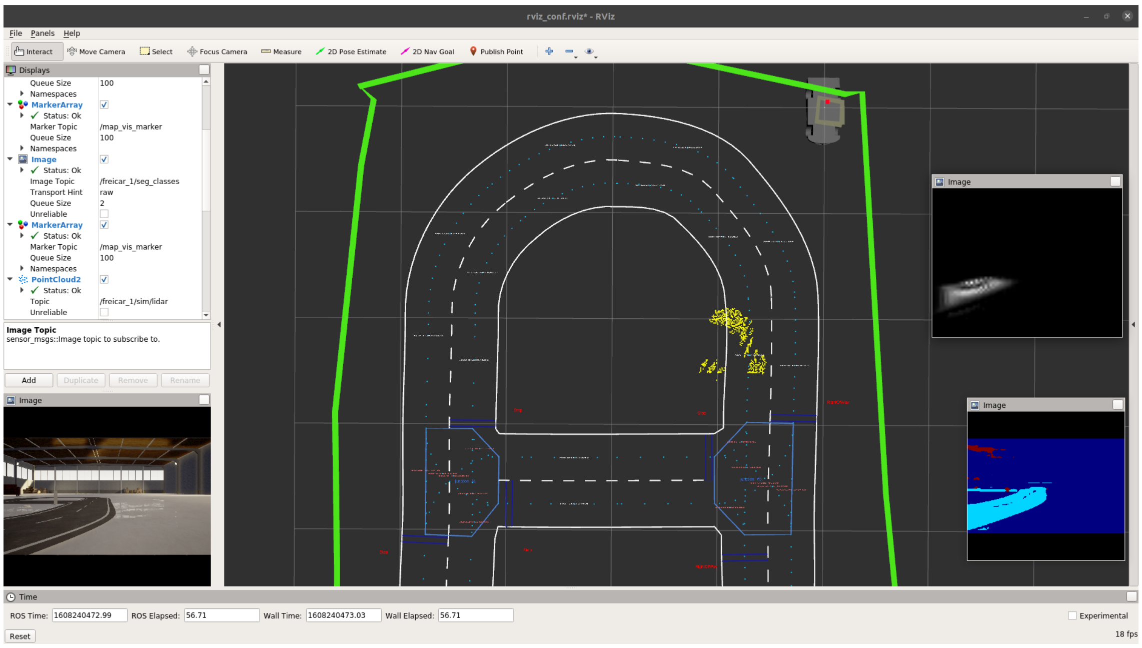


Figure 2: L1 loss for 15 epochs, weighing factor 0.05

5 ROS:Publishing Images for Semantic Segmentation and Lane Regression

In this section, we create a ROS node, called *seg_reg_publisher.py*, that subscribes to the front rgb camera image of the FreiCAR. After processing, we feed this image to the modified Fast-SCNN model and get as output tensors for segmentation and lane regression. These tensors are again processed to obtain a CV image of shape (384 x 640). These are then converted to ROS images using *cv_bridge* and published on their respective topics: */freicar_1/seg_classes* and */freicar_1/seg_lanes* .



```
Queue Size        100
Namespaces
MarkerArray       ✓
  Status: Ok
  Marker Topic    /map_vis_marker
  Queue Size      100
  Namespaces
Image             ✓
  Status: Ok
  Image Topic     /freicar_1/seg_classes
  Transport Hint  raw
  Queue Size      2
  Unreliable      ☐
MarkerArray       ✓
  Status: Ok
  Marker Topic    /map_vis_marker
  Queue Size      100
  Namespaces
PointCloud2       ✓
  Status: Ok
  Topic           /freicar_1/sim/lidar
  Unreliable      ☐

Image Topic
sensor_msgs::Image topic to subscribe to.

ROS Time: 1608240472.99  ROS Elapsed: 56.71  Wall Time: 1608240473.03  Wall Elapsed: 56.71   ☐ Experimental
                                                                                              18 fps
```

Figure 3: Screenshot of rviz, 15 Epochs, weight factor 0.05 [For better visualization is uploaded on git]

6 ROS:Applying Inverse Perspective Mapping

In this section, we expand on the ROS node described in the previous section. The image for lane regression is transformed into birds eye view using inverse perspective mapping. The class for performing this was provided in the exercise code. The output image is gray scale of shape (600 x 600) and has pixel values ranging from 0 to 255. 0 indicates black color and 255 indicates white. The regression output from the model is defined in a way such that the lane center is white (255) and this value decreases exponentially the further we are away from the center. In this ROS node (called *lane_center_publisher.py*) we first extract all pixels whose values exceed a threshold of 150, and then we randomly sample up to 700 of these points and publish them as *MarkerArrays* to be able to visualize them in *rviz*.

7 Conclusion

Training and Evaluation: We use the mIoU and L1 loss plots over training and validation split to determine which is the best combination of weighing factor, evaluation metrics(i.e mIoU and L1 loss) over given epochs. We used weighing factors 0.1,0.02,0.03,0.05 and tried running it for 10,15,21 epochs (not feasible to run all combinations due to time constraint)and found out that 0.02(over 10 epochs) and **0.05(over 15 epochs)** gives the results which show better *plots*, better *segmentation* with *visJetColorCoding()*, better *representation with rosnodes.*(The plots for other combinations are uploaded in folder 'Training_evaluation_plots') For the competition, we plan to automate this selection to get more stable training loss and better predictions for driving our FreiCar.

ros: To sum it all up, visualizing the output of the ROS topics in *rviz* helps us to better understand the model outputs of segmentation and regression. We will use these implementations in the future to autonomously drive the FreiCAR around. One idea would be to provide the lane centers as navigation goals and follow them around. This could ensure that the FreiCAR always stays in the middle. Furthermore, the semantic segmentation output will help us realize when another car or obstacle approaches, giving the autonomous car enough time to make a decision of turning away to avoid accidents. Further work on this task will include incorporating it with navigation commands.