Self-Driving Cars: Perception : Final Project Report

ROB535 Team 9: Parth Chopra, Richard Higgins, Sahib Dhanjal, Ung Hee, Chris Rockwell
December 19, 2018

I. INTRODUCTION

We were required to: *(i)* classify a given image, and *(ii)* regress the 3D-bounding box centroid of the detected object in an image. We provide an overview of our method and results. The code related to this project can be found at `github.com/relh/rob535_team9_perception`

II. TASK1

For Task 1, our goal was to classify the given images into three classes - car-like vehicles, non-car-like vehicles (motorbikes, utility vans, etc), and other objects. We experimented in PyTorch with multiple convolutional architectures such as resnets, squeeze-and-excitation resnets, PolyNet, VGGNet, and various inception networks, densenets, and nasnets. Also, to augment our dataset and improve generalization, we applied a composition of image transforms with bounded random parameters such as affine transform, color jitter, horizontal flips, and normalization.

In our initial experiments, we replaced the last layer of each of these networks with a 3-class softmax output (instead of the original 23 classes). However, we observed that our 3-class output accuracy plateaued despite different learning rates (1e-1 to 1e-4), learning rate schedulers, dropout (0 to 0.35), and optimizers. We switched back to using 23 classes, hypothesizing that this would help the network learn class specific features. To provide better feedback, we modified the loss function from a cross-entropy (CE) loss over 3-classes to a weighted sum of CE losses for both 3 and 23 class output. We trained a squeeze-and-excitation resnet from a random weight initialization with a customized number of residual blocks and layers. However, even after about 20 epochs with learning rates between 1e-2 and 1e-4, we only achieved a validation accuracy of 0.60.

Our custom network may not have learned good low-level features in initial layers, so to overcome this problem, we chose to finetune an Inception v4 network [1] pre-trained on ImageNet. This version of Inception v4 still included our modifications for the custom CE loss and modified last layer of 23-class outputs which we had developed previously. With this setup, we ultimately achieved 0.95+ validation accuracy by training using an initial *learning rate* of 1e-3, *LR scheduling, dropout* of 0.35, *weight decay* of 8e-2, and *batch size* of between 12 and 20 depending on available GPU memory. We then trained a final few epochs with different train-validation dataset splits (using different folds analogous to k-fold cross-validation) before submitting. However, this trained model achieved only 0.706 test accuracy upon submission. To further augment our dataset, we used `ImgAug`, adding random variations to our data such as salt and pepper noise and Gaussian blur. We were not able to achieve better generalization in the test set despite achieving equivalent validation accuracy.

We conducted random search over architectures and hyperparameters in an effort to improve test results. We also obtained additional data from FCAV's Driving in the Matrix project [2] and made an additional dataloader with the Element Tree package to extract image label pairs from the VOC format.

The code for this task can be found on the `master` branch of the linked repository.

III. TASK2

Localizing detected objects in 3D coordinates is crucial for self-driving applications, specifically for safety and planning. For this task, we modified the architecture of a popular pre-trained network, Inception v4 [1] to localize the vehicles in images. We replaced the last Softmax layer of the network with a 3 neuron fully connected layer to regress the centroids, given an image. Out of the several methods we explored, we chose this method because of 2 reasons: *(i)* each image just had one object in it and it would be plausible for the network to regress the object centroid well enough to pass the baseline, *(ii)* it was the easiest to implement. We provided the pair of 3D coordinates and the corresponding image for training the network. The parameters with which we obtained our best results on the task were: *batch size* of 50, *learning rate* of 1e-3 and *momentum* of 0.9. We also divided the training dataset in a $85:15$ train to validation set ratio. A RMSE of 16.41 was achieved after training the network for 1 epoch. The code for this task can be run by specifying `--task 2` in the `master` branch of our repository.

IV. CONCLUSION

In task 1, improved validation performance didn't correlate well with leaderboard result (we finished in the top half). Despite finetuning over 1000+ different models and using almost half the dataset solely for validation (and comparing folder level and subfolder level image splits), it was unclear which trained models might have the best test results. In task 2, we passed the baseline. Further improvements to the same method can be made by using semantic segmentation of point clouds deploying networks like Pointnet [3] or Complex YOLO [4]. Another feasible approach would be to use object detection networks (*i.e.*,YOLO, Mask RCNN, etc) to detect the 2D bounding box and regress that into 3D using image geometry [5].

REFERENCES

[1] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning.," in *AAAI*, vol. 4, p. 12, 2017.

[2] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, K. Rosaen, and R. Vasudevan, "Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks?," in *IEEE International Conference on Robotics and Automation*, pp. 1–8, 2017.

[3] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, vol. 1, no. 2, p. 4, 2017.

[4] M. Simon, S. Milz, K. Amende, and H.-M. Gross, "Complex-yolo: Real-time 3d object detection on point clouds," *arXiv preprint arXiv:1803.06199*, 2018.

[5] A. Mousavian, D. Anguelov, J. Flynn, and J. Košecká, "3d bounding box estimation using deep learning and geometry," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pp. 5632–5640, IEEE, 2017.