Classifying and Localizing Vehicles in Images from a Game Engine

Fall 2018 Team 3
University of Michigan
Ann Arbor, United States
wenzhe@umich.edu

***Abstract*—We presented our work on the design and implementation of a vehicle classification model using deep neural network on images taken from a game engine. There were two essential goals of the model in task 1. One was to recognize and find location of a vehicle in each snapshot, and the other was to classify the image based on the category of vehicle presented. We presented a model built upon the R-CNN object detection architecture that accomplished both goals. For task 2, we had one main goal which was to localize the vehicle in every snapshot. We presented a solution which took advantage of constant factors naturally existent in the game engine and created an algorithm similar to the nearest neighbor algorithm to localize the centroid of the vehicle.**

***Index Terms*—perception, classification, localization, R-CNN**

I. Introduction

A convolutional neural network approach to image classification can lead to good results when choosing the appropriate model parameters and architecture. However, the task became more complicated when images contained a variety of backgrounds without bounding box around the main object that could take form of many different sizes and colors. To mitigate this problem, we took a R-CNN approach which extracted candidate regions from input image and classifies the detected objects. The same problems also arose when we approached task 2. We utilized the same method in task 1 of predicting the bounding box. Since the game engine provided the ground truth about the centroids for each vehicle in the training set, we simply found the most similar box to that of what we were observing to localize the vehicle in the snapshot.

II. Methods

A. Task 1

Our approach to this task utilized pre-trained models (faster_rcnn_resnet101_kitti, ssd_mobilenet_v1_coco, faster_rcnn_inception_resnet_v2_atrous_coco) from Tensorflow Object Detection API as our baseline method. A data preprocessing step was required to convert input data(image, bounding box, class label) to the .record format. Based on the prediction results of models trained on the above 3 models, we extracted features such as box coordinates, widths, heights, area, ratio, predicted labels. These features were then fed into a LightGBM model to output a combined prediction.

B. Task 2

We first predicted the position of the bounding boxes around the vehicle. Utilizing the minimum and maximum bounding box coordinates predicted, we searched for the most similar bounding box in the training set using the nearest neighbor algorithm. The centroid of this new vehicle was assigned the same value as the centroid found in training set.

III. Results

A. Task 1

From Table. I, we can see the characteristics pertaining to each CNN. For faster_rcnn_resnet101_kitti, we find that this takes the most training steps and training time but with the least amount of accuracy. As for ssd_mobilenet_v1_coco, it takes the least amount of training time with around the same amount of training steps and accuracy as faster_rcnn_inception_resnet_v2_atrous_coco. However, faster_rcnn_inception_resnet_v2_ atrous_coco takes the longest amount of training time. In addition, by combining all these three models, we find through Fig. 1 that the top three features scoring the highest lightGBM are kitti-can1-score, cocov1-can1-score, and incep-can1-score.

B. Task 2

The results of implementing our "1-nearest neighbor" algorithm are very effective. For the performance time, the running time takes around 10 minutes for O(MN) where M is the testing size and N is the training set size. As for the actual quality of performance, we score 15.18059. Judging from these results, this is a simple method but a very effective solution.

IV. Conclusions

In task 1, we have utilized the 3 pretrained models and eventually combined them to take in many features leading us to the 3rd place in the Kaggle competition. Using the same approach to predicting bounding boxes in task 1, we have created a nearest neighbor algorithm scoring the top half of the Kaggle competitors. For more details, refer to our repository at `https://github.com/xuwenzhe/rob535-perception` on Github.

Team Members

Pengcheng Cai, Zhaoer Li, Sisi Luo, Mingshuo Shao, Xuetong Sun, Yichuan Wang, Huiwen Xu, Wenzhe Xu, Haixuan Ye, Yan Zhang

TABLE I
CHARACTERISTICS OF EACH CNN

CNN name	training steps	batch size	test accuracy	training time
faster_rcnn_resnet101_kitti	665,352	1	0.68871	34h
ssd_mobilenet_v1_coco	200,977	24	0.71607	24h
faster_rcnn_inception_resnet_v2_atrous_coco	200,000	1	0.71721	45h

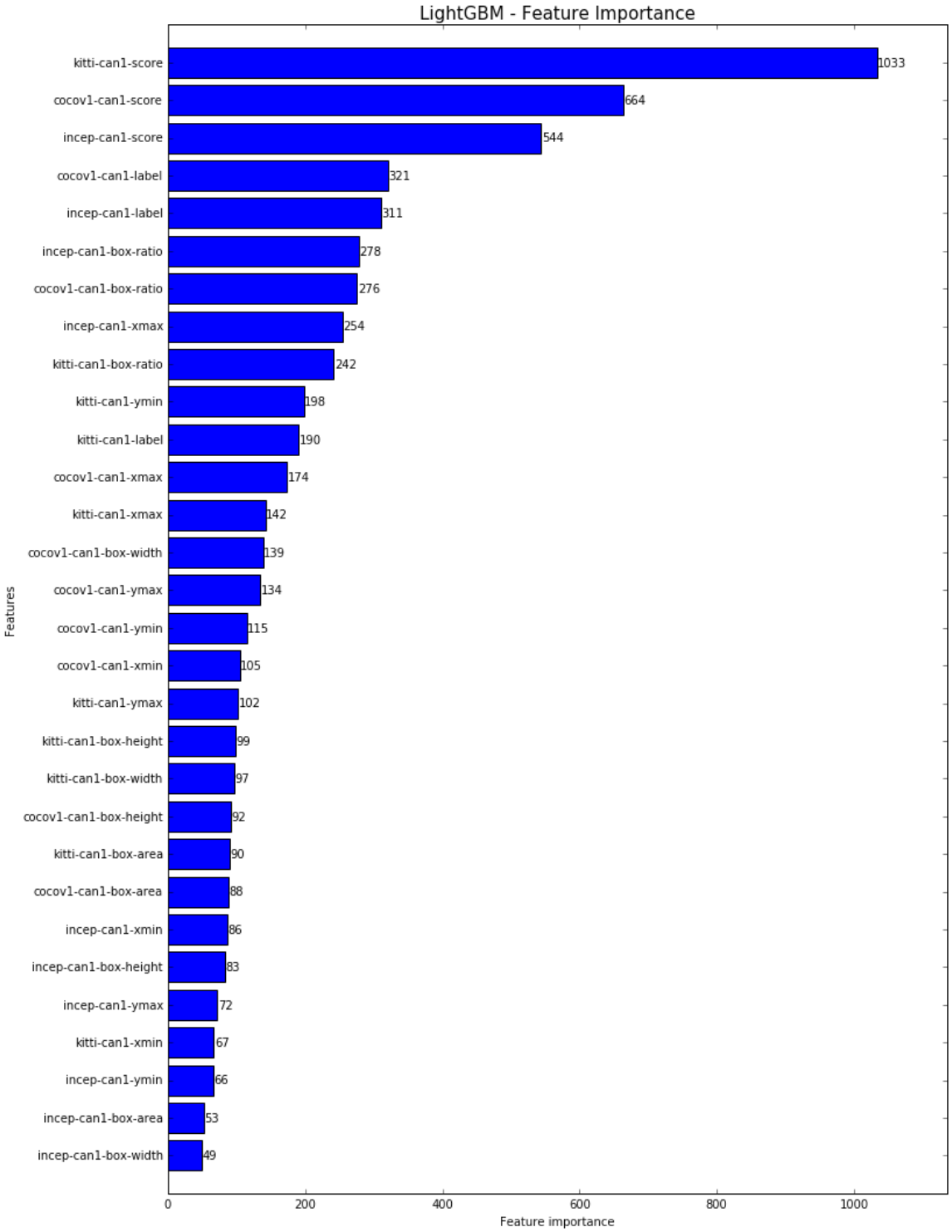

Fig. 1. Importance of features using LightGBM