CS230

HOV Car Recognition CS230-Winter 2019

Ivan C.H. Ho*
Department of Computer Science
Stanford University
ivanho1@stanford.edu

Abstract

A pipeline consisting of several neural network models was built to detect whether an image contains a vehicle likely qualified to be on the High-Occupancy Vehicle lane in California. This paper describes the scope and the detection mechanisms used, as well as providing an analysis of the results and next steps.

1 Introduction

Inspired by recent California initiatives[2][3] to evaluate / deploy commercial solutions to intelligently detect violations of cars on HOV lanes, this project attempts to identify images of cars that comply with either of the following conditions:

- Vehicle has clean air sticker/decal on car bumper area (see California Vehicle Code Section 21655.9 for details).
- Vehicle has high passenger occupancy (in most cases in California it's 2 or more) See California Vehicle Code Section 21655.5 for details.

For simplicity, non-regular 4-wheel passenger vehicles and identification of car models are excluded from the detection scheme.

Data set (train, validation, test) images are from search on internet. They are then manually labeled to enable training and verification.

Source code and trained models can be found here: https://github.com/alrightyi/hov

Presentation video can be found here: https://youtu.be/-PwwZRGBpV0

LabelBox project for image labeling: https://app.labelbox.com/projects/cjsiiydg61y3s0728b8kglkhy/overview

2 Related work

Commercially, there are already existing solutions[4] claiming that the systems are $\tilde{9}5$ percent effective[5] at identifying violating cars.

*Many thanks to CS230 TA Gael Gurvan Colas for the mentorship and suggestions on how to proceed with implementation.

[2]https://www.govtech.com/transportation/California-Deploys-Cameras-to-Catch-Carpool-Cheaters.html

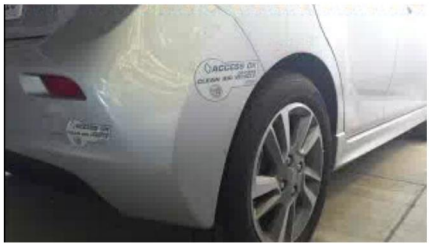
[3]https://www.sfchronicle.com/bayarea/article/The-battle-to-catch-Bay-Area-carpool-cheats-takes-12799627.php

[4]https://www.wired.com/2016/10/xerox-yeah-xerox-found-way-bust-carpool-lane-cheaters/

[5]https://www.sfchronicle.com/bayarea/article/The-battle-to-catch-Bay-Area-carpool-cheats-takes-12799627.php

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

(a) California Clean-Air Decal



(b) Car Bumper Image

Car and passenger (person) classification and detection schemes have been well studied with known training models (FC-CNN, YOLOv2) and data sets.(6)(1)(2)[6]

2.1 FC-CNN:

This type of neural network is useful for object classification. It typically contains several convolutional layers for feature identification. Examples include: (7), (8)

2.2 YOLO:

This type of NN is very effective at multi-object detection in a single image. First published in 2016(9), it has since been iterated over several versions(10) to improve the performance. It directly trains the bounding boxes and improve probabilities instead of treating object detection as a regression classifier.

Besides choosing the main neural network models, this project also uses deep learning techniques to accomplish the goal as it requires a multi-conditional identification/detection with limited data set. They include:

1. Transfer learning - ability to take pre-trained models and replace layers to detect new objects.(4)
2. Data augmentation - due to limited availability of data set, techniques such as crop, shift, rotate, brighten, channel-shift, shear, zoom are use to improve bias.[7]
3. Dropout - used to improve variance by dropping x percent of neurons at each layer to avoid over-fitting the training data.
4. Early stopping - this is used to avoid over-shooting dev set error.
5. Batch normalization - this is to speed up the learning process by grouping the input data during weight training.

3 Data set and features

The key features to be detected in an image are: car, number of passengers, car bumper, clean-air sticker on the bumper.

Known pre-trained models in FC-CNN for car (and YOLOv2 for car and passenger) identification are available. Remaining datasets required are car bumpers and clean-air stickers, as shown in 1b and 1a, respectively.

To increase the number of available training images for clean-air stickers, all versions regardless of the applicable year are used. In total 1663 images (split 586 positive and 1077 negative) were used for training and validation, randomly split 80/20. They were obtained from internet search and manually classified into "sticker" and "no sticker" buckets.

For car bumpers, images of cars and bumpers were obtained from internet and then uploaded to LabelBox.[8]. The images were then manually labelled with bounding boxes on car bumpers. Resulting

[6] http://www.image-net.org/

[7] https://keras.io/preprocessing/image/

[8] https://labelbox.com/

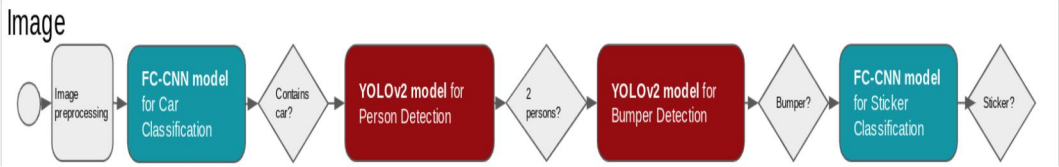


Figure 2: HOV car detection pipeline

dataset and labels were then processed to distill a list of images and pixel location of the bounding box (x-min, y-min, x-max, y-max). Total of 680 images with bounding labels were used for training and validation with a 90/10 split. (In hindsight 80/20 or lower ratio split should have been used instead to have enough validation dataset.)

4 Methods

As this is a multi-conditional detection pipeline, the detection scheme is broken into 4 major blocks to distinctively detect car, passenger, car bumper, and clean-air sticker. See figure 2.

Main reasons for using a multi-stage detection pipeline instead of one E2E are:

1. Ease of iterative implementation - given the tight time line of the project, it was lower risk to break the problem down into small subsets.
2. Known pre-trained models - car and person detection are well implemented and open-sourced. Use them as practice to tackle harder problem set.
3. Speed up training - training for object classification are typically simpler and faster than training model for object detection. In this case it was broken down into 2 sub-tasks: car bumper detection and sticker classification.
4. Increase dataset - images of car bumpers are more abundant than images of car bumpers with clean-air stickers, which would assist in increasing training accuracy.

4.1 Car classification

It uses a 3-layer Fully-Connected CNN model to train car image classification. Pre-trained model and original car training source code comes from https://github.com/antevis/CarND-Project5-Vehicle_Detection_and_Tracking. Dataset comes from Imagenet[9]. Other hyper parameters include:

1. Image size: 64x64x3
2. Batch size: 32
3. Epochs: 3
4. Optimizer: Adam
5. Loss: MSE
6. Train/Val: 80/20

To run the code, go to https://github.com/alrightyi/hov and find instructions in README.md. (demo.py under cnn_model folder)

Author claims[10] that accuracy is 99.4 percent, though during experiment the accuracy depends on the consistency of the test images.

[9] http://www.image-net.org

[10] https://github.com/antevis/CarND-Project5-Vehicle_Detection_and_Tracking

4.2 Person detection

It uses YOLOv2 model directly from Coursera class[11] with pre-trained model to implement person detection. The hyper parameters used are:

1. Image size: 608x608x3
2. Batch size: 32
3. Epochs: 30
4. Optimizer: Adam
5. Loss: Classification, Coordinates Loss, Non-Max Suppression, IOU boxes

To run the code, go to https://github.com/alrightyi/hov and find instructions in README.md. (yolo.py under yolo folder)

4.3 Bumper detection

It uses YOLOv2 model and heavily modified code from Allan Zelener[12] to retrain YOLOv2 from above to train car bumper detection. Resulting code is in yolo/retrain_yolo.py. Labelled data are downloaded and processed using yolo/get_labels.py. Hyper parameters different from above are:

1. Image size: 416x416x3
2. Train/Val: 90/10
3. Loss: Classification, Coordinates Loss, Non-Max Suppression, IOU boxes, Early Stopping

To run the code, go to https://github.com/alrightyi/hov and find instructions in README.md. (retrain_yolo.py under yolo folder)

Transfer Learning technique is used to leverage the existing weights and parameters of the YOLOv2 model found in (2) to re-train with dataset of car bumpers and clean air stickers.

4.4 Sticker classification

It uses the same 3-layer Fully-Connected CNN model from stage 1 to train clean-air sticker classification. Training and verification were done from scratch. Hyper parameters different from car classification model are:

1. Train/Val: 80/20
2. Dataset size: 1663 (1077 neg, 586 pos.)
3. Data Augment: flip, shift, rotate, shear, zoom, brightness
4. Accuracy: 67 percent

To run the code, go to https://github.com/alrightyi/hov and find instructions in README.md. (sticker_model.py)

Finally, pipeline.py connects all the building blocks together; it loads the models and weights into separate Tensorflow graphs and sessions and predicts each image from data/HOV directory.

5 Experiments/Results/Discussion

Execution of the final pipeline was done by running: "python -m pipeline" in the top directory of the git repository. It takes an optional parameter to specify a single image path to be passed into the pipeline for prediction. It also runs the prediction on ALL images under the data/HOV folder, with truth labels separated under sub-folders "HOV" and "noHOV".

[11] https://www.coursera.org/learn/convolutional-neural-networks/notebook/bbBOL/car-detection-with-yolov2

[12] https://github.com/allanzelener/YAD2K

Figure 3: Sample image results

The results (see 3) show a high-level of accuracy (98%) at rejecting images that do not qualify to be on HOV lane, while accuracy is low (39.3%) at correctly identifying HOV-qualified vehicles.

Results:

1. number of pos images: 56, accuracy: 0.39285714285714285
2. number of neg images: 50, accuracy: 0.98
3. number of total images: 106, accuracy: 0.6698113207547169

Major contributors to the low accuracy: 1) Difficulty in finding images of car exterior with passengers inside. Images with passengers are typically taken from the inside of a car or from the dashboard, which would not contain the exterior view of the car. This affects the detection logic as the CNN model for car classification is based on car exterior views. Logic was changed to ignore car classification altogether and the accuracy improved by a small margin.

2a) The number of images in training dataset for bumpers could have been higher. Only 680 images were labeled due to time-consuming manual labor to capture the binding boxes.

2b) Location of the clean air sticker relative to the bumper is also relevant. There are cases where the YOLO model detected a bumper but the bounding box did not include the sticker due to its location. Sample output images with predicted bumper bounding boxes can be found in folder yolo/output_images.

3) The number of images of cars with California Clean Air decals (stickers) are low. Only 586 images were identified from search on Google, with possible duplicates and non-conformity.

4) Consistency of the images taken could have also affected the accuracy. The images retrieved were in various angles, road conditions, environment, view-points, background conditions. If the system was built with dedicated cameras taking specific images of cars at a set view-point and focus, the results could have improved due to improved distribution of the images.

6 Contribution/Conclusion/Future Work

A 4-stage detection pipeline was implemented to working condition. Results show 98 percent accuracy for rejecting non-HOV vehicle images (40 percent accuracy for detecting HOV vehicles) with test dataset of 106 images. Vast majority of the time was spent to implement a working retrained model for car bumper detection, as well as sticker detection, with manual effort to label bounding boxes, and writing script to process the image and labeling data. Significant effort was also spent on debugging and understanding how to load multiple models and run separate predictions in the same process within the Tensorflow framework.

Besides more data analysis, I would also like to compare this implementation against an E2E trained model that detects all attributes (car, person, car bumper, and clean-air sticker), most likely using an improved YOLOv3 model. This method would require obtaining datasets of car, persons, car bumper, and clean-air stickers.

For dataset, could also explore using heat sensors to identify passengers in car instead of regular images. It offers protection of privacy, as well as ease of identification since frontal images of cars often contain glares on the windshield, making identification more difficult.

References

[1] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. *You Only Look Once: Unified, Real-Time Object Detection*. https://arxiv.org/abs/1506.02640, 2015.

[2] Joseph Redmon, Allan Zelener. *YOLO9000: Better, Faster, Stronger*. https://arxiv.org/abs/1612.08242, 2016.

[3] Allan Zelener. *YAD2K: Yet Another Darknet 2 Keras* https://github.com/allanzelener/YAD2K, 2016.

[4] Prakash Jay. *Transfer Learning using Keras* https://medium.com/@14prakash/transfer-learning-using-keras-d804b2e04ef8, 2017.

[5] Yong-Kul Ki, Doo-Kwon Baik. *Vehicle-Classification Algorithm for Single-Loop Detectors Using Neural Networks* IEEE Transactions on Vehicular Technology (Volume: 55 , Issue: 6 , Nov. 2006), 2006.

[6] Prabhu. *Understanding of Convolutional Neural Network (CNN) - Deep Learning* https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148, 2018.

[7] Lin, Tsung-Yu and RoyChowdhury, Aruni and Maji, Subhransu. *Bilinear CNN Models for Fine-Grained Visual Recognition* The IEEE International Conference on Computer Vision (ICCV) December, 2015.

[8] Zhang, Xiaopeng and Xiong, Hongkai and Zhou, Wengang and Lin, Weiyao and Tian, Qi. *Picking Deep Filter Responses for Fine-Grained Image Recognition* The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June, 2016

[9] Redmon, Joseph and Divvala, Santosh and Girshick, Ross and Farhadi, Ali. *You Only Look Once: Unified, Real-Time Object Detection* The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June, 2016.

[10] Mohammad Javad Shafiee, Brendan Chywl, Francis Li, Alexander Wong. *Fast YOLO: A Fast You Only Look Once System for Real-time Embedded Object Detection in Video* https://arxiv.org/abs/1709.05943, September, 2017.