VNU Journal of Science: Comp. Science & Com. Eng., Vol. 34, No. 1 (2018)

Deep learning based detection of vehicles, lane and street sign for behavior cloning in autonomous car

Nguyen HuuNhat Minh, Phan Xuan Hieu*, Tran Quoc Long

University of Engineering and Technology, Vietnam National University, Hanoi, Vietnam

Abstract

With the growth of Artificial intelligence and Machine learning, the amount of research for autonomous vehicle is also growing nonstop. But a self-driving system is far too complex with the hardware integration, LiDAR and RADAR involvement, how exactly are these machine learning algorithms being applied, where can a machine learning researcher start to research self-driving car. This report will cover how traditional machine learning methods and state of the are deep learning (semantic segmentation, convolutional neural network - CNN) are being applied to build autonomous cars. The algorithm uses image processing techniques to label lane pixels, classical machine learning and Histogram of Oriented Gradient (HOG) to label vehicle and street sign pixels. Then the data is used to train a semantic segmentation network to extract features for a final CNN to combine with the original image and predict driving command.

Received ……, revised …….., accepted ……

Keywords:

1. Introduction

The recent developments in deep learning had been applied to many aspects of the world. The autonomous car industry is a promising field of research with great investments from large corporations. For a self-driving car to work safely and effectively, it needs both state of the art software and hardware integration.

The machine learning algorithms can be applied in this field include but are not limited to: object detection, object tracking, 3D image reconstruction, semantic segmentation, …

Object detection can be applied to detect obstacles that can block the vehicle's movements.This includes other vehicles (cars, motorcycles, bikes), pedestrians, other anomalies that can appear on the road. With recent advances in deep learning, the algorithm can detect vehicles or pedestrians with high accuracy given a large enough dataset. However, if an unusual object suddenly appears in front of the vehicle, it is uncertain whether or not the algorithm can detect that object. This is an extremely dangerous scenario that can challenge any "good" model. One option to minimize risk of software failure is to use Lidar and Radar to detect presence of obstacles. This option is considered an optimal addition to help with object that are difficult for machine learning models to visually detect.

Object tracking is used to track detected vehicles and calculate their relative speeds, therefore determining their trajectories. This require highly precise modeling and stable camera angle. Effective tracking can reduce the work for the detection model and help improve the overall speed of the algorithm.

3D image reconstruction is used to construct a 3 dimensional view of the lane so that the path planning algorithm can derive a

*Corresponding author. E-mail.:.....

path for the vehicle. In an autonomous car, the reconstruction process can be supported by either a RGBD camera or the distance calculated by Lidar and Radar system. Current deep learning method for reconstructing 3D images can be applied but their accuracy is not yet reliable for an autonomous car to utilize therefore further hardware utilization is needed.

Semantic segmentation has many uses for self-driving cars like identifying lane pixels, obstacles, … Semantic segmentation can also be used to determine which regions of the image the vehicle can reach and which regions the vehicle should not move toward. The biggest drawback for semantic segmentation models is the cost of data construction. A dataset for semantic segmentation require label for each pixel over hundreds of thousands of images. In this research we also propose a method for construction a large dataset for semantic segmentation.

The rest of this paper is organized as follows. Section 2 describes the proposed method in detail. Experimental results are discussed in section 3. Finally, section 4 concludes this paper.

2. Proposed method

In this paper, we propose a feature engineering step before feeding a convolutional neural network. Convolutional neural networks are option trained end to end, this allows the networks to select which features to learn on their own. The feature maps only depend on the dataset, the batching mechanism, model initialization and hyper-parameters. The programmer does not have control over which feature will be learnt.

For a self-driving car, the important features such as lane, obstacles and street signs are highly correlated to the driving commands. Therefore, we propose to combine these 3 features with the original image to form a 6 channels image. And uses theses 6 channels images to train a convolutional neural network predicting speed, steer and brake.

The idea derives from the success of the ResNet's [4] residual blocks. The residual architecture got its name from its unique method for passing information through each layer. After passing through a block the information is added with the input to the layer to keep both the original information and the filtered features. Concatenating these features to the original image does not increase the amount of information the image contains as the features are derived from the original image. But divert the learning process to learn more from the 3 features more than the original image.

To calculate the features, we used image processing techniques to label lane pixels, classical machine learning to label vehicles and street sign pixels. Then the original images and the calculated features are passed to a semantic segmentation network to calculate these feature much faster than the original method (using image processing and classical machine learning).

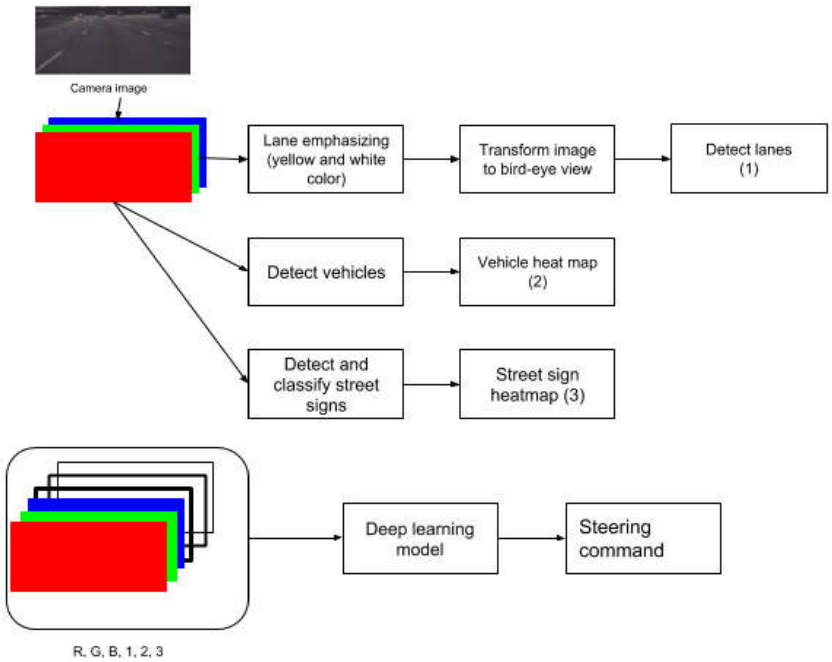
The algorithm for lane detection uses image processing techniques to identify lane pixels. This algorithm is based on a greedy assumption that lane markers are often white and yellow, made visible at most light conditions. This does not require any training data, but is dependent on the fixed camera angle and image size.

For vehicle detection, we used the dataset from Udacity[1]. This dataset contains roughly 30000 images from 2 classes (vehicle/ not vehicle). This is especially fitting for classical machine learning method that utilize binary classification on each proposal window. The process then creates a heat map that indicates vehicle presence in the image.

The street sign detection and classification use the German street sign dataset[2] which contain over 50000 images from 40 classes of street signs. Some different algorithms are attempted for this classification problem were Support vector machine (SVM) [5], Random forest [7], Extremely randomized trees [3].

The dataset for behavior cloning is Comma.ai open dataset[3] of 7.25 hours of highway driving. This dataset comes with steering angle, speed, brake and other attributes of the vehicle.

[1] https://github.com/udacity/CarND-Vehicle-Detection
[2] http://benchmark.ini.rub.de/
[3] https://github.com/commaai/research

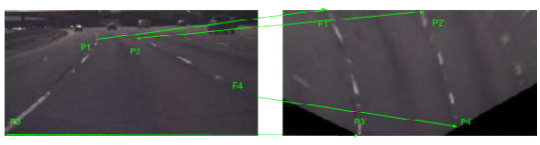


a. Lane detection

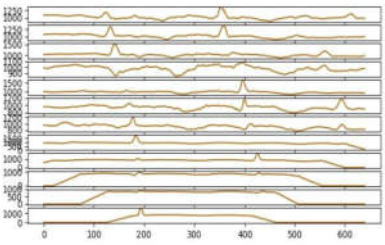
The lane detection algorithm makes use of the fact that lane markers are often white or yellow and are made stand out from the rest of the road so that driver can spot them at all light conditions.

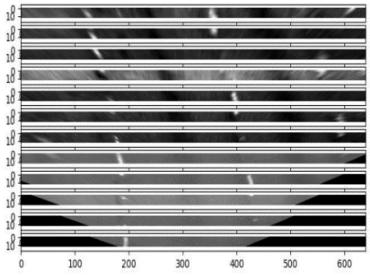
The algorithm is divided into 7 steps:

- Threshold and emphasize yellow pixels
- Threshold and emphasize white pixels
- Transform the image to bird eye view

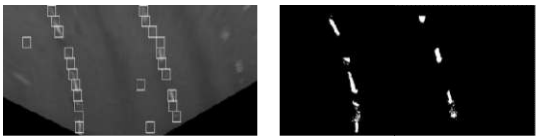


- Split the image to horizontal stripes and detect peaks in brightness histograms





-



- From the window containing those histogram peaks find the pixels that belong to the lane markers
- Fit a second order polynomial to the pixels of each lane markers
- Draw the lane lines on the bird eye image and transform it back to regular view

b. Vehicle detection

Unlike other mainstream object detection problems, the camera is mounted to the windshield of an Acura ILX 2016 and has a fixed angle over the road ahead. Another property that make this problem easier to solve is that most vehicles have a somewhat similar size. This plus the fixed camera angle means that we can limit the number of proposal windows for detection. The rest of the problem is to determine whether a proposed window contain a vehicle or not.

As summary, the problem is tackled by

- Proposing detection windows
- Binary classification whether a window contain a vehicle or not

- Proposing detection windows

Since the camera angle is fixed, the further a vehicle is to the camera, the smaller it will appear in the image.

Therefore, the proposal window's size is proportional to the its position on the horizontal axis. Plus, two adjacent windows will only need to overlap at most ⅔ of each other. This limits the number of proposal windows greatly.

- Vehicle/ Not vehicle classification

This is one of the classic image classification problem. But one of the major limitation is that the dataset is limited to the 16000 images. It is worth to mention that this dataset cannot be extended nor augmented without heavy manual labelling. One may consider vehicle categories from the CIFAR dataset, but the images in the data is taken at drivers' view on the road (most of which are rear and side view of cars and trucks) and the CIFAR dataset contain clear images of vehicles (most of which are frontal side view).

We attempted several methods of image classification and found that the ensemble model Extremely randomized trees obtained the best result:

Model	F1 Score	Comment
Shallowest, most parameter pruned convolutional neural net	91%	Actual performance was still horrible due to overfitting
HOG features + SVM	70%	Failed even with multiple parameter settings. Diagnosis: images might form too many clusters that made it impossible for an SVM to separate
HOG features + Ensemble (random forest, extremely randomized trees)	96%	Random forests tend to over-fit, but with the right pruning methods this gave the best result as it can handle clusters as long as there are sufficient examples.

c. Street sign detection

Like the vehicle detecting problem, street sign detection belongs to the class of object detection. There are plenty of work done regards to this class of problems already. In this research, we propose a way to decrease the number proposal windows needed to detect street signs using the lane line feature.

Clearly, we are not looking for street signs on the road or on other travelling vehicles, this limits the number of windows we are sorting through by plenty. Added with the relative size of the street signs by perspective of the camera, we can reduce the number of proposals further, therefore increases the speed of the algorithm.

The detected traffic signs are then passed to another algorithm for classification. In this research, we covered this with another random forest classifier. The difference is that the input to the model is both the shrunk version of original image and the histogram of oriented gradient because the color of the street sign is a good feature for classifying.

d. Speed up with semantic segmentation

The process of detecting lanes given an image input takes up to 1 second to run on a 2.6 GHz Intel i5 CPU, unacceptable for any autonomous vehicles. This part of the algorithm heavily depends on heuristics and greedy assumptions. The process is difficult to be replicated and parallelized on a GPU. Difficult, not impossible if we attempt to write CUDA code for every steps of the algorithm. But this is highly unnecessary.

We can use the process defined earlier as a mean to label every images with their corresponding label of lane, vehicle heat map and street sign heat map. Then we use a neural network structure for semantic segmentation to speed up the process of collecting these features.

There has been plenty of research on semantic segmentation using deep learning. A few can be named as all time's classic: Fully Convolutional Network [8], SegNet [1], PSPnet[9], RefineNet [10], ...

The SegNet architecture would easily handle a sequence of 320x160x3 images at 20Hz, encoder decoder structure for the win.

e. Behavior cloning

A Convolutional Neural Network is used to predict the driving commands based on the images from our windshield camera and their interpreted information.

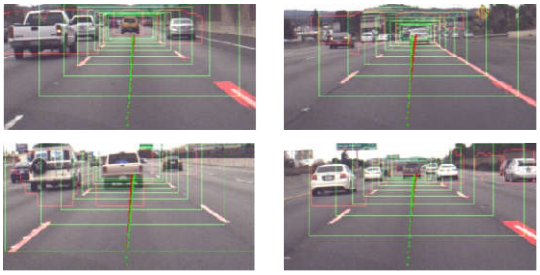
The 320x160x6 is passed along 3 convolutional blocks, each contains a Convolutional layer, Batch Normalization, Rectified Linear, Swish activation and a pooling layer.

The 3 kernels for the convolutional layers are 7, 5 and 3 respectively and the pooling windows is 2x2 max pooling.

After these convolutional blocks, the features are passed to 3 dense layers and applied a final Tangent Hyperbolic activation.

The dataset from Comma.ai provided us with the measurements of the Acura ILX on a 100 Hz time-base. After extracting the measurements and align them with the correct frame, we squash the vehicle velocity, steering angle and break to the range [-1, 1].

3. Experimental results



As we have shown, the algorithm is not flawless, there are many errors when it comes to vehicle detection. And with the dataset mostly being highway driving, there are rarely any cases that the algorithm can detect a traffic sign. Even though these are some of the best images we humbly present to demonstrate our awesome algorithm, it is nowhere near perfect and made plenty of mistakes when exposed to extreme light conditions or edge cases.

There are no standardized metrics to calculate the accuracy of driving behaviors currently available, therefore it is difficult to determine how good the model is, except from manually looking at the result or testing with an actual car. However, the accuracy of some individual models can be measured.

The validation process was done with k-fold cross validation 5 times with a test size equals to 20% of the original dataset. And the validation metric for classification is the F1 score:

$$Precision = \frac{true\ positive}{true\ positive + false\ positive}$$

$$Recall = \frac{true\ positive}{true\ positive + false\ negative}$$

	Actual range	Negative	0	Positive
Velocity	[-50, 50] km/h	Driving in reverse	Vehicle standing still	Driving forth
Steering angle	[-40, 40] degree	Steer left	Steer straight	Steer right
Brake	[0, 8000] units	0	0	Positive brake

$$F_1 = 2\ \frac{Precision * Recall}{Precision + Recall}$$

Model	Brief description	F1-score
Vehicle/ Not vehicle classification	Extra_Trees classifier + HOG	96%
Street sign/ not street sign classification	Extra_Trees classifier + HOG + down_sized image	97%
Street signs classification	Extra_Trees classifier + HOG + down_sized image	94%

All in all, our method for behavior cloning has a better rate of convergence than the end to end method that Nvidia proposed. The number of epochs need to reach a convergence of training and testing loss was decreased to ⅔ of the end to end method. However, this has a tradeoff.

The training time is decreased but the execution time for each prediction is increased because of the extra features to calculate at each steps. This is not a big issue because the feature engineering process and prediction can both be accelerated by GPU. The extra features proposed in this report are highly valuable and should not be omitted by training end to end. For example, the lane lines and vehicle heat map can be used with Lidar and Radar for path planning and street sign information can be used with rule-based algorithms go set speed limits, etc.

The feature engineering process was inspired by the ResNet's [11] residual blocks. The residual architecture got its name from its unique method for passing information through each layer. After passing through a block the information is added with the input to the layer to keep both the original information and the filtered features. We looked into this idea, and used the entire preprocessing network as a "learning block" and instead of adding up with the original information, we concatenated them to create 6 channels for training the convolutional network.

4. Conclusion

Our methods had shown some promising results. There is a tradeoff between faster training time using the added features and the slower speed for each prediction because at each turn we have to calculate the features again. This trade off if completely acceptable if we want to construct some safety measures using lanes, obstacles or traffic signs (which we most likely want to do). Therefore, our method of feature engineering for a neural network did show its advantage in training efficiency.

This could theoretically be applied to other image classification/ regression methods as well if we know which feature has strong correlation with the desired output.

The results for the individual components are not as good as state of the art models. But this work has provided good foundation for deeper research into the field. In the future, this project can be expanded and integrated to a vehicle for testing. Most of the core algorithms were already implemented, the remaining work would be to implement a path planning algorithm and add hardware integration. Other work regarding the algorithm can be mentioned is to improve the accuracy of individual machine learning algorithms.

The vehicle detection algorithm was implemented with an Extremely Randomized Trees classifier with Histogram of Oriented Gradient features. This was the optimal option when the training dataset for detection contained only 30000 total images of 2 classed (vehicle / not vehicles). Better accuracy can be obtained with state of the art object detection neural network once more data is collected. Another approach would be to fine-tune a pre-trained neural network with the limited amount of data. The dataset for this detection need to have the same prevalence as seen from the driver's perspective instead of general vehicle images like what seen in the CIFAR dataset.

The same can be done for the street sign detection and classification algorithm. The current method used classical machine learning and some feature engineering. State of the art deep learning can be applied by fine-tuning top class networks with the dataset and this could improve the performance of the model on both speed and accuracy.

More accurate driving command can be obtained by reconstructing the environment surrounding the vehicle either with machine learning techniques or with Lidar and Radar systems. The input to the model would then be a 3 dimensional image and the CNN would then perform 3d convolution and pooling. On theory, this would add information to the model input and give better accuracy.

Acknowledgement: We would like to thank A/Professor Le Thanh Ha for reading the manuscript and giving useful comments.

References

[1] Badrinarayanan, Vijay, Alex Kendall, and Roberto Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation." *IEEE transactions on pattern analysis and machine intelligence* 39.12 (2017): 2481-2495.

[2] Bojarski, Mariusz, et al. "End to end learning for self-driving cars." *arXiv preprint arXiv*:1604.07316 (2016).

[3] Geurts, Pierre, Damien Ernst, and Louis Wehenkel. "Extremely randomized trees." Machine learning 63.1 (2006): 3-42.

[4] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[5] Hearst, Marti A., et al. "Support vector machines." IEEE Intelligent Systems and their applications 13.4 (1998): 18-28.

[6] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint arXiv:1502.03167 (2015).

[7] Liaw, Andy, and Matthew Wiener. "Classification and regression by randomForest." R news 2.3 (2002): 18-22.

[8] Lin, Guosheng, et al. "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation." *IEEE Conference on Computer Vision and Pattern Recognition* (CVPR). 2017.

[9] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2015.

[10] Ramachandran, Prajit, BarretZoph, and Quoc V. Le. "Swish: a Self-Gated Activation Function." arXiv preprint arXiv:1710.05941 (2017).

[11] Ren, Haoyu, and Ze-Nian Li. "Object detection using edge histogram of oriented gradient." *Image Processing (ICIP), 2014 IEEE International Conference on. IEEE*, 2014.

[12] Zhao, Hengshuang, et al. "Pyramid scene parsing network." *IEEE Conf. on Computer Vision and Pattern Recognition* (CVPR). 2017.