Object Detection and Classification in Driving

Xia Shang
Department of Mechanical Science and Engineering
University of Illinois at Urbana-Champaign
Email: xshang3@illinois.edu

Yuan Cheng
School of Information Science
University of Illinois at Urbana-Champaign
Email: yuanc3@illinois.edu

***Abstract*—In this study, we aim to develop a model to detect and classify traffic objects using a camera during driving. We applied four classification methods during model training and validation process to compare the prediction accuracy. In addition, we tested different image preprocessing approaches to augment the features that improve modeling performance.**

The dataset for model training and validation contains 6,000 instances with four labels of not-car, car, truck, and pedestrian. The best performing classifier is based on the algorithm of convolution neural network, which consists of two convolution layers and three fully connected layers. A maximum pooling layer was used after each convolution layer, and a dropout step was used after each of the first two fully connected layer. The model achieved around 100% accuracy on training dataset and 87% accuracy on the validation dataset with ELU activation function and ADAM optimizer. In addition, the proposed model can be trained within 2 hour without GPU on a personal laptop.

Finally, we select the model that yields the highest prediction accuracy to detect cars, trucks, and pedestrians using driving videos collected from a camera. A sliding window approach that samples small windows in a single frame is used to detect objects.

I. Introduction

Object detection and classification of driving images play an important role in applications of autonomous car, traffic control, drive assistant, and security monitoring. In the U.S., around 35,000 people die due to traffic accidents every year. Autonomous cars could reduce 90% accidents in the U.S., according to multiple reports, saving 3 million lives per decade and eliminating $190 billion cost due to damages and health-care cost. It is estimated that autonomous vehicles will reach 25% of the global market in 15 to 20 years. [1]

Self-driving car is typically equipped with sensors, cameras, and high definition maps (at centimeter-level accuracy), which are much more precise than GPS coordinates. Sensors such as light detection and ranging systems (LiDARs) are very expensive. On the other hand, camera imagery requires advanced artificial intelligent models that produce accurate results in real-time.

In this study, we aim to develop robust and fast modeling approaches to detect and classify objects based on images obtained from a camera that is mounted in front of the car.

II. Data preparation

A. Labeled database

In this study, two labeled vehicle image datasets are used in this study.

	xmin	xmax	ymin	ymax	Frame	Label
0	785	533	905	644	1479498371963069978.jpg	Car
1	89	551	291	680	1479498371963069978.jpg	Car
2	268	546	383	650	1479498371963069978.jpg	Car
3	455	522	548	615	1479498371963069978.jpg	Truck
4	548	522	625	605	1479498371963069978.jpg	Truck
5	1726	484	1919	646	1479498371963069978.jpg	Car
6	758	557	807	617	1479498371963069978.jpg	Car
7	633	561	680	597	1479498371963069978.jpg	Car
8	682	557	718	593	1479498371963069978.jpg	Car
9	710	540	836	665	1479498372942264998.jpg	Car

Figure 1. Screen-shot of a part of the csv file in the CrowdAI drving dataset. The CrowdAI dataset contains driving video frames and a csv file, which shows the coordinates of objects in each frame and the corresponding labels (i.e., Car, Truck, and Pedestrian).

The first dataset is the Vehicle Image Database from Graz University of Technology (GTI). [2] This data set contains around 7400 images of cars and 7000 images of not-cars. All the pictures in the GTI dataset are resized into $64 \times 64 \times 3$.

The second dataset is an annotated driving dataset generated by CrowdAI. This dataset contains 9423 pictures of driving videos with size $1200 \times 1800 \times 3$ and a encoded csv file, as shown in Figure 1. In the csv file, the x/y coordinates of certain objects in every frames are listed with their corresponding labels, which shows Car, Truck, and Pedestrian.

1) Testing data: A few frames from the CrowdAI driving dataset are used in the model testing stage. In addition, we collect driving frames for model testing using a camera (FITFORT 30 FPS) mounted in the car around the campus of UIUC.

Figure 2. Left: image of the testing camera (FITFORT 30FPS) used in this study. Right: a frame of driving video taken in Urbana, IL.

Figure 3. Example of the best performing approach of image pre-processing using four images from each label. The initial images were first resized into the size of 64 $\times$ 64 $\times$ 3, and converted to gray-scale. Then we adjusted the local contrast with normalization to augment the patterns. The size of each image after normalization is 64 $\times$ 64. Note that this approach is only tested in the classifier of convolution neural network.

2) Dataset for training and validation: Both of the GTI and the CrowdAI datasets are combined to yield four labels: Not-car, Car, Truck, and Pedestrian. Training and validation dataset each contains 6,000 resized images that consists of 2,500 not-cars, 2,500 cars, 500 trucks, and 500 pedestrian.

III. Feature processing

We tried different approaches to pre-process the features before using them to train classifiers. The images for car, truck, and pedestrian were first resized to 64 $\times$ 64 $\times$ 3 (as shown in the left 16 images in Figure 3).

1) Converting to gray-scale: All the images in the dataset were converted to grey-scale image since all the objectives in this study should be color invariant (middle 16 images in Figure 3).

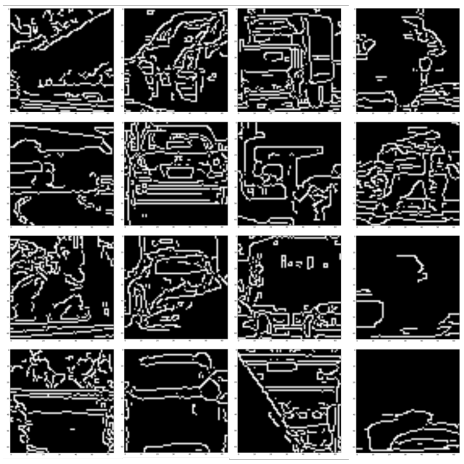

Figure 4. Features of edges only. Note that the raw images that generate the features of edges can be seen in the poster and they are not the same as the images in Figure 3. This feature was tested in the first three classifiers.

2) Adjust the local contrast: Some features in the gray-scale images are difficult to recognize due to low contrast with the surrounding environment. To augment the key features, the local contrast was adjusted by applying a local contrast function package. The resulting features are shown in the 16 images on the right side of Figure 3 as an example.

3) Normalization: Before feeding into different classifiers, all the images were normalized to the range of -1 to 1 using the equation $\frac{2*x}{256} - 1$.

4) Using edges only: One approach we tried in the study (which is also shown in the poster) is to use the edges of the objectives, as shown in Figure 4. The features of edges are obtained by applying the Canny functions in OpenCV package, which is based on the gradient of pixel changes in different directions.

5) PCA featuers: Another approach we tested is using PCA to reduce the number of features from the raw images. Figure 5 shows the first nine principle component features.

IV. Classifier: Fully connected Neural Network

We first applied three basic classifiers of fully connected neural network with total number of layers equals to 1, 2, and 3. Note that the 1-layer neural network is a linear classifier.

After briefly tuning the modeling parameters, the final parameters that yields the best performance is the following:

- Learn rate: 0.0005
- Mini-batch size: 128
- Epochs: 100
- Hidden layer number of features: 256 (all hidden layers have the same dimension)
- Activation function: relu
- Final layer activation: softmax
- Optimization algorithms: Gradient descent optimizer

Tensorflow was used to construct all classifiers in this study. Since the training and testing processes are mainly done on a

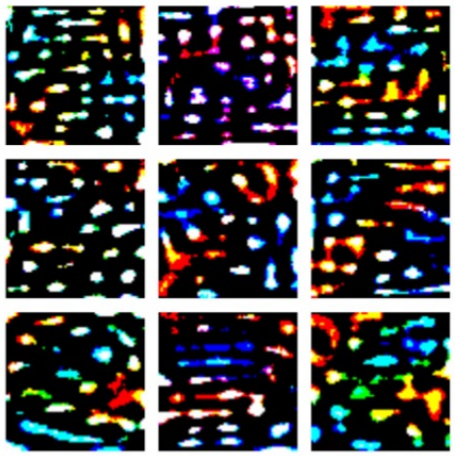

Figure 5. First 9 pca features. This method was used in the fully connected neural network classifiers.

personal laptop (2012 MacBook with 2.9 GHz Intel Core i7, 8 GB 1600 MHz DDR3), the 6,000 instances were divided into small batches (minibatch) that feed to the classifier during training. All data type was converted to float32 to further reduce memory usage.

Figure 6 shows the changes of validation accuracy during model training for the three neutral networks with different number of layers using the raw image features. Monotonic increase in validation accuracy was observed for the 1-layer and 2-layers neural net. The 3-layers neural network classifier reached around 70% validation accuracy after 10 epochs and saturated. Overfitting was not observed during the training process.

Next we repeat the training and validation process for the three neural network classifiers using three other features: gray-scale images, edge-only images, and pca features.

The validation accuracy after 100 epochs of the four different features are summarized in Figure 7. The 3-layers neural net using raw colored images results in the highest

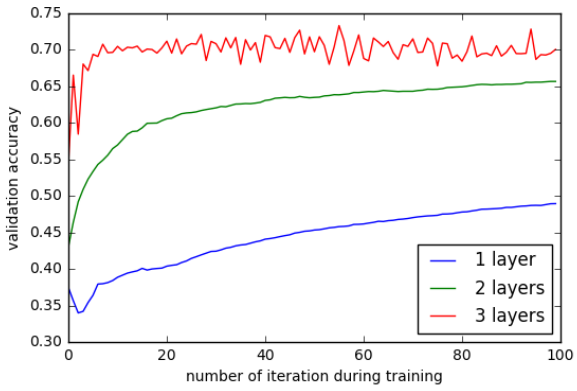

Figure 6. validation accuracy in the first 100 epochs during model training for three neural network classifiers. The raw images are used as input features.

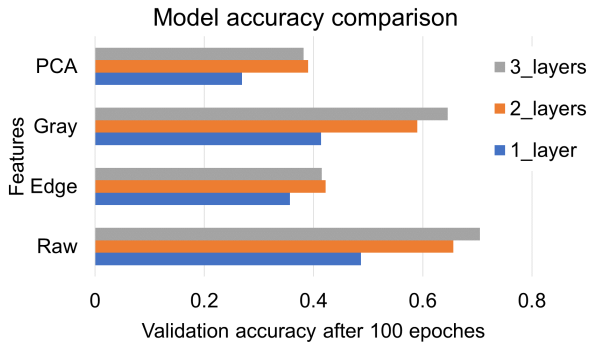

Figure 7. Comparison of model accuracy for three neural network classifiers using different features after training.

validation accuracy at 71%. However, 71% validation accuracy is still pretty low. This is probably due to the nature of neural network, which is not very good at distinguishing specific shapes in image classification. For example, if the same patterns appears at different location in the image, neural network can not recognize it. In the next section, we construct a convolution neural network (CNN) that uses weight sharing algorithm to further improve the validation accuracy.

V. Classifier: Convolution Neural Network

With a large number of data, CNN has been proven to be a successful and wide-used algorithm that is very good at recognizing image patterns. [3] [4] Compared to traditional neural network like the classifiers in the previous section, a convolution layer is obtained by scanning an image with small size neurons that share the same weights. As a result, CNN can recognize a pattern that locates at different position in an image.

The state-of-art classifier for object detection and classification typically involves very deep CNN architectures. For example, the second version of the You-Only-Look-Once (YOLO) contains 19 layers of convolution layers, 5 maximum pooling layers, and a few fully connected layers. [5] The training process for such a model typically requires a week on GPUs. In this study, we construct a relative shallow and simple CNN architecture that contains two convolution layers, two maximum pooling layers, and three fully connected layers, as shown in Figure 8.

The input features are the contrast augmented gray-scale images after normalization (see Figure 3). During training, we reduce the mini-batch size to 64 so that the whole process can be done on my laptop. The other parameters used in the final version of the model is:

- Conv1: width/height = 5, strike space = 2, zero-padding
- Conv2: width/height = 5, strike space = 1, zero-padding
- Max-pooling: width/height = 2, strike space = 2, zero-padding
- Dropout rate: 60%
- Optimization algorithms: Adam-optimizer
- Learn rate: 0.0005

Layers	Size	Activation
Input	64x64x1	
Conv1	32x32x6	elu
Max pooling	16x16x6	
Conv2	16x16x16	elu
Max pooling	8x8x16	
L1, fully connected	1024	elu
Dropout		
L2, fully connected	512	elu
Dropout		
L3, fully connected	4	softmax_cross_entropy

Figure 8. Convolution neural network architecture used in this study.

- Mini-batch size: 64
- Epochs: 100

The Adam algorithm is a gradient-based stochastic optimization approach based on adaptive estimates of moments. [6] After switching from the conventional gradient descendant optimization to the Adam algorithm in tensorflow, the training speed increased significantly. The validation accuracy achieves about 83% after 5 epochs and saturates at 87% after 20 epochs (as shown in Figure 9). It seems that overfitting still exists as the training set accuracy reaches 100% towards the end the training process, even though the validation accuracy maintained at 87%.

To further analyze the model prediction, we plot the confusion matrix (without normalization and with normalization) of the four classes after training, as shown in Figure 10. The trained CNN model shows good accuracy on the classes of Not-car and Car. However, the validation accuracy is low for the truck (64% accuracy) and pedestrian (72% accuracy). This

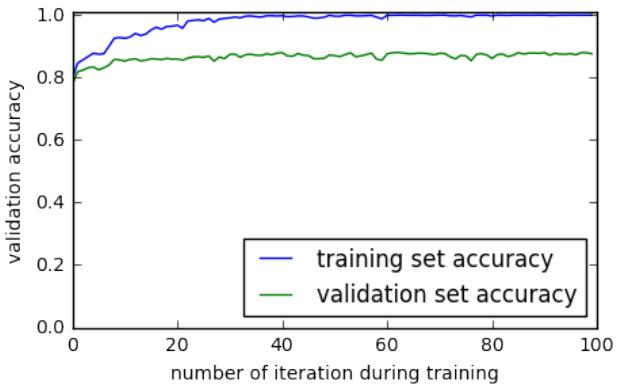

Figure 9. Convolution neural network architecture used in this study.

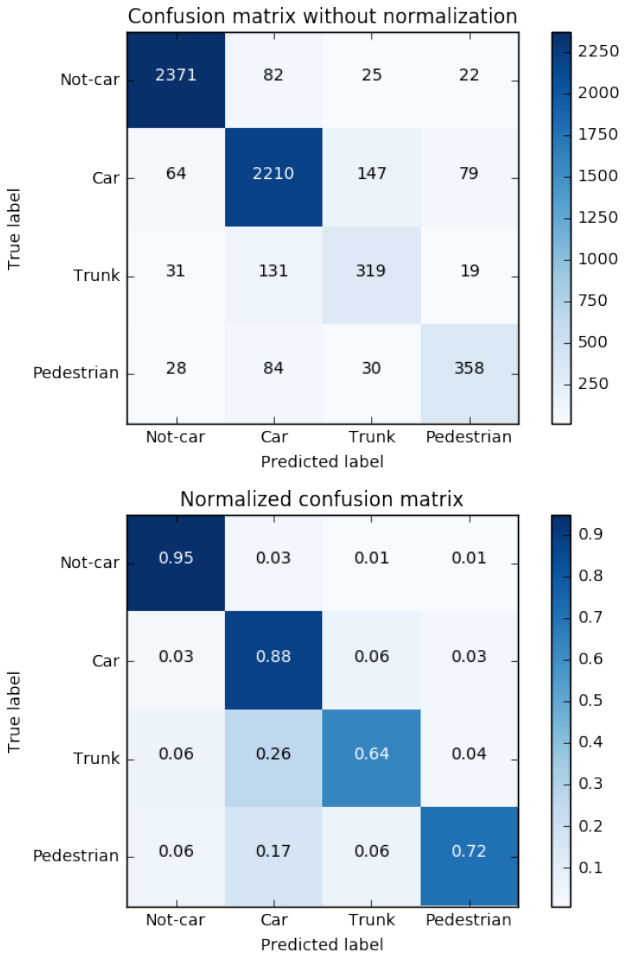

Figure 10. Visualization of the confusion matrix generated using the trained CNN model on the validation dataset. Top: without normalization; bottom: with normalization.

is probably due to 1) the sample size for truck and pedestrian is five times smaller than the other two classes; and 2) the initial images for truck and pedestrian are typically long, which may deform after resizing into squared images.

VI. Detection and Model Testing

A sliding window approach is applied to detect objects in a full-size driving image.

In the first step, we generated a number of small sliding-window to sample sub-images from the full-size driving frame, as shown in Figure 11 (a). Windows of four different sizes are generated at different location to capture objects at different distances. For example, for objects far away from the camera, we only need very small windows aligned along the central line of the image. The total amount of windows depends on the size of the driving images and the purpose of the detection. For example, we probably need vertical long windows instead of squared windows to detect pedestrians.

After resizing all the sampled images into the size of $64 \times 64 \times 3$, we processed all the images using the same approach as in the data pre-processing stage (see Figure 3). Then we

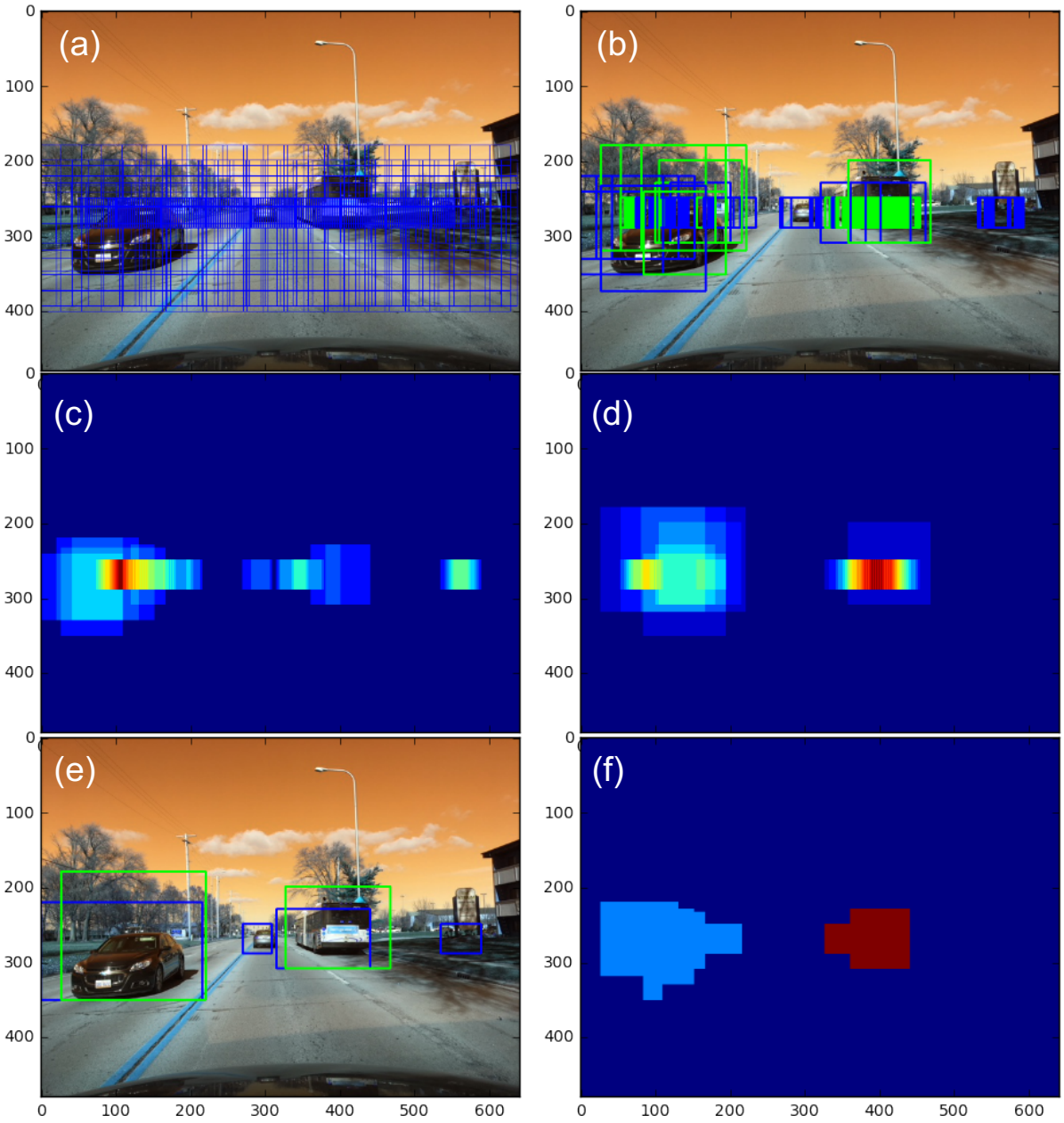

Figure 11. Example of the how object detection works in this study. (a) A number of squared windows (around 200 - 1000 in total depending on image size) are generated to sample small images first. (b) All the images are fed to the trained classifier. The coordinates of the window is saved if the model classifies either a car, a truck, or a pedestrian with probability higher than a threshold (which is set at 90% for most of the time. (c-d) A heatmap for each class is generated. In this case, image of (c) is the heatmap of car, and image of (d) is the heatmap of truck. (e) Another threshold of a minimum count can be applied to determine the final detection. (f) A future improvement in the detection stage is to further process the area that shows positive detections of multiple classes. In this case, two areas show detection of both a car and a truck.

applied the pre-trained classifier to treat the images, producing the prediction labels and the corresponding accuracy. If the accuracy is higher than a threshold (which is set at 90% in most cases), the coordinates of the window is used to increment the values in a heatmap that records the total counts for each class (Figure 11 c-d).

In cases of a lot of false positive detections (as shown in Figure 12), we applied another threshold of minimum count to final determine the regions of object, which is plotted in Figure 11 e. One limitation we observed is that some regions (Figure 11 f) showed positive detection of multiple classes. A future improvement is to compare the maximum count in those regions to decide which object the regions represent.

VII. Comment on Training and Testing Speed

All of the computation tasks in this report were done on a person laptop (2012 MacBook 2.9 GHz Intel Core i7, 8

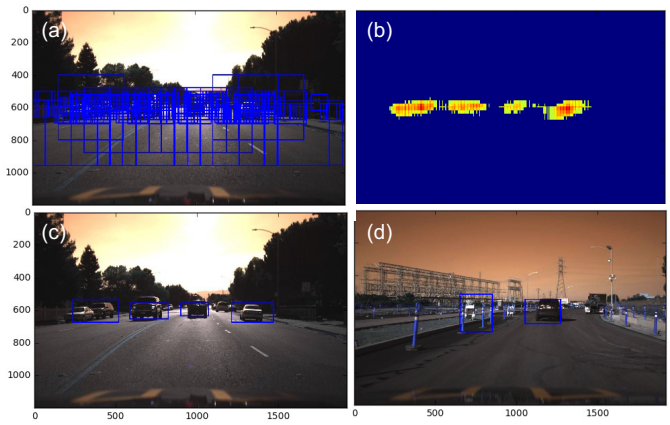

Figure 12. Detection of cars using the 3-layers neural network classifier. (a) Many false positive detections were shown after the classification step. (b-c) A threshold of minimum count has to be used in the heatmap to produce correct detection. (d) False positive test still exists when structural objects shown in the background.

GB 1600 MHz DDR3) without GPU usage. The training process for the best performing model (CNN classifier) took less 2 hours to finish 100 epochs. Considering the validation accuracy saturated in the first 20-30 epochs, the training can be finished in about 20 minutes with a validation accuracy of around 87%. The testing stage takes only a few seconds depending how many sliding-windows are used during the detection step. This implies that this model can be used in applications such as initial testing of different datasets or fast prototyping applications.

VIII. Conclusion

In this study, we developed models to detect and classify cars, trucks, and pedestrian using video frames during driving. We develop four different classifiers and compare their prediction accuracy using different feature processing approaches. The model that is based on convolution neural network shows the highest validation accuracy and fast speed during training and testing. A sliding window approach was used to localize and detect objects in driving images.

The model still surfers a number of limitations that can be improved in the future. 1) False-positive detection was shown when structural objects exist in the background, such as houses, buildings, steel frames, and commercial advertising stands. One solution is to divide the class of "Not-car" into more detailed sub-classes. 2) The model is still not very good at detecting pedestrian in real driving image. This is mainly due to the fact that the pattern of pedestrians in the training and validation dataset were resized into squared ones. One possible solution is to generate additional vertical long sliding-windows in the detection stage. 3) The model is hard to separate two objects that are overlapped (e.g., one car is in front of another one). 4) The sliding window approach slows down the detection process as it generates a lot of images without any objects in the first place. This can be improved by approaches such as determining bounding box or decide regions of objects.

References

[1] D. M. West, "Moving forward: Self-driving vehicles in china, europe, japan, korea, and the united states," 2016.

[2] J. Arróspide, L. Salgado, and M. Nieto, "Video analysis-based vehicle detection and tracking using an mcmc sampling framework," *EURASIP Journal on Advances in Signal Processing*, vol. 2012, no. 1, p. 2, 2012.

[3] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1316–1322.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[5] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," *arXiv preprint arXiv:1612.08242*, 2016.

[6] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.