A Novel Approach Determining Driving Assistance Using Deep Convolutional Neural Networks

Sahana G M[1], Sahana M[2], Suma R[3], Varsha P [4], Mrs. Vidhya K[5]
[1, 2, 3, 4]Dept of Information Science and Engineering
[5]Asst.Prof,, Dept of Information Science and Engineering
[1, 2, 3, 4, 5] East West Institute of Technology, Bangalore, India.

Abstract- *Intelligent vehicle systems, such as advanced driving assistance systems, are relatively popular nowadays, which facilitate the timely prevention of driving-related accidents and human injuries caused by impaired driving. This paper presents a concept for testing camera based ADAS, in order to reduce time and cost in the development phase. By adapting an existing virtual environment for the camera system, identifying and enhancing the important features for the testing detection function. To address the large object scale variation challenge, deconvolution and fusion of CNN feature maps are proposed to add context and deeper features for better object detection at low feature map scales. In addition, soft non-maximal suppression (NMS) is applied across object proposals at different feature scales to address the object occlusion challenge. As the cars and pedestrians have distinct aspect ratio features, we measure their aspect ratio statistics and exploit them to set anchor boxes properly for better object matching and localization.The proposed CNN enhancements are evaluated with various image input sizes by experiments over KITTI dataset. Experiment results demonstrate the effectiveness of the proposed enhancements with good detection performance over KITTI test set.*

Keywords- Convolutional Neural Networks; KITTI Dataset.

I. INTRODUCTION

Visual object detection is a long standing and important research problem for computer vision, with a wide range of real world applications. Advanced driver assistant systems[1] have been implemented in many vehicles to help increase both the safety of drivers and pedestrian. The related technology is also used to develop self-driving cars. Three features including traffic sign recognition, lane deviation detection and car make identification in ADAS that our team built. Traffic sign recognition is crucial to remind the drivers the traffic signs ahead in order to prevent accidents caused by the traffic sign ignorance in the bad weather condition. It can be seen as another eyes to guarantee driving safety on the road. Lane deviation detection system provides lane detection and stability determination, giving drivers warning in the condition that car drifting into directions out of lane. Car make identification is a useful feature when drivers are interested in the make and model of car in their front.Despite fast growth of CNN in object detection over datasets with a large number of object classes, real time visual object detection in driving environment is still very challenging. It is observed that the object detection performance of thepopular CNN detectors including Faster-RCNN[7]. and SSD[6]without modification is not very good over the KITTI benchmark datasets[1]. In the existing multi-scale CNN models, feature map from feature output scales are processed separately to predict existence of objects at fixed scales. In this paper deconvolution of CNN features is applied at smaller feature output scales, which is further fused with features at larger feature output scales, to provide richer context for object detection at individual feature output scale. Such enhancement can effectively address the large object scale variation challenge.The proposed CNN enhancements are evaluated with various image input sizes by experiments over KITTI dataset. Traffic sign recognition is crucial to remind the drivers the traffic signs ahead in order to prevent accidents caused by the traffic sign ignorance in the bad weather condition. It can be seen as another eyes to guarantee driving safety on the road. Lane deviation detection system provides lane detection and stability determination, giving drivers warning in the condition that car drifting into directions out of lane. Car make identification is a useful feature when drivers are interested in the make and model of car in their front.The proposed CNN enhancements are evaluated with various image input sizes by experiments over KITTI benchmark dataset. Good detection performance improvement is observed with both individual and combined CNN enhancements. Com-pared to the published works over KITTI benchmark test dataset our proposed method ranks the first for pedestrian detection category “Easy” and second for categories “Moderate and “Hard”, and is the fastest among the top ten ranked published methods. The object detection time with a GPU computer is 0.08 second per 384 1280 sized image, which can satisfy the real time requirements of driving safety applications.

II. ARCHITECTURE

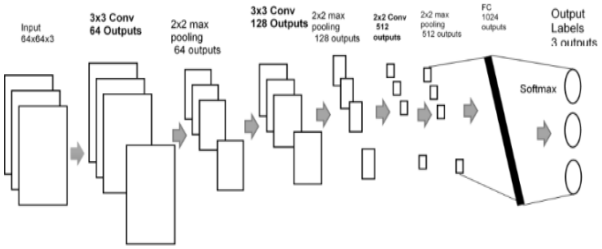
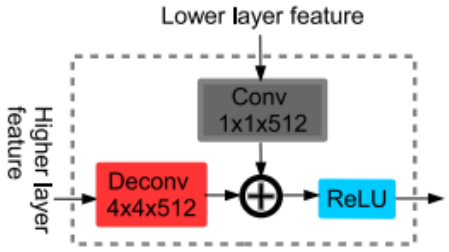


Fig1. Proposed System Architecture

The input to the CNN is an image with size H W D, where H and W denote image height and width in pixels, and D denotes the number of color components.The architecture of the proposed system contain an convolutional layer Max polling and an fully connected layer . The input is given in the form of an image which of matrix form once the input is given goes to the convolutional layer were all the unwanted noise or image will be filtered since the convolutional layer contain n number of filters due to this layer the unwantedly recognized image or noise will be removed. Further the output that has come from the convolutional layer is taken as the input for max polling layer which is responsible for categorizing the picture of under which category it belongs to after completing this operation or process the output from the max polling layer or sub sampling layer is taken as input for again convolutional layer these process continues for certain number of times and finally comes to an fully connected layer which contains huge number of data where the image is matched with each data present in fully connected layer. Fully connected layer is also known as SOFTMAX layer. The main building blocks of the modified CNN model is presented in Fig.1. The baseline network is MS-CNN, which detects candidate objects at multiple feature output layers with different scales. To differentiate from the MS-CNN, the proposed enhancements are highlighted by red boxes in Fig.1. The proposed enhancements to MS-CNN are general and are applicable to other CNN models such as Faster-RCNN and SSD as well. As this layer contain huge number of data or stores different number of database the picture in this layer is been matched with each and every data present in the layer . The final outcome is taken by counting the number of matches found to that particular picture or image , as we know that in machine learning the values varies only from 0 - 1 the hence if the matching count is of 0.95 this value will be considered as the final value or the outcome of the image and the category of the image will be identified of type is the image been found. As shown in the architecture each frame contain n number of layers which helps us to get an accurate value for the image and as checks the accuracy of the image given in the input.

The object detection network has a region of interest pooling layer and a fully connected (FC) layer. The outputs of upsampled feature maps from the lowest output feature layer (i.e. “conv4-3”) and object proposals from soft-NMS building block in the proposal networks are used as input to the detection networks. The pooling layer extracts the feature maps of the object proposals using these inputs. The feature maps are upsampled twice to improve the capacity for location-aware bounding box regression. Then a fully connected layer maps the feature maps into fixed vectors for classification and bounding box regression.



. Fig2. Feature fusion method for deconvolution building block (DBB).

CNN exploits multi-scale features to produce predictions of different scales, which showed improved object detection performance over Faster-CNN[7] and SSD[6] for KITTI datasets[1]. It is a good idea to use the feature maps at larger scales (lower CNN layers) with smaller receptive fields to detect smaller objects and those in smaller scales (higher layers) to detect larger objects. However, shallow feature maps from the low layers of feature pyramid inherently lack fine semantic information for object recognition. There is an opportunity to augment the shallow feature maps with deeper feature maps from higher feature output layers and improve detection performance.

We propose to add DBB to the baseline MS-CNN model, with additional deconvolution layers and lateral connections to aggregate feature outputs from different layers. Using DBBs the semantics from higher layers can be conveyed into lower layers to increase the representation capacity. There are three DBBs used in the proposed CNN model. Fig.2 illustrates the architecture of the DBB used in this paper, which connects one feature output layer with its adjacent higher layer counterpart. Specifically, we first connect a convolution layer. with 512 1 1 filters to an output feature layer as shown in the Fig. 2. In addition, in the horizontal direction, a deconvolution layer (“Deconv 4 4 512”) with 512 4 4 filters is applied to upsample the corresponding higher-level feature maps. Then the outputs of these two associated feature layers, which have the same spatial size and

depth, are merged by element-wise sum and processed by a ReLU layer to produce a new output feature layer. In order to maintain feature aggregation consistence, the number of channels is set to 512 in all DBBs.

There are many possible architecture designs for DBBs. For example, for a given feature output layer, the output feature maps can be merged with those from both higher layers and lower layers. However the computation complexity and memory requirement can be increased significantly. We examined and compared several alternative DBB architectures, some using element-wise multiplication or concatenation instead of element-wise sum used in this paper, and some adding a batch normalization function block after the convolution and deconvolution layers in the DBB as shown in Fig. 3. However, according to results from extensive experiments, it is found that the implementation shown in Fig. 3 has the best detection performance and low computation complexity. The results demonstrated that the design of DBB is not straightforward and specific consideration should be taken for different baseline CNN models.

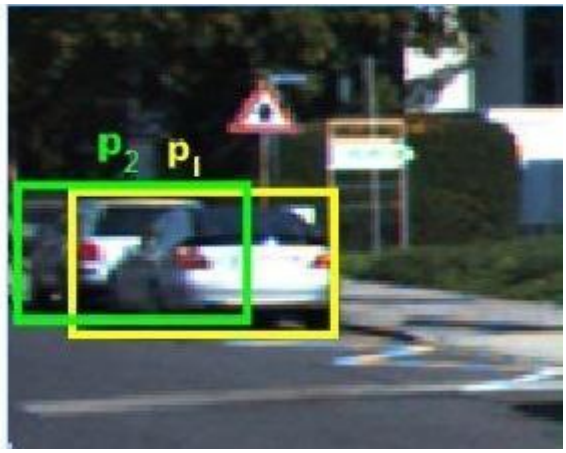


Fig. 3. Example of overlapped proposals.

In many object detection challenge datasets neighbor proposals usually correspond to the same object. But due to heavy object occlusion in KITTI data set, NMS may remove positive proposals unexpectedlyproposals from an image with large overlap inFig.3. The proposal for the occluded back car may be removed with high probability by the traditional NMS method. To address the NMS issue with occluded objects, we apply soft-NMS for suppression of overlapped objects. With soft-NMS the neighbor proposals of a winning proposal are not completely suppressed. Instead they are suppressed according to updated objectiveness scores of the neighbor proposals, which are computed according to the overlap level of the neighbor proposals and the winning proposal. NMS can be viewed as a specific case of soft-NMS, in which the updated objectiveness scores of the neighbor proposals of a winning proposal are simply set to zero.

III. RELATED WORK

the size of the array will be 300x300x3. Where 300 is width, next 300 is height and 3 is RGB channel values. The computer is assigned a value from 0 to 255 to each of these numbers. This value describes the intensity of the pixel at each point, the image is passed through a series of convolutional, nonlinear, pooling layers and fully connected layers, and then generates the output.

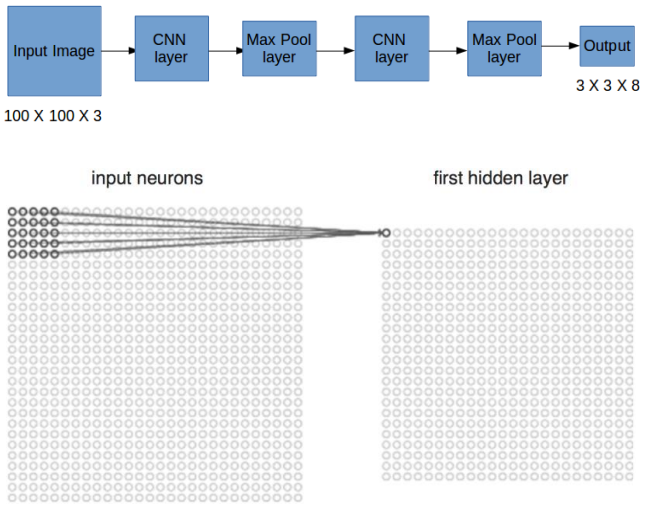


Fig 4. Input Neurons

In recent years, deep learning techniques are achieving state-of-the-art results for object detection, such as on standard benchmark datasets and in computer vision competitions. Notable is the “You Only Look Once,” or YOLO, family of Convolutional Neural Networks that achieve near state-of-the-art results with a single end-to-end model that can perform object detection in real-time.

Yolo-You Only Look OnceAlgorithms based on classification. They are implemented in two stages. First, they select regions of interest in an image. Second, they classify these regions using convolutional neural networks.

The biggest advantage of using YOLO is its superb speed –

it’s incredibly fast and can process 45 frames per second. YOLO also understands generalized object representation.

- YOLO-based Convolutional Neural Network family of models for object detection and the most recent variation called YOLOv3.

- The best-of-breed open source library implementation of the YOLOv3 for the Keras deep learning library.
- How to use a pre-trained YOLOv3 to perform object localization and detection on new photographs

Object detection

To explore the concept of object detection it is useful to begin with image classification. It goes through levels of incremental complexity.

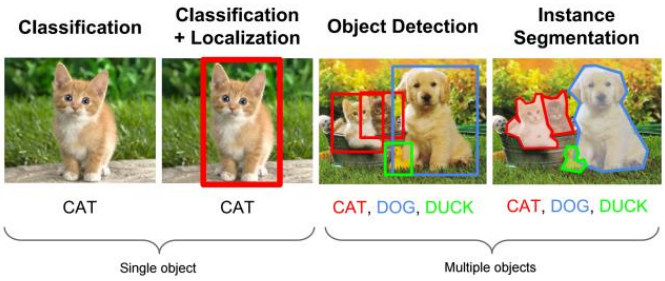


Fig 5. Classification and object detection

Image classification (1) aims at assigning an image to one of a number of different categories (e.g. car, dog, cat, human, etc.), essentially answering the question *"What is in this picture?"*. One image has only one category assigned to it.

Object localization (2) then allows us to locate our object in the image, so our question changes to "What is it and where it is?".
In a real real-life scenario, we need to go beyond locating just one object but rather multiple objects in one image. For example, a **self-driving car** has to find the location of other cars, traffic lights, signs, humans and to take appropriate action based on this information.

Object detection (3) provides the tools for doing just that – finding all the objects in an image and drawing the so-called **bounding boxes** around them. There are also some situations where we want to find exact boundaries of our objects in the process called **instance segmentation**, but this is a topic for another post.

To understand the YOLO algorithm, it is necessary to establish what is actually being predicted. Ultimately, we aim to predict a class of an object and the bounding box specifying object location. Each bounding box can be described using four descriptors:

1. center of a bounding box (**bxby**)
2. width (**bw**)
3. height (**bh**)
4. value **c**is corresponding to a class of an object (such as: car, traffic lights, etc.).

In addition, we have to predict the pc value, which is the probability that there is an object in the bounding box.

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Here,

- p_c defines whether an object is present in the grid or not (it is the probability)
- b_x, b_y, b_h, b_w specify the bounding box if there is an object
- c_1, c_2, c_3 represent the classes. So, if the object is a car, c_2 will be 1 and c_1 & c_3 will be 0, and so on

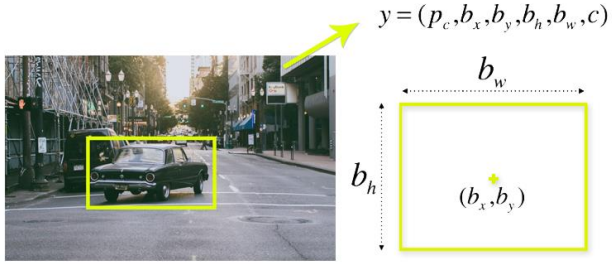


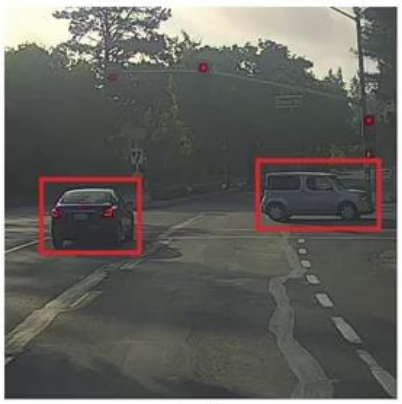
Fig 6 Detection

As we mentioned above, when working with the YOLO algorithm we are not searching for interesting regions in our image that could potentially contain an object.

Instead, we are splitting our image into cells, typically using a 19×19 grid. Each cell is responsible for predicting 5 bounding boxes (in case there is more than one object in this cell). Therefore, we arrive at a large number of 1805 bounding boxes for one image.

How does the YOLO Framework Function?

- YOLO first takes an input image:



- The framework then divides the input image into grids (say a 3 X 3 grid):

- Image classification and localization are applied on each grid. YOLO then predicts the bounding boxes and their corresponding class probabilities for objects

Let's break down each step to get a more granular understanding of what we just learned.We need to pass the labelled data to the model in order to train it. Suppose we have divided the image into a grid of size 3 X 3 and there are a total of 3 classes which we want the objects to be classified into. Let's say the classes are Pedestrian, Car, and Motorcycle respectively. So, for each grid cell, the label y will be an eight dimensional vector:

y =	pc
	bx
	by
	bh
	bw
	c1
	c2
	c3

Here,

- p_c defines whether an object is present in the grid or not (it is the probability)
- b_x, b_y, b_h, b_w specify the bounding box if there is an object
- c_1, c_2, c_3 represent the classes. So, if the object is a car, c_2 will be 1 and c_1 & c_3 will be 0, and so on

Let's say we select the first grid from the above example:

Since there is no object in this grid, pc will be zero and the y label for this grid will be:

y =	0
	?
	?
	?
	?
	?
	?
	?

Here, '?' means that it doesn't matter what b_x, b_y, b_h, b_w, c_1, c_2, and c_3 contain as there is no object in the grid. Let's take another grid in which we have a car (c_2 = 1):

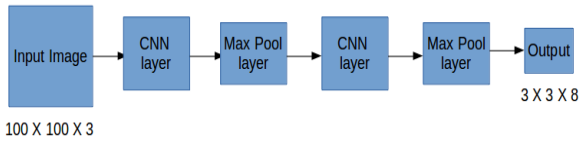
Before we write the y label for this grid, it's important to first understand how YOLO decides whether there actually is an object in the grid. In the above image,

there are two objects (two cars), so YOLO will take the mid-point of these two objects and these objects will be assigned to the grid which contains the mid-point of these objects. The y label for the centre left grid with the car will be:

y =	1
	bx
	by
	bh
	bw
	0
	1
	0

Since there is an object in this grid, p_c will be equal to 1. b_x, b_y, b_h, b_w will be calculated relative to the particular grid cell we are dealing with. Since car is the second class, $c_2 = 1$ and c_1 and $c_3 = 0$. So, for each of the 9 grids, we will have an eight dimensional output vector. This output will have a shape of 3 X 3 X 8.

So now we have an input image and it's corresponding target vector. Using the above example (input image – 100 X 100 X 3, output – 3 X 3 X 8), our model will be trained as follows:



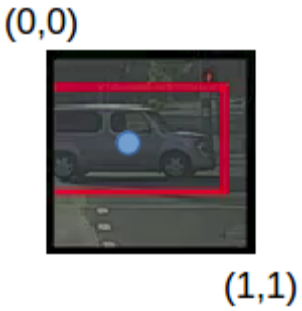
IV. HOW TO ENCODE BOUNDING BOXES?

As I mentioned earlier, b_x, b_y, b_h, and b_w are calculated relative to the grid cell we are dealing with. Let's understand this concept with an example. Consider the center-right grid which contains a car:

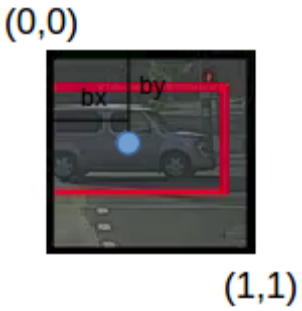
So, b_x, b_y, b_h, and b_w will be calculated relative to this grid only. The y label for this grid will be:

y =	1
	bx
	by
	bh
	bw
	0
	1
	0

$p_c = 1$ since there is an object in this grid and since it is a car, $c_2 = 1$. Now, let's see how to decide b_x, b_y, b_h, and b_w. In YOLO, the coordinates assigned to all the grids are:



b_x, b_y are the x and y coordinates of the midpoint of the object with respect to this grid. In this case, it will be (around) $b_x = 0.4$ and $b_y = 0.3$:



b_h is the ratio of the height of the bounding box (red box in the above example) to the height of the corresponding grid cell, which in our case is around 0.9. So, $b_h = 0.9$. b_w is the ratio of the width of the bounding box to the width of the grid cell. So, $b_w = 0.5$ (approximately). The y label for this grid will be:

y =	1
	0.4
	0.3
	0.9
	0.5
	0
	1
	0

Notice here that b_x and b_y will always range between 0 and 1 as the midpoint will always lie within the grid. Whereas b_h and b_w can be more than 1 in case the dimensions of the bounding box are more than the dimension of the grid.

V. EXPERIMENTS

A. Dataset:

We evaluate the enhanced CNN model over the KITTI 2D object detection benchmark dataset. The dataset contains 300 images with 260 for training and 290 for testing. The image size is 384 1280 pixels. There are over 80000 annotated objects, which are divided into three categories (car, pedestrian and cyclist). Three object detection evaluation categories ("Easy", "Moderate" and "Hard") are set up for each object class, according to object height, occlusion and truncation level, which are presented in Table I. For evaluation, average precision (AP) with different IoU thresholds (0.7 for car, 0.5 for pedestrian and cyclist) is used as the main metric of interest. The AP is computed as the mean precision at a set of equally spaced recall levels.

B. Implementation Details

As a widely adopted practice, the proposed network is fine-tuned on the reduced VGG-16 model, which is pre-trained on the ILSVRC CLS-LOC dataset. We split the raw training dataset into training set and validation set for local performance evaluation. As the number of samples for different object classes are highly imbalanced, detectors are trained separately for detection of cars and pedestrians. The training procedure consists of two stages. In the first stage, only the proposal network is trained by 10000 iterations, with weight term of 0.05, initial learning rate of 0.00005, momentum of 0.9, weight decay of 0.0005. Following the proposal network training, in the second stage the whole network (including both proposal network and detection network) is trained for another 25000 iterations. The learning rate for the second stage is initially set to 0.0005 and is divided by 10 every 10000 iterations. The weight term is the experiments are run with an Intel i7-7700k 4.20GHz server with 8 CPU cores and 32 GB memory and a NVidia GeForce GTX 1080 GPU. Training time ranges from 6 to 10 hours for the models used in this paper.In addition to the various network enhancements, input layer image size impact is also investigated. We train the network with 3 input image sizes, small image 384 1280 (the original image size), medium image 576 1920 and large image 768 2560. The enlargement of images does not increase image resolution. The experiments carried out with different input image size are denoted by the object class and the input image height. For example, experiments for car detection with image size 384 1280 are denoted by "Car-384". Anchor sizes are set differently for different types of experiments. The anchor and associated filter size configurations for different image sizes and different object classes. Note that the other parameters are kept unchanged through all the experiments. we examine and compare the performance of the proposed CNN enhancements for object detection over KITTI benchmark dataset. As the ground truth of the KITTI test set is not publicized and only one submission of the KITTI test results to the benchmark website is allowed, performance comparison of the proposed enhancements is performed over the KITTI training and validation set.

TABLE 1 PERFORMANCE COMPARISON OF RECENT PUBLISHED WORKS AND OUR METHOD ON THE TEST SET.

Method	Pedestrian			Car			Time (s)
	Easy	Mod	Hard	Easy	Mod	Hard	
Faster-RCNN [6]	78.35	65.91	61.19	87.90	79.11	79.19	2
SSD [7]	23.14	16.30	16.06	83.89	67.17	59.09	0.06
YOLOv2 [23]	20.80	16.19	15.43	28.37	19.31	15.94	0.02
spLBP [12]	-	-	-	80.16	77.39	60.59	1.5
Mono3D [32]	77.30	66.66	63.44	90.27	87.86	78.09	4.2
MS-CNN [8]	83.70	73.62	68.28	90.46	88.83	74.76	0.4
Deep3D [33]	-	-	-	90.47	88.86	77.60	1.5
SubCNN [26]	83.17	71.34	66.36	90.75	88.86	79.24	2.0
MV3D [34]	-	-	-	90.53	89.17	80.16	0.36
SDP+RPN [27]	79.98	70.20	64.84	89.90	89.42	78.54	0.4
D_MANTA [35]	-	-	-	97.25	90.03	80.62	0.7
RRC [28]	84.14	75.33	70.39	90.61	90.22	87.44	3.6
Our method	85.12	74.52	69.35	90.49	89.64	77.95	0.24

leader board ranks the approaches based on the AP for "Moderate" detection category, we select the network "M+AR+S" with large image size (768 2560) for competition, which produced the best AP for "Moderate"category over validation set. The results are submitted to the KITTI test set

evaluation server.The AP and inference time results of our proposed method and other top ranked published approaches are presented in Table1. While the original CNN models (Faster-RCNN, SSD and YOLOv2) without adaption to the KITTI datasets have much lower object detection performance over KITTI test set, they are also listed in Table1 for information.A simple comparison of our own results on KITTI test data set to those on validation test shows that there are considerable performance loss possibly due to harder images in the test set. However similar performance loss was observed for the baseline MS-CNN model.According to the object detection results presented in Table and in KITTI benchmark website, it can be observed that the car detection performance for category "Moderate" is almost saturated with very little performance gap over the top 20 detection methods. However, there is still large performance improvement space for pedestrian and cyclist detection. For example the highest AP from the published works is 85.12% and 75.33% for pedestrian category "Easy" and "Moderate", respectively.

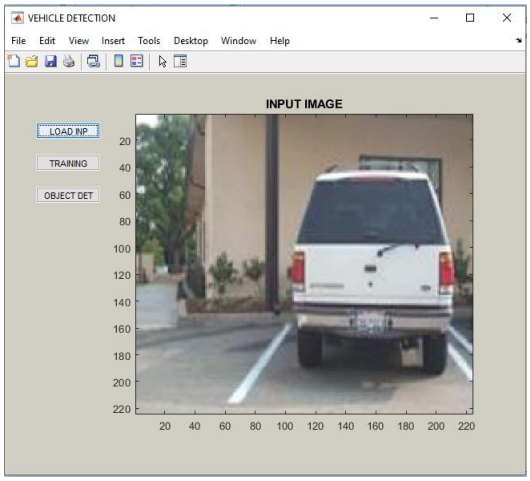


Fig7. Input

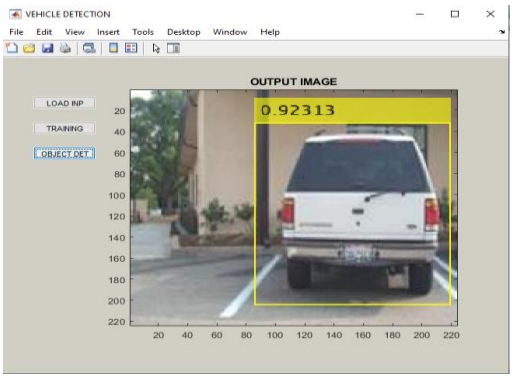


Fig8.Output

The main challenges of the pedestrian and cyclist detection still come from the small size, heavy occlusion or truncation of the objects. In addition other external factors like illumination change and cluttered background can affect the accuracy of our detection method. And compared to the number of car samples in the KITTI dataset, the number of pedestrian and cyclist samples are much smaller, which may be another cause of the relatively poor detection performance for pedestrian detection.

VI. CONCLUSION

Real time accurate object detection is one of the most critical problems for advanced driving assistance systems (ADAS) and autonomous driving. Recently convolutional neural networks (CNN) achieved huge successes on visual object detection over traditional object detectors, which use hand-engineered features. However, due to the challenging driving environment (e.g., large object scale variation, object occlusion and bad light conditions), popular CNN detectors including Faster-RCNN and SSD do not produce good detection performance over the KITTI driving benchmark dataset. Inthis paper we proposed three enhancements on a multiple scale CNN network model for ADAS object detection. Firstly, CNN feature maps deconvolution and fusion was proposed to add context and deeper features for better object detection at lower scale of feature maps, to address the large object scale variation challenge. Then, soft non-maximal suppres-sion (NMS) was applied across object proposals at different image scales to address the object occlusion challenge. As the cars and pedestrians have distinct aspect ratio features, we measured their aspect ratio statistics and exploited them to set anchor boxes properly for better object matching and localization. The proposed CNN enhancements with various input image sizes were individually and jointly evaluated by extensive experiments over KITTI dataset. The effectiveness of the proposed enhancements was verified by experiment results with improved or comparable detection performance over KITTI test set. The average precision (AP) for pedestrian detection category "Easy" and the computation speed rank the first among the published works, the second for pedestrian category "Moderate" and "Hard", the third for car category "Moderate". And the network inference time for cars per 384 1280 image is only 0.08 second, much faster than the other top ranked published methods in KITTI leader board. In our future works we will investigate more CNN models and enhancements to improve object detection for safe and intelligent transport.

REFERENCES

[1] Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pp. 3354–3361, IEEE, 2012.

[2] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisser-man, "The pascal visual object classes (voc) challenge," International journal of computer vision, vol. 88, no. 2, pp. 303–338, 2010.
[3] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar,´ and C. L. Zitnick, "Microsoft coco: Common objects in context," in European conference on computer vision, pp. 740–755, Springer, 2014.
[4] P. Felzenszwalb, R. B. Girshick, and D. McAllester, "Cascade object detection with deformable part models," In CVPR, pp. 2241-2248, 2010.
[5] Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," In NIPS, pp. 1097-1105, 2012.
[6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in Advances in neural information processing systems, pp. 91–99, 2015.
[7] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in European conference on computer vision, pp. 21–37, Springer, 2016.
[8] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, "A unified multi-scale deep convolutional neural network for fast object detection," in European Conference on Computer Vision, pp. 354–370, Springer, 2016.
[9] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, vol. 1, pp. 886–893, IEEE, 2005.
[10] P. Dollar,´ Z. Tu, P. Perona, and S. Belongie, "Integral channel features," 2009.
[11] P. Dollar,´ R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," IEEE Transactions on Pattern Analysis and Ma-chine Intelligence, vol. 36, no. 8, pp. 1532–1545, 2014.
[12] Q. Hu, S. Paisitkriangkrai, C. Shen, A. van den Hengel, and F. Porikli, "Fast detection of multiple objects in traffic scenes with a common detection framework," IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 4, pp. 1002–1014, 2016.
[13] R. N. Rajaram, E. Ohn-Bar, and M. M. Trivedi, "Looking at pedestrians at different scales: A multiresolution approach and evaluations," IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 12, pp. 3565–3576, 2016.
[14] X. Yuan, S. Su, and H. Chen, "A graph-based vehicle proposal location and detection algorithm," IEEE Transactions on Intelligent Transporta-tion Systems, 2017.
[15] R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," International journal of com-puter vision, vol. 104, no. 2, pp. 154–171, 2013.
[16] L. Zitnick and P. Dollar,´ "Edge boxes: Locating object proposals from edges," in European Conference on Computer Vision, pp. 391– 405, Springer, 2014.