YOLO-BASED METHOD FOR INDIAN CAR LICENSE PLATE DETECTION

[1]MONAL PATEL, [2]ARVIND YADAV, [3]CARLOS VALDERRAMA

[1,2]Dept. of Electronics and Communication Parul Institute of Engineering and Technology, Parul University, Vadodara, Gujarat, India
[3]Dept. of Electronics and Microelectronics University of Mons, Mons, Belgium, Europe
E-mail: [1]monalrakeshpatel@gmail.com, [2]arvind.yadav@paruluniversity.ac.in, [3]cavs63@gmail.com

Abstract - This paper presents a novel convolutional neural network (CNN) YOLO-based technique for high-accuracy real time Indian car license plate detection. Several methods for car license plate detection are reasonably effective under the specific conditions or strong assumptions only. However, they exhibit poor performance once the assessed car license plate images have a degree of rotation, as a result of manual capture by traffic police or deviation of the camera. Therefore, in this work we have proposed CNN-based YOLO framework for Indian car license plate detection. Using accurate prediction and a fast intersection-over-union evaluation strategy, our proposed method can elegantly manage problems in real time scenarios. A series of experiments have been carried out to establish that the proposed method outperforms other existing state-of-the-art methods in terms of higher accuracy and lower computational cost.

Keywords - License Plate Detection, Convolutional Neural Network, YOLO, Multi-Direction Indian License Plate.

I. INTRODUCTION

In recent years, the number of privately-owned cars has increased considerably and this has, in turn, exacerbated the traffic management burden. The resultant congestion can cause extreme issues, like traffic accidents or public area vulnerability to crime or terrorist attacks. Physical management of this ubiquity of automobiles is quite challenging, and this problem has encouraged the development of automatic systems to manage traffic jams. Specially, automatic license plate detection can adequately monitor the cars and greatly alleviate traffic management burdens; therefore, this approach has attracted considerable attention of researchers [14], [24]. Further, automatic car license plate detection can also be used in several different scenarios, e.g., for expressway toll collection, over speeding violations [21], surveillance and management of unattended parking lots, etc. Both traditional and convolutional neural network (CNN) based methods are being used to solve the problem of car license plate detection. The traditional methods involve hand-crafted features such as color [1], edge and morphology [16], which are primarily confined by stringent conditions. For example, a number of these systems need high resolution images as input; acquisition of which requires expensive equipment, while others need strict device mounting free from translation and rotation. However, real-world scenarios are quite different where car license plate detection becomes very challenging. Different kinds of cars and roads, changing weather conditions, camera device rotation, and so on curtail the detection performance significantly. Thus, under the complex situation, it is relatively difficult to propose a robust method with hand-crafted features. Although people can employ multiple independent features and incorporate some models together, as mentioned in, it is still hard to distinguish whether it is enough to satisfy the challenge with such limited features and models. To alleviate these problems, CNN-based strategies [18] have been devised, which automatically learn features from the acquired data. These kinds of detection methods have recently yielded very impressive results [4]; Further, despite the viability of both traditional and CNN-based strategies, the issues related with car license plate detection have not yet been satisfactorily resolved to date, because of the difficulties due to the viewpoint variation of hand-held cameras or the accidental rotation of mounted cameras. Inspired by the "you only look once" (YOLO) [28] framework, we propose a CNN-based strategies that can manage the problem reasonably well. The main contributions of our work are summarized as follows:

Proposed a novel accurate prediction technique to comprehend Indian car license plate detection; To further increase the detection accuracy; we devised a pre positive CNN model that is implemented before YOLO; which serves to determine the "attention region" in the overall framework. The method of cascading the two models is based on prior knowledge: as the car license plates are fixed on the cars, some distance will inevitably exist between any two plates. The synergy of this idea is explained in the ulterior section; Rest of the manuscript is structured as follows. Section II gives a brief review of car license plate detection. In Section III, the proposed method is described in detail. Section IV presents a series of experimental results and analyses. Finally, a concise conclusion summarizing the main points of the entire work is rendered in Section V.

II. RELATED WORK

Over the last two decades, impressive research work has been carried out by the computer vision community to address the problem of automatic car license plate detection and recognition [8]. The task of license plate detection can broadly be classified into following three approaches;

1) Region-based approaches
2) Pixel-to-Pixel approaches [23], and
3) Color based approaches.

In region-based technique, input image is segmented into smaller regions, where some pre specified attributes of the license plate are situated in such subsequent regions. Various unified approaches, based on morphological and high-pass filtering, are devised to effectively perform car license plate detection task in [11], [22], [26]. In pixel-to-pixel approach, every pixel is evaluated within the image among its neighboring pixels to form a coarse rectangular box. The whole input image is scanned pixel-by-pixel using a detection window. The response of scanning window is computed at each pixel location and the regions with high response to scanning window are selected as candidate region.

Dlagnekov (2004) has formulated an Adaboost classifier for pixel-to-pixel classification of an image [6]. In [10], Ho et al. addressed the problem of license plate detection using pixel-to-pixel method. The authors contributed by devising the Adaboost classifier at first step and then Scale-invariant Feature Transform (SIFT)-based SVM classifier in the later stage. The SIFT based approach was also exercised by Silva for license plate detection and recognition [3]. In, HOG-features of the segmented license plate were extracted and then a Gentle AdaBoost trained classifier was used for license plate detection. Subsequently, a novel color based approach was developed by Azad et al. [2], by converting the RGB-color images to the HSV-color space. Then, this HSV-image is segmented into smaller blocks, wherein each block is examined for license plate or some part of license plate by using carefully formulated filtering procedure. Deb et al. in [5], proposed three step procedures for car license plate detection and recognition; where first, I) sliding concentric windows (SCWs) based method is used for detection of candidate regions, II) HSI-color model based verification was rendered to identify candidate regions, and finally III) candidate regions were decomposed using position histogram to segment out alphanumeric characters in the plate. Over the couple of last few years, many computer vision problems are being addressed by using deep convolutional neural networks (CNNs); in line with that, many researchers exercised to solve the problems of car license plate detection and recognition using CNNs [18], [27]. CNNs have already exhibited remarkable performance for text and optical character recognition [13], [30], [38] and object detection [14], [20], [28], [29]; whereas, the ancillary task of license plate detection and localization is underway to be better addressed by using deep CNNs; especially in the challenging situation of multi-directional car plate detection, current state of-the-art CNN based models may not be directly used to achieve good performance. Although the problem of multi-directional car license plate detection was aware and has been addressed by some researchers (e.g. [12]), but the performance using traditional method [12] is still far away from practical promising. Recently, researches of computer vision tasks in transportation system have made significant progresses. Under high demand of robustness, some new methods tend to employ the features extracted by CNN instead of hand-crafted features. Besides, it is necessary to enrich the dataset with certain data augmentation strategies. The work of, balancing the proportion of different objects in dataset with hierarchical augmentation, has obviously improved the accuracy. Furthermore, many researchers have tried utilizing the prior knowledge e.g. location, symmetry to improve the performance. Our proposed method in this paper also follows these trends to achieve high detection accuracy.

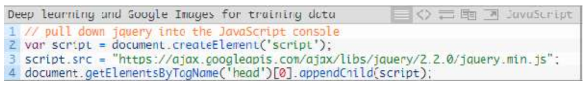
III. METHODOLOGY

Dataset Collection

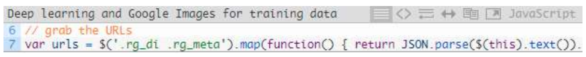
The license plate Dataset for training (image, video) has been collected from [31]. One needs to be aware of what problem domain is ...For example, the wide camera angle shot for Indian license plate, the license plate is very tiny. However, the license plate size gets bigger with the close-up camera shot. These two different types of photo mean the different domains. In our case, we have collected images of only close-up camera shot. Also, the collected images are Only Indian License plate images.

Create a deep learning dataset using Google Images

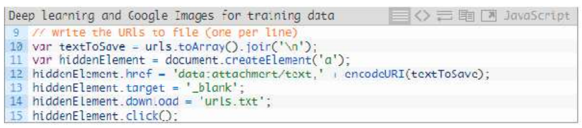
We have search results for Indian License plate. The next step was to use a tiny bit of JavaScript to gather the image URLs. Fire up the JavaScript console (We have made use of Chrome web browser)
clicking View => Developer => JavaScript Console. Keep scrolling until you have found all relevant images to your query. From there, we grab the URLs for each of these images. Switch back to the JavaScript console and then copy and paste this JavaScript snippet into the Console.


Deep learning and Google Images for training data — JavaScript

```javascript
// pull down jquery into the JavaScript console
var script = document.createElement('script');
script.src = "https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js";
document.getElementsByTagName('head')[0].appendChild(script);
```


Deep learning and Google Images for training data — JavaScript

```javascript
// grab the URLs
var urls = $('.rg_di .rg_meta').map(function() { return JSON.parse($(this).text()).
```


Deep learning and Google Images for training data — JavaScript

```javascript
// write the URLs to file (one per line)
var textToSave = urls.toArray().join('\n');
var hiddenElement = document.createElement('a');
hiddenElement.href = 'data:attachment/text,' + encodeURI(textToSave);
hiddenElement.target = '_blank';
hiddenElement.download = 'urls.txt';
hiddenElement.click();
```

After executing the above snippet a file named urls.txt is generated in default downloads directory. The LabelImg software (a graphical image annotation tool) has been used to make annotations to each image, as illustrated in Figure 1. It is written in Python and uses Qt for its graphical interface. Annotations are saved as XML files in PASCAL VOC format, the format used by ImageNet, and it also supports YOLO format.

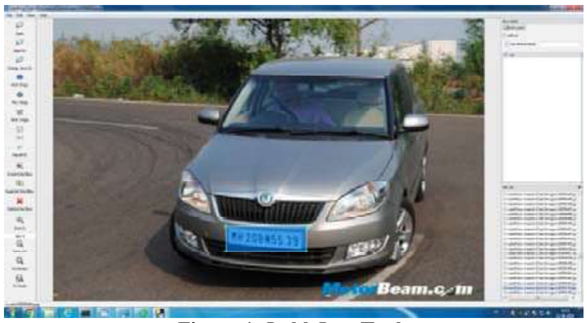


Figure 1: LableImg Tool

Making Own Configuration File (cfg)

A slight change in the configuration file (*.cfg) is required to do in order to adjust existing model to fit into own dataset. For example, YOLOv2's CNN model is trained on ImageNet dataset which contains a number of different object classes. However, we only wish to detect Indian License Plate object class.

- Copy one of the pre-trained configuration file that is to be used, so we have used yolo.cfg.
- Rename the copied configuration file to custom name. Thus we have changed the name yolo.cfg to yolo_custom.cfg.
- Open the configuration file and change classes in the [region] layer (the last layer) to the number of classes required to train for. In our case, classes are set to 1(one).
- Change filters in the [convolutional] layer (the second to last layer) to num * (classes + 5). Thus for our case, num is 5 and classes are 1 so 5 * (1 + 5) = 30 therefore filters are set to 30.

Specifying Classes of Your Interest

The 'labels.txt' file included in the root directory need to be changed. Just specify names of objects. We have removed all text and left 'License Plate' instead.

Downloading Pre-trained Weights

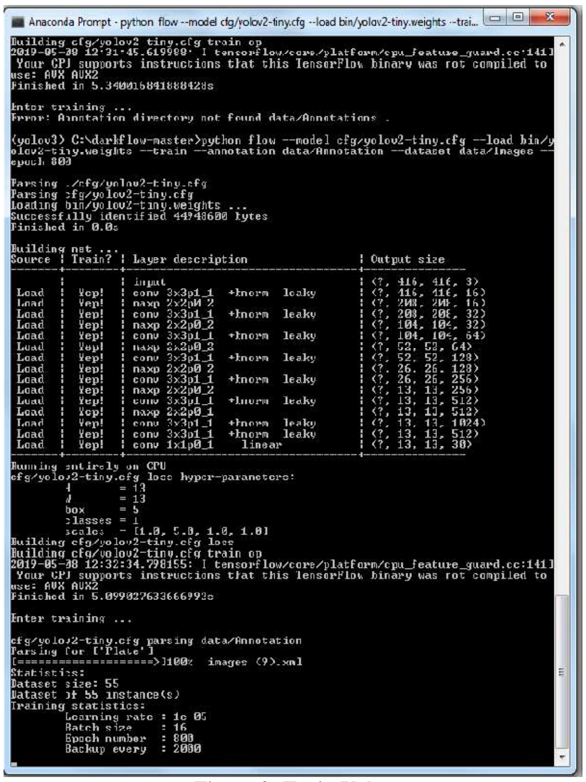
Download pre-trained weights that you are interested. We have downloaded yolo.weights. After downloading is done, locate the file into 'bin' directory in the root directory.

Defining Model Options & Build the Model

model	Configuration file (*.cfg) that you have defined for your situation.
load	Pre-trained weight file.
batch	A number of data to train per batch. We set batch size 8 because our GPU card (NVIDIA GTX 1080Ti) couldn't handle bigger size.
epoch	How much iteration to train. This need to set a bit smaller, In our experiment 100 epochs is giving good result.
gpu	Set 1.0, to fully utilize the GPU hardware. Otherwise, remove this option.
train	Set 'True', if your purpose is to train.
annotation	Directory where the annotation files are stored.
dataset	Directory where the image files are stored.

Training the Model

In this experiment, A desktop PC, having Intel i7 Processor, 16-GB RAM, and NVidia 8-GB Graphics card has been used and it took a day (24 hours) for Training the model. Figure 2, depicts the Training model for YOLO.



```
Anaconda Prompt - python flow --model cfg/yolov2-tiny.cfg --load bin/yolov2-tiny.weights --trai...
Building cfg/yolov2 tiny.cfg train op
2019-05-08 12:31:45.619988: I tensorflow/core/platform/cpu_feature_guard.cc:141]
 Your CPU supports instructions that this TensorFlow binary was not compiled to
use: AVX AVX2
Finished in 5.340016841888428s

Enter training ...
Error: Annotation directory not found data/Annotations .

(yolov3) C:\darkflow-master>python flow --model cfg/yolov2-tiny.cfg --load bin/y
olov2-tiny.weights --train --annotation data/Annotation --dataset data/Images --
epoch 800

Parsing ./cfg/yolov2-tiny.cfg
Parsing cfg/yolov2-tiny.cfg
Loading bin/yolov2-tiny.weights ...
Successfully identified 44948600 bytes
Finished in 0.0s

Building net ...
Source | Train? | Layer description                | Output size
-------+--------+----------------------------------+---------------
       |        | input                            | (?, 416, 416, 3)
 Load  |  Yep!  | conv 3x3p1_1  +bnorm  leaky      | (?, 416, 416, 16)
 Load  |  Yep!  | maxp 2x2p0_2                     | (?, 208, 208, 16)
 Load  |  Yep!  | conv 3x3p1_1  +bnorm  leaky      | (?, 208, 208, 32)
 Load  |  Yep!  | maxp 2x2p0_2                     | (?, 104, 104, 32)
 Load  |  Yep!  | conv 3x3p1_1  +bnorm  leaky      | (?, 104, 104, 64)
 Load  |  Yep!  | maxp 2x2p0_2                     | (?, 52, 52, 64)
 Load  |  Yep!  | conv 3x3p1_1  +bnorm  leaky      | (?, 52, 52, 128)
 Load  |  Yep!  | maxp 2x2p0_2                     | (?, 26, 26, 128)
 Load  |  Yep!  | conv 3x3p1_1  +bnorm  leaky      | (?, 26, 26, 256)
 Load  |  Yep!  | maxp 2x2p0_2                     | (?, 13, 13, 256)
 Load  |  Yep!  | conv 3x3p1_1  +bnorm  leaky      | (?, 13, 13, 512)
 Load  |  Yep!  | maxp 2x2p0_1                     | (?, 13, 13, 512)
 Load  |  Yep!  | conv 3x3p1_1  +bnorm  leaky      | (?, 13, 13, 1024)
 Load  |  Yep!  | conv 3x3p1_1  +bnorm  leaky      | (?, 13, 13, 512)
 Load  |  Yep!  | conv 1x1p0_1    linear           | (?, 13, 13, 30)
-------+--------+----------------------------------+---------------
Running entirely on CPU
cfg/yolov2-tiny.cfg loss hyper-parameters:
        H       = 13
        W       = 13
        box     = 5
        classes = 1
        scales  = [1.0, 5.0, 1.0, 1.0]
Building cfg/yolov2-tiny.cfg loss
Building cfg/yolov2-tiny.cfg train op
2019-05-08 12:32:34.798155: I tensorflow/core/platform/cpu_feature_guard.cc:141]
 Your CPU supports instructions that this TensorFlow binary was not compiled to
use: AVX AVX2
Finished in 5.099027633666992s

Enter training ...

cfg/yolov2-tiny.cfg parsing data/Annotation
Parsing for ['Plate']
[====================>]100%  images (9).xml
Statistics:
Dataset size: 55
Dataset of 55 instance(s)
Training statistics:
        Learning rate : 1e-05
        Batch size    : 16
        Epoch number  : 800
        Backup every  : 2000
```

Figure 2: Train Yolo

Loading from the Latest Checkpoint
How to load custom trained model from the checkpoint. It wills easy your work.

Model: use the configuration model (*.cfg)

Load: Set -1. In this way, the latest checkpoint will be loaded.

Predicting on an Image Results of predict Indian License Plate is given in experiments section.

IV. RESULTS

In this experimentation work, 1000 images have been used for validation. The moving average loss for validation was found to be 0.72 and IoU was more than 85% per class. In license plate detection, we have achieved 72.7% accuracy on test data from our collected dataset [31]. Some of the license plate detection images are shown in Figure 3 for illustration purpose.

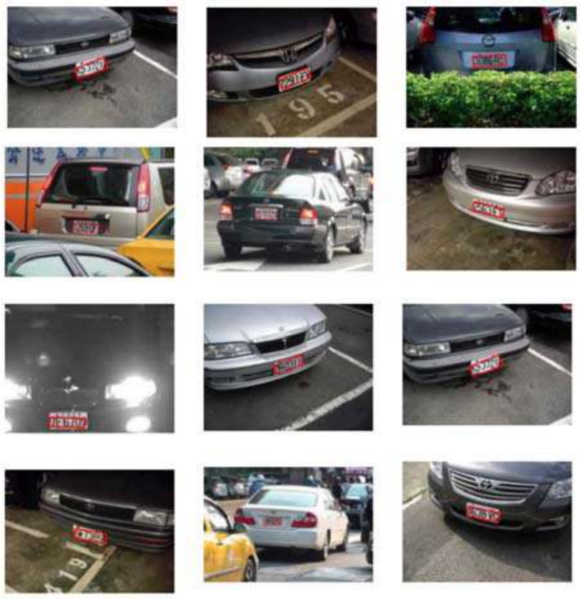


Figure 3. Examples of car license plate detection

V. CONCLUSIONS

In this paper, we have introduced a new YOLO model for Indian car license plate detection. The proposed model can elegantly solve the problem of Indian car license plate detection, and can also be deployed easily in real-time circumstances, because of its reduced computational complexity compared with previous CNN based methods. With the introduction of an attention-like pre positive CNN model, the YOLO framework yields new state-of the-art accuracy. Moreover, the Indian car license plate detection method presented in this paper can handle challenging real-world scenarios reasonably well, even in cases when only limited computational resources, such as CPUs, are available. However, there are still a couple of issues for robust Indian license plate detection, which need to be addressed in future. Firstly, the lack of availability of enough data of Indian car license plates limits the performance of our method, which will force us to collect more data or employ some advanced techniques to synthesize data. Besides, although we focus on the Indian car license plate detection in this paper, the detection still aims for the subsequent recognition. Therefore, it deserves to further explore whether we can propose an end-to-end model to simultaneously deal with detection and recognition. Last but not least, the tasks of Indian car license plate detection under poor conditions such as low resolution, terrible illumination, and accidental occlusion and so on remain great challenges waiting to be solved.

REFERENCES

[1] A. H. Ashtari, M. J. Nordin, and M. Fathy, “An iranian license plate recognition system based on color features,” IEEE Trans. Intell. Transp. Syst., vol. 15, no. 4, pp. 1690–1705, 2014.

[2] R. Azad, F. Davami, and B. Azad, “A novel and robust method for automatic license plate recognition system based on pattern recognition,” Adv. Comput. Sci., Int. J., vol. 2, no. 3, pp. 64–70, 2013.

[3] F. A. da Silva, A. O. Artero, M. S. V. de Paiva, and R. L. Barbosa, “ALPRS-A new approach for license plate recognition using the sift algorithm. Signal & Image Processing : An International Journal (SIPIJ), vol.4, No.1, pp. 17-33, 2013.

[4] J. Dai, Y. Li, K. He, and J. Sun, “R-FCN: Object detection via region based fully convolutional networks,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 379–387.

[5] K. Deb, H.-U. Chae, and K.-H. Jo, “Vehicle license plate detection method based on sliding concentric windows and histogram,” JCP, vol. 4, no. 8, pp. 771–777, 2009.

[6] L. Dlagnekov, “License plate detection using Adaboost,” Dept. Comput. Sci. Eng., San Diego, CA, USA, Tech. Rep., 2004, pp. 1-6. [Online] Available: https://cseweb.ucsd.edu/classes/fa04/cse252c/projects/louka.pdf.

[7] P. Dollár, R. Appel, S. Belongie, and P. Perona, “Fast feature pyramids for object detection,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 36, no. 8, pp. 1532–1545, 2014.

[8] S. Du, M. Ibrahim, M. Shehata, and W. Badawy, “Automatic license plate recognition (ALPR): A state-of-the-art review,” IEEE Trans. Circuits Syst. Video Technol., vol. 23, no. 2, pp. 311–325, 2013.

[9] R. Girshick, “Fast R-CNN,” in Proc. Int. Conf. Comput. Vis. (ICCV), 2015, pp. 1440–1448.

[10] W. T. Ho, H. W. Lim, and Y. H. Tay, “Two-stage license plate detection using gentle adaboost and SIFT-SVM,” in Proc. 1st Asian Conf. Intell. Inf. Database Syst. (ACIIDS), Apr. 2009, pp. 109–114.

[11] B. Hongliang and L. Changping, “A hybrid license plate extraction method based on edge statistics and morphology,” in Proc. 17th Int. Conf. Pattern Recognit. (ICPR), vol. 2. 2004, pp. 831–834.

[12] G.-S. Hsu, J.-C. Chen, and Y.-Z. Chung, “Application-oriented license plate recognition,” IEEE Trans. Veh. Technol., vol. 62, no. 2, pp. 552–561, 2013.

[13] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, “Reading text in the wild with convolutional neural networks,” Int. J. Comput. Vis., vol. 116, no. 1, pp. 1–20, 2016.

[14] H. Karwal and A. Girdhar, “Vehicle number plate detection system for indian vehicles,” in Proc. IEEE Int. Conf. Comput. Intell. Commun. Technol. (CICT), Feb. 2015, pp. 8–12.

[15] K.-H. Kim, S. Hong, B. Roh, Y. Cheon, and M. Park. (2016). “PVANET: Deep but lightweight neural networks for real-

time object detection." pp. 1-7, 2016 [Online]. Available: https://arxiv.org/abs/1608.08021.

[16] H. Li and C. Shen, "Reading car license plates using deep convolutional neural networks and LSTMS," CoRR, Jan. 2016, pp. 1-17. [Online]. Available: https://arxiv.org/abs/1601.05610

[17] W. Liu et al., "SSD: Single shot multibox detector," in Proc. Eur. Conf. Comput. Vis. (ECCV), 2016, pp. 21–37.

[18] D. C. Luvizon, B. T. Nassu, and R. Minetto, "Vehicle speed estimation by license plate detection and tracking," in Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP), May 2014, pp. 6563–6567.

[19] F. Martín, O. Martín, M. García, and J. L. Alba, "New methods for automatic reading of VLP's (vehicle license plates)," in Proc. IASTED Int. Conf. SPPRA, 2002, pp. 126–131.

[20] J. Muhammad and H. Altun, "Improved license plate detection using HOG-based features and genetic algorithm," in Proc. 24th Signal Process. Commun. Appl. Conf. (SIU), May 2016, pp. 1269–1272.

[21] C. Patel, A. Patel, and D. Shah, "A novel approach for detecting number plate based on overlapping window and region clustering for Indian conditions," Int. J. Image, Graph. Signal Process., vol. 7, no. 5, pp. 58–65, 2015.

[22] Q. Wang, J. Gao, and Y. Yuan, "A joint convolutional neural networks and context transfer for street scenes labeling," IEEE Trans. Intell. Transp. Syst., vol. 19 no.5, pp. 1457-1470, 2017.

[23] Q. Wang, Y. Yuan, P. Yan, and X. Li, "Saliency detection by multiple instance learning," IEEE Trans. Cybern., vol. 43, no. 2, pp. 660–672, 2013.

[24] Q. Wang, G. Zhu, and Y. Yuan, "Multi-spectral dataset and its application in saliency detection," Comput. Vis. Image Underst., vol. 117, no. 12, pp. 1748–1754, 2013.

[25] R. Wang, N. Sang, R. Wang, and L. Jiang, "Detection and tracking strategy for license plate detection in video," Opt.-Int. J. Light Electron Opt., vol. 125, no. 10, pp. 2283–2288, 2014.

[26] S.-Z. Wang and H.-J. Lee, "Detection and recognition of license plate characters with different appearances," in Proc. IEEE Intell. Transp. Syst., vol. 2. Oct. 2003, pp. 979–984.

[27] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng, "End-to-end text recognition with convolutional neural networks," in Proc. 21st Int. Conf. Pattern Recognit. (ICPR), 2012, pp. 3304–3308.

[28] D. Wazalwar et al., "A design flow for robust license plate localization and recognition in complex scenes," J. Transp. Technol., vol. 2, no. 1, p. 13, 2012.

[29] M. S. Yee and L. Hanzo, "Radial basis function decision feedback equaliser assisted burst-by-burst adaptive modulation," in Proc. Global Telecommun. Conf., vol. 4. 1999, pp. 2183–2187.

[30] K. M. A. Yousef, M. Al-Tabanjah, E. Hudaib, and M. Ikrai, "Sift based automatic number plate recognition," in Proc. 6th Int. Conf. Inf. Commun. Syst. (ICICS), Apr. 2015, pp. 124–129.

[31] https://www.google.com/search?q=indian+licence+plate&source=lnms&tbm=isch&sa=X&ved=0ahUKEwib9LfVq7zjAhWq6XMBHfUNBEIQ_AUIECgB&biw=1366&bih=625

★★★