International Journal of Scientific Research and Engineering Development-– Volume X Issue X, Year

Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

CLASSIFICATION AND DETECTION OF TRAFFIC SIGN USING DEEP LEARNING

Mansiben Nareshbhai Rana
(Computer Engineering [Software Engineering], Gujarat Technological University, Ahmedabad / Ipcowala Institute of Engineering & Technology, Dharmaj
Email: mnrana1998@gmail.com
Contact No. : +91 8320053394)
Supported By : Rushikesh Chaudhari, Kajal Patel, Premal J. Patel, Vaibhav J. Rana

----------**********----------

Abstract:

The route is the basic system that takes passengers to their destination. Since when we go to a familiar place to cruise in our car, we can take the road signs lightly. But, if we have to travel somewhere new, suddenly traffic signals become our best friends. It is hard to imagine a world without traffic signs. They weren't always there, because in the past the traffic was not the same as it is today. In today's technological age, road signs are installed at specific places to ensure the safety of passengers. These signs also guide when and where drivers may or may not turn. We have proposed a system for traffic sign detection and classification. Nowadays people are victims of accidents, one of the reasons is that people do not recognize the traffic sign boards, or do not follow them. That's why I want to create an automatic traffic sign recognizer that can inform vehicles and drivers about oncoming traffic signals and obey them. This can reduce road accidents. Convolution Neural Network is part of deep learning and is widely used in image recognition.

Keywords — **You Only Look Once (YOLO), Convolution Neural Network (CNN), Traffic Sign Detection, Traffic Sign Classification**

----------**********----------

1. INTRODUCTION

Technology is growing very fast nowadays. And we are in touch with various new technologies on a daily basis. Artificial intelligence technology is widely used nowadays in computer science. Artificial intelligence is ready to build an intelligent machine. Artificial intelligence is currently working with various subfields. Such as self-driving car, proving the theorem, playing chess, playing music, drawing etc.

Self-driving vehicles are cars or trucks in which human drivers are never required to take control to safely drive or operate the vehicle. “Autonomous” or “driverless” cars combine sensors and software to control, navigate and drive the vehicles. Self driving car performing many automated tasks. But according to my topic, self driving car detects and classify traffic signs. And here classification and detection operation are performing using deep learning technique. Now the classification is the process of categorizing things on the basis of properties. Detection is the act of discovering something. Traffic signs / road signs are a sign near the road that has information and suggestion for drivers and deep learning is the machine learning technique that teaches computers to do what comes naturally to humans. This way my topic says that using deep learning techniques my system will

detect the traffic sign and classify according their properties. Properties like their colour and shape.

Traffic sign recognization is an important task in self driving cars. Traffic sign recognization has two stages : first is in real traffic scene, find the location of the traffic sign & the second is classify detected traffic sign into their particular group / subgroup. Group / subgroup like regulatory sign, warning sign, guide sign, services sign, construction sign, recreation sign, school zone sign, incident management sign, etc.

(1.1) OVERVIEW OF TRAFFIC SIGN DETECTION AND CLASSIFICATION BASED ON DEEP LEARNING

Traffic signs play an important role in road transportation systems because they provide useful and vital road information and instruction to road users. For example, to regulate the traffic safety, variable speed limitations, informational signs, and directional signs are placed along the road according to the environmental conditions and traffic situations of the road. Therefore, rapidly updating traffic signs is essential for transportation agencies to manage and monitor the status and usability of traffic signs. Current traffic sign detection and recognition systems are based mainly on digital images and videos. To warn and guide drivers, traffic signs, well defined by highly contrasting colours (e.g., red, blue, yellow, and white), can be distinguished easily from a complex environment. Additionally, transportation systems mainly use circular, triangular, rectangular, square, octagonal, and pentagonal shapes of traffic signs to regulate the traffic. The combination of shapes and colours represents traffic-sign categories, such as prohibition, danger, obligation, and warning. Therefore, based on image features derived from colour and shape information, traffic signs are usually described by various feature descriptors, including the following: scale-invariant feature transform, wavelet decomposition, and histograms of oriented gradients, local binary pattern, and correlating Fourier descriptors.

Image- or video-based traffic-sign detection and recognition systems generally have been developed by applying the features to a variety of classifiers and object-classification algorithms, such as the following: support vector machine, random forest, neural network, convolutional neural network (CNN), decision fusion and reasoning module, and template-matching-based methods. However, these image- or video-based systems suffer from the following limitations: 1) weather conditions (e.g., fog and rain), affecting the visibility of traffic signs, 2) shadows, caused by other adjacent objects or different illumination levels, 3) light condition, and 4) sign condition, including colour fading, deformation, and similarity of signs. The traffic sign detection task has proven to be effective and reliable.

Figure : 1.2 : Detected and classified traffic sign

In this (Figure : 1.2) image, traffic sign classification and detection operations are performed. Here we can see that one traffic sign is detected and classified. Here we can see Construction traffic sign image with their particular labeling. The square around the traffic sign is bounding box. This bounding box gives the location of the particular traffic sign and detects the traffic sign. And the classification methods are classified particular traffic sign with their labeling using classification methods.

(1.2) Deep Learning Introduction:

Deep learning can be considered as a subset of machine learning. It is a field that is based on learning and improving on its own by examining computer algorithms. While machine learning uses simpler concepts, deep learning works with artificial neural networks, which are designed to imitate how humans think and learn. Until recently, neural networks were limited by computing power and thus were limited in complexity. However, advancements in Big Data analytics have permitted larger, sophisticated neural networks, allowing computers to observe, learn, and react to complex situations faster than humans. Deep learning has aided image classification, language translation, speech recognition. It can be used to solve any pattern recognition problem and without human intervention.

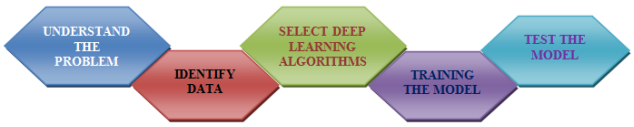

Figure: 1.3 : Deep Learning Process

(1.3) OBJECTIVES

- One of the main objectives is to create automatic traffic sign detection and classification for India.
- The purpose of traffic sign detection is to find the locations and sizes of traffic signs in landscape images.
- The colors and shapes are the two main components for traffic sign detection.
- Thus, we can divide the detection methods into two categories: color-based and shape-based.
- The purpose of traffic sign classification is to categorize detected traffic signs into their specific sub-categories.
- Traffic sign classification is the process of automatically recognizing traffic signs along the road, including speed limit signs, yield signs, merge signs, etc.
- The system should be able to detect traffic signs in the image independently of their appearance. Because of that, it should be mandatory for: Perspective distortion, Lighting changes, Partial barriers and Shadows.
- In addition, it provides information about the presence of potential problems: Lack of visibility, Bad condition and Bad placement.
- Being able to automatically recognize traffic signs enables us to build a "smart car".

(1.4) MOTIVATION

- Motivation is the process that initiates, guides and maintains goal-oriented behaviors.
- In everyday usages, the term “MOTIVATION” is used to describe “why a person does something?”
- Nowadays, Automatic Systems are trending.
- For example, Automatic systems are used in HOME like smart lighting, smart lock, smart security, etc.
- Automatic systems are often referred as “AUTOMATION”.
- “AUTOMATION” is a technique that is used to complete tasks with minimal or no human intervention or interfere.
- This technique is used to make devices, systems or processes to operate “AUTOMATICALLY”.
- AI is the part of Automation.
- And AUTOMATION is the new trend, from mobile phones to driverless cars.
- And the ROUTE is the basic system that takes passengers to their destination.
- When we go to a familiar place, we can take the road signs lightly.
- But if we want to travel somewhere new, Suddenly Traffic Sign becomes our best friends.
- It is hard to imagine a world without Traffic Sign.
- The primary goal is to detect boards that are assigned with Traffic signs.
- That’s why I want to create an automatic traffic sign recognizer that can inform vehicles and drivers about oncoming traffic sign and obey them.

2. WRITING STUDY [LITERATURE SURVEY]

PAPER-1 : Traffic Sign Classification And Detection Of Indian Traffic Signs Using Deep Learning

Publication Year : 2021

Author : Manjiri Bichkar, Suyasha Bobhate, Prof. Sonal Chaudhari

Journal Name: International Journal of Scientific Research in Computer Science, Engineering and Information Technology

Summary:

- In this paper the classification of traffic signs using CNN with different filters were helped to improve accuracy and trained using GTRSB dataset and tested with Indian dataset. The detection model used YOLO and BLOB analysis to detect traffic signs from environment and classified according to their classes. Transfer learning was used to classify and detect Indian Traffic Signs specifically.

PAPER-2 **:** Real Time Detection and Classification of Traffic Signs Based On Yolo Version 3 Algorithm

Publication Year : 2020

Author : Valentyn Sichkar , Sergey A. Kolyubin

Journal Name : Scientific and Technical Journal of Information Technologies Mechanics and Optics

Summary :

- This paper presents the recognition problem for a variety of traffic sign classes. As for detection, deep convolutional YOLO version 3 model was trained on GTSDB to predict locations of traffic signs among 4 categories. Further, one-layer convolutional model, trained on GTSRB, was stacked to utilize final classification among one of the 43 classes. Experiments showed that training of deep network with only 4 categories to detect traffic signs gives high mAP accuracy. One more convolutional layer stacked in order to implement classification, creates efficient and fast system that can be used in real time applications. The proposed method can be compared with other implementations by CNNs.

PAPER-3 **:** Traffic Sign Classification and Detection Using Deep Learning

Publication Year : 2020

Author : Sunitha.A, Shanthi.S

Journal Name: INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN TECHNOLOGY

Summary :

- The major objective of the paper was to survey the classifier algorithms used to classify and recognize the traffic signs. The methods applied in classification and recognition was either color or shape information of the traffic sign. However, the image quality in real-world traffic scenarios were usually poor due to low resolution, weather condition, varying lighting, motion blur, occlusion, scale and rotation and so on. In addition, traffic signs were usually in a variety of appearances, with high similarity, and with complicated backgrounds.
- In this paper, a recognition and classification method based on CNN-SVM has been proposed. In the training process, the deep image features were extracted by CNN in the YCbCr color space. SVM was connected with the last layer of CNN for further classification, which contributes to better training results. On the other hand, some images preprocessing procedures were conducted in the testing process, in order to eliminate those negative impacts, e.g., insufficient illumination, partial occlusion and serious deformation. Experiment-based comparison with other state-of-the-art methods verifies that their model was superior to the others both in training accuracy and

speed. Furthermore, they found that some traffic signs were miss-recognized when they apply this method in the unmanned ground vehicle.

PAPER-4 : Real-Time Traffic Sign Detection And Classification Using Machine Learning And Optical Character Recognition

Publication Year : 2020

Author : Victor Ciuntu, Hasan Ferdowsi

Journal Name : IEEE

Summary :

- The proposed traffic sign detection and classification system shown in this paper demonstrates good performance and accuracy when using a neural network in conjunction with a custom OCR algorithm for traffic sign detection and classification. This neural network implementation of Mask RCNN in Tensor pack shows an average of 5FPS when running it on a GTX 1070. This kind of frame rate was sufficient for real time processing and can be improved with a more powerful graphics card. Both the detection and the classification accuracy were over 90% and can be further improved with better filtering. The custom OCR algorithm for digit detection also showed good accuracy and performance. It achieved a classification accuracy of 93.1% and a frame rate of 18.5FPS on an Intel Core i7 7700. The task is single threaded so if there is more than one speed limit sign on one image, the frame rate will stay the same as long as there are available processing cores.

PAPER-5 : The Mapillary Traffic Sign Dataset For Detection And Classification On A Global Scale

Publication Year : 2020

Author : Christian Ertler, Jerneja Mislej, Tobias Ollmann, Lorenzo Porzi, Gerhard Neuhold, Yubin Kuang

Journal Name : Springer

Summary :

- In this work, they introduced MTSD, a large-scale traffic sign benchmark dataset that includes 105*K* images with full and partial bounding-box annotations, covering 400 traffic sign classes from all over the world. MTSD was the most diverse traffic sign benchmark dataset in terms of geographical locations, scene characteristics, and traffic sign classes. They showed in baseline experiments that decoupling detection and fine-grained classification yields superior results on previous traffic sign datasets. Additionally, in transfer-learning experiments, they show that MTSD facilitates fine-tuning and improves accuracy substantially for traffic sign datasets in a narrow domain. They saw MTSD as the first step to drive the research efforts towards solving fine-grained traffic sign detection and classification at a global scale. With the partial annotated set, they show a new scalable way to collect additional training images without the need of extra manual annotation work. Moreover, they also saw it paving the way for further research in semi-supervised learning in both classification and detection.

PAPER-6 **:** Traffic Sign Detection And Classification Based On Combination Of Mser Features And Multi-Language Ocr

Publication Year : 2020

Author : Ali Retha Hasoon Khayeat, Ashwan A. Abdulmunem, Rafeef Fauzi Najim Al-Shammari, Xianfang Sun

Journal Name : Webology

Summary :

- In this paper, a novel algorithm has been proposed for traffic sign detection and recognition based on MSER and multi-language OCR. The highest-level grayscale 2D image has been generated which contains the strongest features of the input image. The experimental

work illustrates that using the MSER with the highest intensity image can produce better results than using the standard grayscale image. Moreover, the required computation time has been reduced and enhancement has been achieved in the findings by using the following two steps: Firstly, the use of MSER instead of a machine learning approach to train a classifier to detect and understand the traffic sign(s) which is time-consuming. Secondly, the use of multi geometrical properties of text to remove non-text areas within simple thresholds. This method removes the non-text (non-sign) region(s) and retains the interest area(s). Followed by thinning operation to detect the character and merge all characters in a text-line(s). Consider that using a multi-language OCR can help the driver to understand the traffic sign in foreign countries.

PAPER-7 **:** A Robust System for Road Sign Detection and Classification Using Lenet Architecture Based On Convolutional Neural Network

Publication Year : 2019

Author : Amal Bouti, Med Adnane Mahraz, Jamal Riffi, Hamid Tairi

Journal Name : A. Bouti et al. (Springer)

Summary :

- In this paper, they tried to propose a system for the detection and classification of road signs. Their system consists of two phases: a detection phase and a classification phase. In the sign detection phase, they used HOG and SVM, while in the classification phase; they adopted an optimized model of CNN architecture. The learning of their classifier was done using the German dataset. In the experimental part, they tested their system on live video scenes and they proved that their system can achieve a very high detection and recognition rate, which was a very encouraging result.

PAPER-8 **:** Optimization of Traffic Sign Detection And Classification Based On Faster R-CNN

Publication Year : 2017

Author : Kun Qiao, Hanzhou Gu, Jiaming Liu, Pei Liu

Journal Name : International Conference on Computer Technology, Electronics and Communication

Summary :

- This paper provides an optimization of traffic sign detection and classification based on Faster R-CNN, and obtains quite good experiment results. The result shown that detection with deep learning algorithm had a greater nonlinear learning ability and a higher precision. Meanwhile, after RPN shares convolutional features with detection network, the validation accuracy of detection and classification had improved to a higher degree based on Faster R-CNN. However, this paper still had something considered to be insufficient. Even though the method had a comparatively better robustness, the computation involved was too heavy to be applied in real-time detection. As a matter of fact, the driving environment was another story in real world, which was more complicated and had other negative factors for detection and recognition. This paper would carry on further research to reduce the algorithm's complexity and made it more applicable.

PAPER-9 **:** An Overview of Traffic Sign Detection and Classification Methods

Publication Year : 2017

Author : Yassmina Saadna1, Ali Behloul1

Journal Name : Int J Multimed Info Retr

Summary :

- In this paper, they had presented an overview of some recent and efficient traffic sign detection

and classification methods. Detection methods were divided into three categories: color based that was classified according to the color space, shape based, and learning based that include deep learning methods. The recent detection methods achieve a detection rate varied from 90 to 100% with available dataset described briefly in the paper. Nevertheless, it was arduous to decide which method was the best. To obtain a high classification rate, it was necessary to adopt discriminative features and a powerful classifier. Acceptable results achieved by learning methods using hand-crafted features, and furthermore, classification methods performances boosted with deep learning methods such as CNN and they achieved a high accuracy rate>99%. Since available datasets reached saturation, a new universal dataset more complex was indispensable. However, detection and classification methods achieved a high accuracy rate, and they were still far from a real-time ADAS application where the sign should be detected and classified in real time.

PAPER-10 : Traffic-Sign Detection And Classification Under Challenging Conditions: A Deep Neural Network Based Approach

Publication Year : 2017

Author : Uday Kamal, Mohammed Abid Abrar, Sowmitra Das, Md Kamrul Hasan

Journal Name : ResearchGate

Summary :

- In this paper, they proposed a CNN-based model for detecting and classifying traffic signs from images which were captured in challenging conditions. They used CLAHE for preprocessing images and also used a CNN-based denoiser for images affected with Rain. The challenge-detector and classifier were built using VGG-16 type architecture, whereas, the localizers used a deep U-Net architecture. The model was tested and trained on the IEEE VIP Cup 2017 video dataset. The proposed model shown an overall precision and recall of 32% and 19% for real data, and, 65% and 43% for synthesized data. The reason for this bias is that there were much more frames with bounding boxes in the synthesized videos than there were in real videos. Besides, there were also some traffic-signs in the real video sequences which were not classified in the dataset. This resulted in erroneous training of the localizing models. Our training procedure ensures a low training time for the model. The model was also suitable for real-time application due to its low computational cost at each stage. As such, it can be implemented in a standard PC configuration by using a low-power GPU.

PAPER-11 : Traffic Sign Detection And Classification Using Colour Feature And Neural Network

Publication Year : 2016

Author : Md. Abdul Alim Sheikh, Alok Kole, Tanmoy Maity

Journal Name: International Conference on Intelligent Control Power and Instrumentation

Summary:

- This paper proposed Color-based detection framework that would identify and recognize road signs from static digital images in a reasonable time frame. The algorithm consists of two main stages: road sign detection, and classification. For evaluation purpose four type road signs, Stop Sign, No Entry Sign, Give Way Sign, and Speed Limit Sign were used. Altogether 300 sets images, 75 sets for each type road sign were used for training purposes. 200 images have been tested and the performance of the system is evaluated. The experimental results shown the detection rate was above 90% and the accuracy of recognition was more than 88%. The visual and quantitative results validated the usefulness of the proposed

framework. Apart from the promising results presented in this paper, there were different aspects for future research.

PAPER-12 : Indian Traffic Sign Detection and Classification Using Neural Networks

Publication Year : 2016

Author : Arun Nandewal, Abhishek Tripathi, Satyam Chandrra

Journal Name : An International Journal of Engineering Sciences

Summary :

- This paper had described a complete method to detect and classify the Indian Road traffic Signs. The algorithm proposed has three main parts, the color segmentation, blob detection and NN classification taking into consideration many outdoor environment difficulties like poor lighting and occlusion of signs. The improvements were successful study of images with background similar to color being segmented, images with environmental distortions like poor lighting, partially occluded images and introduction of Multiple NN for classification with classification of STOP and GIVE WAY signs also. Multiple signs presented at same instant were also detected successfully. Experimental results indicated the proposed methodology was efficient and accurate. Working with a standard database multiple real time images and images taken from a video sequence the algorithm given result with high accuracy. It allowed them to detect different signs invariant to the rotation and change of orientation.

PAPER-13 : A Practical Approach For Detection And Classification Of Traffic Signs Using Convolutional Neural Networks

Publication Year : 2016

Author : Hamed Habibi Aghdam, Elnaz Jahani Heravi, Domenec Puig

Journal Name : Journal of Robotics and Autonomous Systems

Summary :

- In this paper, they proposed an end-to-end learning framework for detecting and classifying traffic signs. Specifically, a lightweight ConvNet was designed for detecting traffic sings on high-resolution images. This ConvNet was trained in two steps. In the first step, training was done using the negative samples that were randomly selected from the training images. Then, it was applied on the training images in order to mine hard-negative samples. These samples were used in the next step to train the ConvNet using more appropriate data. Using this technique, the average precision of the detection ConvNet increases to 99.89% on the German Traffic Sign Detection Benchmark dataset. They also explained how to implement the sliding window technique within the detection ConvNet using dilated convolutions. They further analyzed time-to-completion of the ConvNet in different settings and showed how to use statistical information from the dataset to speed up the forward pass of the ConvNet. Using this information, 37.72 high-resolution images can be processed per second with high accuracy to locate traffic signs. Next, they modified their previously proposed ConvNet for classification of traffic signs by replacing the color image with a gray-scale image, removing the linear transformation layer, reducing the kernel size in all layers, adding one more fully connected layer and replacing LReLU activation functions with PReLU activation functions. Using these modifications, the time-to-completion of the classification ConvNet was reduced compared with the previous ConvNet. Their experiments on the German Traffic Sign Recognition Benchmark dataset showed that single classification ConvNet was able to classify

99.55% of the test image, correctly. In addition, an ensemble of 3 ConvNets was able to successfully classify 99.70% of the samples. Next, they evaluated the stability of the ConvNet empirically and showed that the classification ConvNet was robust against the Gaussian noise with s < 10. Finally, the ConvNet was further studied using the Lipschitz constant. The results suggest that, in general, ConvNets were not stable against a strong noise since they are highly non-linear functions and the output vector may significantly change even with small perturbations in the inputs.

PAPER-14 : Traffic-Sign Detection And Classification In The Wild

Publication Year : 2016

Author : Zhe Zhu, Dun Liang, Songhai Zhang, Xiaolei Huang, Baoli Li, Shimin Hu

Journal Name : IEEE Conference on Computer Vision and Pattern Recognition

Summary :

- They had created a new benchmark for simultaneously detecting and classifying traffic signs. Compared with previous traffic sign benchmarks, images in this benchmark were more variable, and signs in these images were much smaller. It contains more images than previous benchmarks, and the images had a higher resolution. Furthermore, pixel-wise segmentation of signs was provided. This benchmark provided a new challenge for the traffic sign recognition community. They had trained two networks on this benchmark: one treats all sign classes as a single category and can be regarded as a traffic sign detector. The other network can simultaneously detect and classify traffic signs.

PAPER-15 : Towards Real-Time Traffic Sign Detection And Classification

Publication Year : 2015

Author : Yi Yang, Hengliang Luo, Huarong Xu, and FuchaoWu

Journal Name : IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS

Summary :

- In this paper, they aim to address the problem of real-time traffic sign recognition. To this end, they proposed two fast algorithms for traffic sign detection and classification, respectively. The detection module first extracted traffic sign proposals by using color probability model and MSER region detector. Then an SVM classifier was used to filter out false positives and classify the remaining proposals into their super classes based on a novel color HOG feature. The classification module further classified the detected signs into their specific sub-classes within each super class. Experiments on both German and Chinese roads show that our detection module could achieve the state-of-the-art AUC values and our classification module could classify the detected signs into their sub-classes with high accuracy. Most importantly, their method could detect and classify traffic signs at ~6 fps on 1360 × 800 image, making it a good choice for real-time traffic sign recognition. It was worth to note that their method could be accelerated with GPU, which could further improve the computational efficiency.

(2.1) DIFFERENTIAL ANALYSIS (LITERATURE SURVEY)

Sr. No.	METHOD	ADVANTAGE	LIMITATION
1	YOLOv3	• Real time detection • Simple Network Structure	• Low detection Precision • Locate objects with Horizontal Bounding Box

			• Poor results for small and dense objects • Easy to dislocate
2	CNN	• Character Recognition, Natural Images • Find Edges, Corners, Endpoints, Local 2-D Structure	• Single Classification for Single Patch • The speech quality might be degraded
3	SVM	• High Accuracy performance capability • Working well even if the dataset is not linearly separable	• The High cost of computation • High memory usage
4	BLOB	• BLOBs are a good option for adding large binary data files to a database and can be easily referenced • It is easy to set access rights using rights management • Database backups or dumps contain all the data	• Not all databases permit the use of BLOBs • BLOBs are inefficient due to the amount of disk space required and access time • Creating backups is highly time consuming due to the file size of BLOBs
5	ANN	• A great capacity in predicting models • Appealing attributes of nonlinear identification and control • Suitable for non-mathematical models • Able to manage abundant number of data and input variables. • Trustworthy predictions	• Operating of neural network needs to train. • Takes long time to process of large neural network • Expending a lot of time for off-line training. • Quality predictions need large amount of data
6	HOG	• Robust • Good Performance	• Slow • Large concatenated HOG vectors
7	OCR	• Cheaper than paying someone to manually enter large amounts of text. • Much faster than someone manually entering large amounts of text. • The latest software can recreate tables and the original	• Not 100% accurate, there are likely to be some mistakes made during the process. • All documents need to be checked over carefully and then manually corrected • If the original document is of poor quality or the handwriting

		layout.	difficult to read, more mistakes will occur • Not worth doing for small amounts of text
8	R- CNN	• The network transforms the object detection problem into the classification problem and greatly improves the accuracy.	• It generates partially overlapping candidate areas from each detection target.
9	Fast R-CNN	• High detection precision • Low misdetection rate	• Non-real time detection • Locate objects with horizontal bounding box
10	SGD	• It is easier to fit in the memory due to a single training example being processed by the network. • It is computationally fast as only one sample is processed at a time. • For larger datasets, it can converge faster as it causes updates to the parameters more frequently. • Due to frequent updates, the steps taken towards the minima of the loss function have oscillations that can help to get out of the local minimums of the loss function (in case the computed position turns out to be the local minimum).	• Due to frequent updates, the steps taken towards the minima are very noisy. This can often lean the gradient descent into other directions. • Also, due to noisy steps, it may take longer to achieve convergence to the minima of the loss function. • Frequent updates are computationally expensive because of using all resources for processing one training sample at a time. • It loses the advantage of vectorized operations as it deals with only a single example at a time.

3. EXISTING WORK PROCESS

(3.1) LAYOUT OF EXISTING WORK

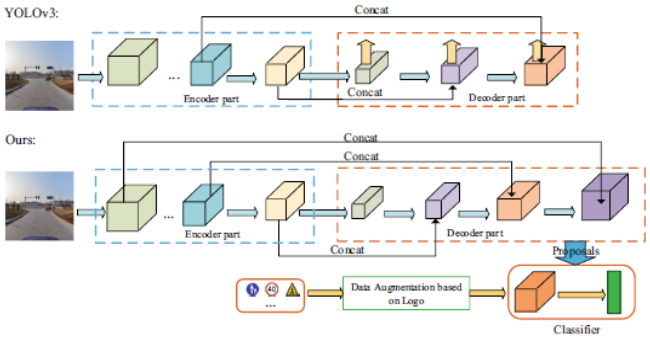

Fig. 3.1 : Base work architecture

Table 1 Our revised network based on YOLOv3, the input resolution is set to 512*512

	Encoder part				Decoder part			
YOLOV3	Conv1 block	Conv2 block	Conv3 block	Conv4 block	Conv5 block	Deconv1 block	Deconv2 block	
	256*256	128*128	64*64	32*32	16*16	32*32	64*64	-
Our network	Conv1 block	Conv2 block	Conv3 block	Conv4 block	Conv5 block	Deconv1 block	Deconv2 block	Dconv3 block
	256*256	128*128	64*64	32*32	16*16	32*32	64*64	128*128

Table : 3.1 : YOLOV3 Revised Network

(3.2) Base Work Explanation

- Their method was based on the YOLOv3 structure.
- Compared with YOLO and YOLO9000, YOLOv3 uses a new feature extracting network with some shortcut connections : darknet53, and does predictions across 3 scales.
- Instead of using softmax loss, the authors use logistic loss to deal with more complex domains like the Open Images Dataset.
- The YOLOv3 network was organized as follows : the classifier network darknet53 with many Residual blocks achieves similar performance to ResNet-152 but 2 × speeds.
- The darknet53 executes down-sampling operation using a convolution layer with stride 2 instead of using pooling layer.
- After each down-sampling operation, the Residual block was employed for in that scale.
- The darknet53 contains 5 down-sampling operations and results in 32-fold scale zooming.
- Based on the darknet53, YOLOv3 adds 3 prediction blocks from different scales respectively.
- Specifically, for each prediction block, the up-sampling operation and router operation are employed to get higher resolution feature maps.
- The YOLOv3 structure was illustrated in the Fig. 3.1.
- Fig. 3.1 shows the architecture of their method. The RPM consists of the feature extraction network and the location prediction network. The CM accepts the proposals from the RPM for further classification. In this work, they select darknet19 as our classifier network
- Their goal was to design an ultra-efficient traffic-sign object detection network towards real scenes, therefore in this work they mainly focus on the small object detection and unlimited traffic sign classification while ensuring the real-time performance.
- However, the general detection framework did not work since many classes were not included in the dataset.
- To solve this problem they proposed a novel two-level network architecture, which was formed by two modules, i.e., the region proposal module (RPM) and the classifier module (CM).
- The RPM aims to regress the locations of the object whereas the CM aims to predict multi-class labels based on the locations, as shown in Fig. 3.1.
- For the unlimited traffic sign detection, they design two-level detection architecture, as shown in Fig. 3.1.
- In the first stage, they focus on regressing the locations of the traffic sign by using the RPM.
- And in the second stage, they target at classifying the specific categories of the traffic signs by the CM.
- They view all traffic signs as one class and use the RPM to propose the location of the traffic signs, and then they add an extra classifier module to acquire the labels of the objects.

(3.3) Data augmentation based on logo for the RM

- There were many datasets for traffic sign detection and classification, such as TT100k, GTSDB, and GTSRB and so on.
- However, the traffic signs distributed seriously unevenly no matter which country it was.

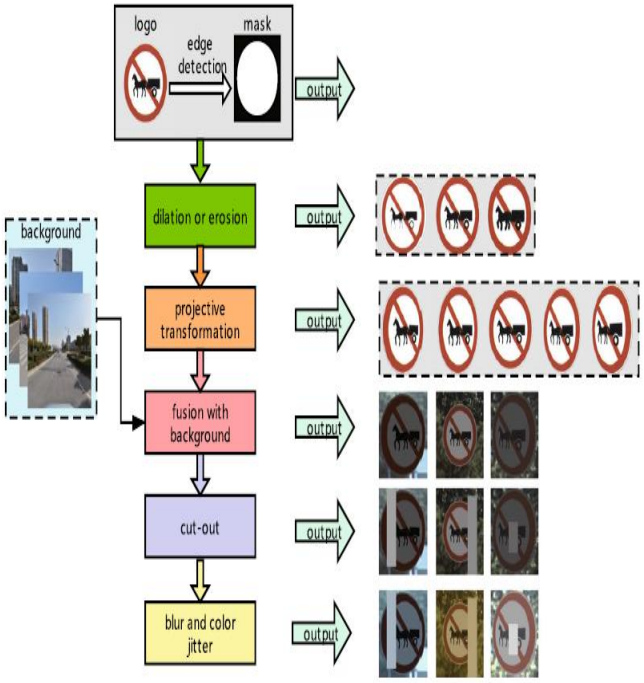

Fig. (3.3.1) : The pipeline of the data augmentation based on logo

- For example, in TT100k, *No Parking*, *No Entry* and *Speed Limit* signs were very common with the numbers of more than 1000, while the signs such as *Mountain danger*, *Falling rocks* were very rare.
- The current situation of non-uniform distribution of traffic signs makes the recognition of traffic signs a difficult problem.

➢ Through the careful observation of the existing data sets, they find that the differences in the same traffic signs were mainly reflected in the following aspects : viewpoint, background, size, color, illumination, contrast, occlusion, and pollution.
➢ To get enough and evenly distributed images and simulate as many situations as possible in the real world, a specific data augmentation technology based on traffic sign logo was proposed.
➢ The pipeline of the data augmentation based on logo is shown in Fig. (3.3.1).

- Firstly, the logo of traffic sign was acquired mutually and the mask of the sign was formed automatically by canny operator.
- To simulate the change of viewpoint in the real scene, the perspective transformation was applied to the traffic sign logo.
- A 2-d image was transformed into another plane through perspective transformation.
- This process can be expressed as:

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} .x' = \frac{a}{c}, y' = \frac{b}{c}$$

- *Mij* was the transformation matrix including 9 parameters,*(x, y)* was the coordinates before the transformation, *(ẋ , ẏ)* was the coordinates after transformation.

➢ In real traffic scenarios, the shapes of traffic signs vary in thickness due to the influence of illumination, contamination, and manufacturing technology and so on.
➢ The erosion and dilation were the primary morphological operation of the image.
➢ These two operations have completely opposite effects, which can further increase the variations of the samples.

- The next step was the fusion of the sign and the background.
- We select randomly an image from the real traffic scenes as the background, and crop out a sub-image with the same size of the sign.
- Then the fusion operation of the sign and the sub-image can described as follows:

$$I_o = mask \odot I_s + (1 - mask) \odot I_b$$

- Where *Is* denotes the traffic sign logo, *mask* denotes the mask of the sign and *Ib* denotes the image cropped out from the real traffic scenes.

- Finally, color jitter was employed to the fused image.
- In addition, we add Gaussian noise with different kernel size including {1×1, 3×3, 5×5, and 7×7} to the image for the blurring and the cut-out was applied to the synthetic image which enables the network to have more generalization performance.
- The reasonable experiments were conducted to valid the effectiveness of our data augmentation method.
- Some synthetic images are shown in Fig. (3.3.2).

Fig. (3.3.2) : Some synthetic images

Fig. (3.3.3) : Some detection results when the input resolution is 512*512. The label appears as an image in the bottom right corner of the box

4. PROPOSED WORK PROCESS

(4.1) PROPOSED WORK LAYOUT

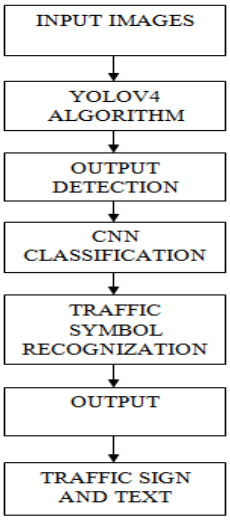

Fig. 4 : Proposed Work Layout

(4.2) PROPOSED WORK ALGORITHM

Step: 1: Gathering image from dataset which contains traffic signs.

Step: 2: Collected images divided into their particular group / subgroup.

Step: 3: Renaming all images according their group / subgroup.

Step: 4: I need to Labelling out all images using LabelImg/ Bounding Box tool.

Step: 5: That creates a text file according images.

Step: 6: Put all images and text files into one folder.

Step: 7: I need software that runs my code which is Google colab.

Step: 8: Now I need to mounting Google drive and need to connect GPU.

Step: 9: Download YOLOV4 regarding repositories.

Step: 10: I need to change configuration file according classes I take and also change file for training and testing.

Step: 11: Put in training and it takes time according my data which is taken by me.

Step: 12: After training I get .weights file which are used for testing.
Step: 13: Then I test my images and I get bounding box with labelling images.

(4.3) PROPOSED WORK EXPLAINATION

- I propose my work same as my base work but I use some new methodology.
- In my base paper, they used yolov3 for detection and darknet 19 for classification.
- Here I want to use yolov4 instead of yolov3 which is improved version of yolov3 and more accurate.
- And for classification purpose I want to use CNN for image classification.
- Initially I need input images. Those images collected from GTSDB dataset.
- I get these images from their official site using jupyter notebook.
- These images are in ppm format then I need to convert in .jpg format.
- After that I have bounding box / labelimg tool which is used to find out class id, x, y axis and height and weight of the bounding box.
- And these values are saved in text file automatically.
- After that I need platform when we can run our code. And that platform is Google colab.
- After that I need to perform coding for training and testing perspective.
- For that I need configuration file.
- After that we need to process the file.
- Then we get training and testing files.
- And finally I get traffic sign with bounding box.
- After getting bounding box we perform classification that is image classification using cnn.
- And after performing that I have labeling traffic sign.
- And that is my final output.

5. DATASET, IMPLEMENTATION AND RESULT

(5.1) DATASET

(5.1.1) TT100K dataset.

- TT100K means Tsinghua- Tencent 100k.
- This dataset is used for detecting and classifying the traffic signs.
- TT100K is a country-specific traffic sign dataset with images collected in China.
- That contains 10,000 images with traffic signs and 90,000 background images without any traffic signs.

(5.1.2) GTSDB dataset.

- GTSDB means German Traffic Sign Detection Benchmark
- GTSDB contains 3 categories namely mandatory, prohibitory and danger.
- GTSDB data set is split into a training set of 600 images and a test set of 300 images and covers natural traffic scenes of various roads (road, rural, urban) recorded during the day and dusk.

(5.2) UTILIZING TOOLS & TECHNOLOGY PRESENTATION

1) Python:

- Python is a high level, interpreted and general-purpose dynamic programming language that focuses on code readability. It has fewer steps when compared to Java and C. It was founded in 1991 by developer Guido Van Rossum. It is used in many organizations as it supports multiple programming paradigms. It also performs automatic memory management.

Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses. There are two major Python versions: Python 2 and Python 3. Both are quite different.

2) NumPy:

- NumPy is a Python library used for working with arrays and it stands for 'Numerical Python'. NumPy was created in 2005 by Travis Oliphant. It is an open- source project and you can use it freely. It is a library consisting of multi-dimensional array object sand a collection of routines for processing of array. It also has functions for working in domain of linear algebra, Fourier transform, and matrices. Which stands for Numerical Python, is a library consisting of multidimensional array.

3) Kera's:

- Kera's runs on top of open-source machine libraries like TensorFlow, Theano or Cognitive Toolkit (CNTK). Theano is a python library used for fast numerical computation tasks. kesars is an advanced neural networks API, written in Python and accomplished by running on top of TensorFlow, CNTK, or Thea no. It uses libraries such as Python, C#, C++ or stand-alone machine learning tool kits. It was developed with a focus on enabling quick experimentation. Kera's is based on a minimal structure that provides a clean and easy way to create deep learning models based on TensorFlow or Theano. Kera's is designed to quickly define deep learning models. Well, Kera's is an optimal choice for deep learning applications. Being equal to go from idea to outcome with the least possible delay is key to doing good quality research. The version used by the SRCNN model is python.

4) Tensor Flow :

- Tensor Flow is a software library or framework, designed by the Google team to implement machine learning and deep learning concepts in the easiest manner. It combines the computational algebra of optimization techniques for easy calculation of many mathematical expressions. It is a foundation library that can be used to produce Deep Learning models openly or by using wrapper libraries that create simpler the process built on top of TensorFlow. It includes a variety of machine learning and deep learning algorithms. The version used by the SRCNN model. It includes a feature that defines, optimizes and calculates mathematical expressions easily with the help of multi-dimensional arrays called tensors.

5) Jupyter Notebook

- The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter.

6) Google Colab

- Collaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.

7) Transfer Learning

- Transfer learning is a machine learning method where we reuse a pre-trained model as the starting point for a model on a new task. To put it simply—a model trained on one task is repurposed on a second, related task as an optimization that allows rapid progress when modeling the second task.

8) GPU

- GPU stands for Graphics processing unit. Using Google Colab environment, we have free access to the "NVIDIA Tesla K80" GPU. But keep in mind that you are limited to use it for 12 hours continuously, after that you may not be able to access it for a particular duration of time unless you purchase Colab pro.

9) Darknet

- Darknet is an open source neural network framework written in C and CUDA. It is fast, easy to install, and supports CPU and GPU computation. It is a highly accurate (accuracy for custom trained model depends on training data, epochs, batch size and some other factors) framework for real time object detection (also can be used for images).

(5.3) EXISTING SYSTEM IMPLEMENTATION

- I have data that are images and related text files. I need to train and test that data.
- So I use Google Collaboratory for performing operation on my data.
- My base work is completed into Ubuntu 16.04 LTS.
- But here, I am using Google Colab instead of Linux for executing existing work.
- So that I am going into my drive and create one folder that is YOLOV3 and under this folder I create new colab file.
- Initially I need to connect GPU through Notebook Setting. Otherwise I get an error.
- Next I performed following steps.

Step : 1: Here I need to mounting Google drive. That means I give the permission to use my drive.

```
[1]  from google.colab import drive
     drive.mount('/content/gdrive')

Mounted at /content/gdrive
```

Fig. (5.3.1) : Mounting Google Drive

Step : 2 : Here after mounting I link to my drive.

```
[2]  !ln -s /content/gdrive/My\ Drive/ /mydrive
```

Fig. (5.3.2) Link to my drive

Step : 3 : Now I am in my drive and navigate to YOLOV3 folder.

```
[3]  %cd /mydrive/YOLOV3
     /content/gdrive/My Drive/YOLOV3
```

Fig. (5.3.3) : Navigating YOLOV3 folder

Step : 4 : For performing multiple operations I need to download Darknet Repository. This contains multiple folders and files.

```
[4]  !git clone https://github.com/AlexeyAB/darknet.git

Cloning into 'darknet'...
remote: Enumerating objects: 15412, done.
remote: Total 15412 (delta 0), reused 0 (delta 0), pack-reused 15412
Receiving objects: 100% (15412/15412), 14.02 MiB | 7.31 MiB/s, done.
Resolving deltas: 100% (10356/10356), done.
Checking out files: 100% (2050/2050), done.
```

Fig. (5.3.4) : Darknet Repository Download

Step : 5 : After downloading darknet, it will take place in YOLOV3 folder. Darknet contains Makefile and I need to change that Makefile. Darknet also contains cfg, data, backup, etc folders. In cfg folder yolov3.cfg file available and I need to change that configuration file according my classes and save / renaming it. Cfg includes batch, subdivisions, max-batches, steps, filters, classes, etc. And this cfg file use for training and testing.

```
[5]  %cd darknet/
     !sed -i 's/OPENCV=0/OPENCV=1/' Makefile
     !sed -i 's/GPU=0/GPU=1/' Makefile
     !sed -i 's/CUDNN=0/CUDNN=1/' Makefile
     !sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile
     !sed -i 's/LIBSO=0/LIBSO=1/' Makefile
     !sed -i 's/OPENMP=0/OPENMP=1/' Makefile

     /content/gdrive/MyDrive/YOLOV3/darknet
```

Fig. (5.3.5) : Make changes in Makefile

Step : 6 : After changes made in Makefile, I need to Make / Run that file.

```
[6]  !make
     g++ -std=c++11 -std=c++11 -I
```

Fig. (5.3.6) : Run the Makefile

Step : 7 : I need to make changes in darknet. So that I need to give permission.

Step : 8 : I need to locate data folder and put my dataset and other related files into that data folder.

Step : 9 : Run python process.py file which create train and test file according my dataset.

Step : 10 : I need to back out darknet folder.

```
[7]  !chmod +x ./darknet

[8]  %cd /content/gdrive/MyDrive/YOLOV3/darknet/data
     /content/gdrive/MyDrive/YOLOV3/darknet/data

[9]  !python process.py

[10] %cd ..
     /content/gdrive/MyDrive/YOLOV3/darknet
```

Fig. (5.3.7) : Generate train and test file

Step : 11 : Now I have train and test files. So I start training process. During / after training I get weights files. And we can test model using that weights files.

```
[11] !./darknet detector train data/TRAFFIC_SIGNS.data cfg/yolov3_custom.cfg darknet53.conv.74 -dont_show -map
     total_bbox = 160967, rewritten_bbox = 0.000000 %
```

Fig. (5.3.8) : Train the Model

Step : 12 : Using this code I can get chart which contains information about training process.

```
[12] def imShow(path):
       import cv2
       import matplotlib.pyplot as plt
       %matplotlib inline

       image = cv2.imread(path)
       height, width = image.shape[:2]
       resized_image = cv2.resize(image,(3*width, 3*height), interpolation = cv2.INTER_CUBIC)

       fig = plt.gcf()
       fig.set_size_inches(18, 10)
       plt.axis("off")
       plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
       plt.show()
     imShow('chart.png')
```

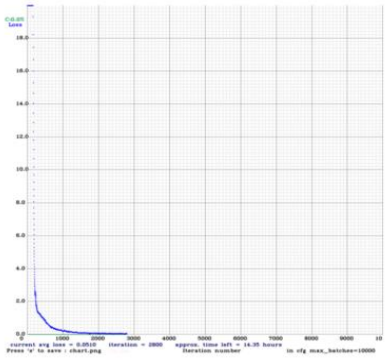

Fig. (5.3.9) : Training Chart

Step : 13 : During / after training I can get weights on backup folder. So I can test my model using that trained weights.

```
[14] img_path = 'data/TRAFFIC_SIGNS/CONSTRUCTION_DANGER_10.jpg'
     !./darknet detector test data/TRAFFIC_SIGNS.data cfg/test_yolov3_custom.cfg backup/yolov3_custom_last.weights {img_path} -dont_show

      CUDA-version: 11010 (11020), cuDNN: 7.6.5, CUDNN_HALF=1, GPU count: 1
      CUDNN_HALF=1
      OpenCV version: 3.2.0
       0 : compute_capability = 750, cudnn_half = 1, GPU: Tesla T4
     net.optimized_memory = 0
     mini_batch = 1, batch = 1, time_steps = 1, train = 0
         layer   filters  size/strd(dil)      input                output
           0 Create CUDA-stream - 0
      Create cudnn-handle 0
```

Fig. (5.3.10) : Test the Model

Step : 14 : Now I can show my prediction which I was tested above and this prediction gives me bounding box with their labelling. And this way I performed testing operation on other images and that images shows my results.

```
[15] import matplotlib.pyplot as plt
     fig = plt.figure(figsize=(12,12))
     plt.axis(False)
     processed_image = plt.imread("./predictions.jpg")
     plt.imshow(processed_image)

     <matplotlib.image.AxesImage at 0x7f450df7ab90>
```

Fig. (5.3.11) : Predicted Construction Images

Fig. (5.3.12) : Predicted Keep Left Images

Fig. (5.3.13) : Predicted No Entry Images

Fig. (5.3.14) : Predicted Road Narrows Images

Fig. (5.3.15) : Predicted Slippery Road Images

Step : 15 : I can also tried to testing on videos this way and I can save it into my data folder.

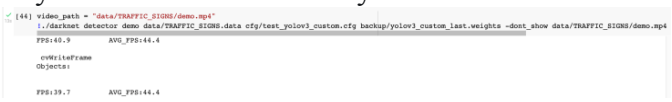

Fig. (5.3.16) : Test the Video

(5.4) PROPOSED SYSTEM IMPLEMENTATION

- I have data that are images and related text files. I need to train and test that data.
- So I use Google Collaboratory for performing operation on my data.
- So that I am going into my drive and create one folder that is YOLOV4 and under this folder I create new colab file.
- Initially I need to connect GPU through Notebook Setting. Otherwise I get an error.
- Here I have not green tick because when I taking Screen Shot GPU was disconnected.
- Next I performed following steps.

Step : 1: I need to mounting Google drive. That means I give the permission to use my drive.

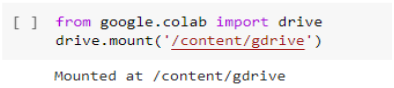

Fig. (5.4.1) : Mounting Google Drive

Step : 2 : Here after mounting I link to my drive.

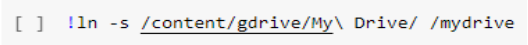

Fig. (5.4.2) Link to my drive

Step : 3 : Now I am in my drive and navigate to YOLOV4 folder.

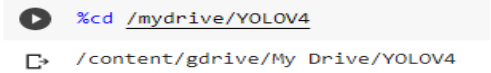

Fig. (5.4.3) : Navigating YOLOV4 folder

Step : 4 : For performing multiple operations I need to download Darknet Repository. This contains multiple folders and files.

```
[ ]  !git clone https://github.com/AlexeyAB/darknet.git

     fatal: destination path 'darknet' already exists and is not an empty directory.
```

Fig. (5.4.4) : Darknet Repository Download

Step : 5 : After downloading darknet, it will take place in YOLOV4 folder. Darknet contains Makefile and I need to change that Makefile. Darknet also contains cfg, data, backup, etc folders. In cfg folder yolov3.cfg file available and I need to change that configuration file according my classes and save / renaming it. Cfg includes batch, subdivisions, max-batches, steps, filters, classes, etc. And this cfg file use for training and testing. Here shows fatal error because I disconnected and run all cell again and darknet repository already downloaded.

```
[ ]  %cd darknet/
     !sed -i 's/OPENCV=0/OPENCV=1/' Makefile
     !sed -i 's/GPU=0/GPU=1/' Makefile
     !sed -i 's/CUDNN=0/CUDNN=1/' Makefile
     !sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile
     !sed -i 's/LIBSO=0/LIBSO=1/' Makefile
     !sed -i 's/OPENMP=0/OPENMP=1/' Makefile

     /content/gdrive/MyDrive/YOLOV4/darknet
```

Fig. (5.4.5) : Make changes in Makefile

Step : 6 : After changes made in Makefile, I need to Make / Run that file.

```
[ ]  !make

./src/image_opencv.cpp: In function 'void draw_train_loss(char*, voi
./src/image_opencv.cpp:1147:13: warning: this 'if' clause does not g
             if (iteration_old == 0)
             ^~
```

Fig. (5.4.6) : Run the Makefile

Step : 7 : I download yolov4 pretrained weights. And this is called Transfer Learning.

```
[ ]  !wget https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.weights

--2022-05-05 06:11:52--  https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.weights
Resolving github.com (github.com)... 52.192.72.89
Connecting to github.com (github.com)|52.192.72.89|:443... connected.
HTTP request sent, awaiting response... 302 Found

[ ]  !wget https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.conv.137

--2022-05-05 06:12:28--  https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.conv.137
Resolving github.com (github.com)... 52.69.186.44
Connecting to github.com (github.com)|52.69.186.44|:443... connected.
```

Fig. (5.4.7) : Download Pre-trained weights

Step : 8 : I need to make changes in darknet. So that I need to give permission.

Step : 9 : I need to locate data folder and put my dataset and other related files into that data folder.

Step : 10 : Run python process.py file which create train and test file according my dataset.

Step : 11 : I need to back out darknet folder.

```
[ ]  !chmod +x ./darknet

[ ]  %cd /content/gdrive/MyDrive/YOLOV4/darknet/data

     /content/gdrive/MyDrive/YOLOV4/darknet/data

[ ]  !python process.py

[ ]  %cd ..

     /content/gdrive/MyDrive/YOLOV4/darknet
```

Fig. (5.4.8) : Generate train and test file

Step : 12 : Now I have train and test files. So I start training process. During / after training I get weights files. And we can test model using that weights files.

```
[ ] !./darknet detector train data/TRAFFIC_SIGNS.data cfg/yolov4-custom.cfg yolov4.conv.137 -dont_show -map

Streaming output truncated to the last 5000 lines.
 total_bbox = 235425, rewritten_bbox = 0.000850 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.726790), count: 16, class_loss = 1.124256,
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.729584), count: 9, class_loss = 0.147945,
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.000000), count: 1, class_loss = 0.000045,
 total_bbox = 235450, rewritten_bbox = 0.000849 %
```

Fig. (5.4.9) : Train the Model

Step : 13 : Using this code I can get chart which contains information about training process.

```
[ ]  def imShow(path):
       import cv2
       import matplotlib.pyplot as plt
       %matplotlib inline

       image = cv2.imread(path)
       height, width = image.shape[:2]
       resized_image = cv2.resize(image,(3*width, 3*height), interpolation = cv2.INTER_CUBIC)

       fig = plt.gcf()
       fig.set_size_inches(18, 10)
       plt.axis("off")
       plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
       plt.show()
     imShow('chart.png')
```

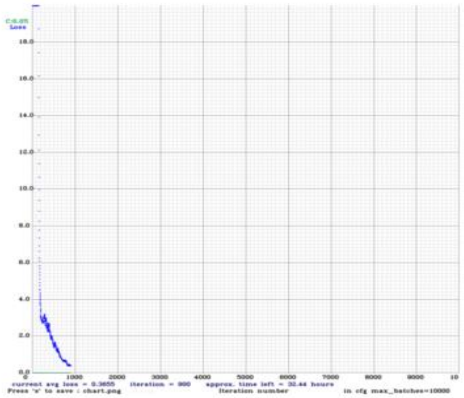

Fig. (5.4.10) : Training Chart

Step : 14 : During / after training I can get weights on backup folder. So I can test my model using that trained weights.

```
[ ] img_path = "data/TRAFFIC_SIGNS/CONSTRUCTION_DANGER_10.jpg"
    !./darknet detector test data/TRAFFIC_SIGNS.data cfg/test-yolov4-custom.cfg backup/yolov4-custom_last.weights {img_path} -dont_s

 CUDA-version: 11010 (11020), cuDNN: 7.6.5, CUDNN_HALF=1, GPU count: 1
 CUDNN_HALF=1
 OpenCV version: 3.2.0
 0 : compute_capability = 370, cudnn_half = 0, GPU: Tesla K80
net.optimized_memory = 0
mini_batch = 1, batch = 64, time_steps = 1, train = 0
   layer   filters  size/strd(dil)      input                output
   0 Create CUDA-stream - 0
 Create cudnn-handle 0
conv     32      3 x 3/ 1    416 x 416 x   3 ->  416 x 416 x  32 0.299 BF
   1 conv     64      3 x 3/ 2    416 x 416 x  32 ->  208 x 208 x  64 1.595 BF
   2 conv     64      1 x 1/ 1    208 x 208 x  64 ->  208 x 208 x  64 0.354 BF
   3 route  1                     ->  208 x 208 x  64
```

Fig. (5.4.11) : Test the Model

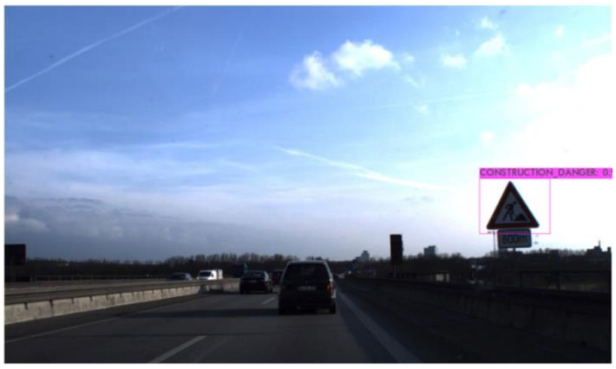

Fig. (5.4.12) : Predicted Construction Images

Step : 15 : Now I can show my prediction which I was tested above and this prediction gives me bounding box with their labelling. And this way I performed testing operation on other images and that images shows my results.

```
[ ] import matplotlib.pyplot as plt
    fig = plt.figure(figsize=(12,12))
    plt.axis(False)
    processed_image = plt.imread("./predictions.jpg")
    plt.imshow(processed_image)

    <matplotlib.image.AxesImage at 0x7f45b4a35510>
```

Fig. (5.4.13) : Predicted Keep Left Images

Fig. (5.4.14) : Predicted No Entry Images

Fig. (5.4.15) : Predicted Road Narrows Images

Fig. (5.4.16) : Predicted Slippery Images

Step : 15 : I can also try to testing on videos this way and I can save it into my data folder.

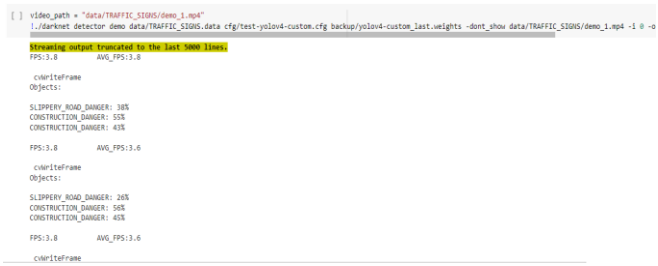

Fig. (5.4.17) : Test the Video

(5.5) FUTURE WORK

- In future I want to apply classifier which can recognize text also. I also take more images for training and testing. I want to apply this type of methods on more videos. And I want to apply this type of methods on that type of system which performing automatic operation. That means Self Driving Vehicles.

6. CONCLUSION

I conclude that I learned a lot during this research. Every day I would get some error and it would be solved on the second or third day using Google or YouTube. I knew Ubuntu. But there was a lot to learn in Ubuntu during this research. I installed Ubuntu myself and used dual windows and Ubuntu. I didn't know YOLO, Google Colab, Jupyter Notebook, etc before. But now I know how to operate it. I saw many research papers in which YOLO is used for object detection. So that's why I want to continue my research according YOLO. When I practical this, I found out that YOLOV4 and YOLOV3 are the same. But YOLO4 is faster than YOLO3. In this Research I discuss a variety of methods used for traffic sign detection and classification. In my base paper, traffic sign detection was completed using YOLOV3 and classification are completed using Darknet19. I propose my work in Proposed paper as same as base work but I including some other new methods. I using YOLOV4 instead of YOLOV3 because it is implemented version of YOLOV3 and it is more accurate.

7. REFERENCES

1. Manjiri Bichkar, Suyasha Bobhate, Prof. Sonal Chaudhari (2021) "TRAFFIC SIGN CLASSIFICATION AND DETECTION OF INDIAN TRAFFIC SIGNS USING DEEP LEARNING " International Journal of Scientific Research in Computer Science, Engineering and Information Technology

2. Valentyn Sichkar , Sergey A. Kolyubin (2020) "REAL TIME DETECTION AND CLASSIFICATION OF TRAFFIC SIGNS BASED ON YOLO VERSION 3 ALGORITHM " Scientific and Technical Journal of Information Technologies Mechanics and Optics

3. Sunitha.A, Shanthi.S (2020) "TRAFFIC SIGN CLASSIFICATION AND DETECTION USING DEEP LEARNING" INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN TECHNOLOGY

4. Victor Ciuntu, Hasan Ferdowsi (2020) "Real-Time Traffic Sign Detection and Classification Using Machine Learning and Optical Character Recognition " IEEE

5. Christian Ertler, Jerneja Mislej, Tobias Ollmann, Lorenzo Porzi, Gerhard Neuhold, Yubin Kuang (2020) " The Mapillary Traffic Sign Dataset for Detection and Classification on a Global Scale " Springer

6. Ali Retha Hasoon Khayeat, Ashwan A. Abdulmunem, Rafeef Fauzi Najim Al-Shammari, Xianfang Sun (2020) "TRAFFIC SIGN DETECTION AND CLASSIFICATION BASED ON COMBINATION OF MSER FEATURES

AND MULTI-LANGUAGE OCR ” Webology

7. Amal Bouti, Med Adnane Mahraz, Jamal Riffi, Hamid Tairi (2019) “A robust system for road sign detection and classification using LeNet architecture based on convolutional neural network” A. Bouti et al. (Springer)

8. Kun Qiao, Hanzhou Gu, Jiaming Liu, Pei Liu (2017) “OPTIMIZATION OF TRAFFIC SIGN DETECTION AND CLASSIFICATION BASED ON FASTER R-CNN ” International Conference on Computer Technology, Electronics and Communication

9. Yassmina Saadna1, Ali Behloul1 (2017) “AN OVERVIEW OF TRAFFIC SIGN DETECTION AND CLASSIFICATION METHODS ” Int J Multimed Info Retr

10. Uday Kamal, Mohammed Abid Abrar, Sowmitra Das, Md Kamrul Hasan (2017) “TRAFFIC-SIGN DETECTION AND CLASSIFICATION UNDER CHALLENGING CONDITIONS: A DEEP NEURAL NETWORK BASED APPROACH ” ResearchGate

11. Md. Abdul Alim Sheikh, Alok Kole, Tanmoy Maity (2016) “TRAFFIC SIGN DETECTION AND CLASSIFICATION USING COLOUR FEATURE AND NEURAL NETWORK ” International Conference on Intelligent Control Power and Instrumentation

12. Arun Nandewal, Abhishek Tripathi, Satyam Chandrra (2016) “ INDIAN TRAFFIC SIGN DETECTION AND CLASSIFICATION USING NEURAL NETWORKS ” An International Journal of Engineering Sciences

13. Hamed Habibi Aghdam, Elnaz Jahani Heravi, Domenec Puig (2016) “A practical approach for detection and classification of traffic signs using Convolutional Neural Networks” Journal of Robotics and Autonomous Systems

14. Zhe Zhu, Dun Liang, Songhai Zhang, Xiaolei Huang, Baoli Li, Shimin Hu (2016) “TRAFFIC-SIGN DETECTION AND CLASSIFICATION IN THE WILD ” IEEE Conference on Computer Vision and Pattern Recognition

15. Yi Yang, Hengliang Luo, Huarong Xu, and FuchaoWu (2015) “TOWARDS REAL-TIME TRAFFIC SIGN DETECTION AND CLASSIFICATION ” IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS

16. Aarohi Singla “Object Detection Using YOLO v4 on Custom Dataset | Practical Implementation” https://www.youtube.com/watch?v=yGMZOD44GrI&list=PLv8Cp2NvcY8ATPRk4LycJWr5YWB_svhrW&index=2, Code With Aarohi

