OBSTACLE AND LANE DETECTION USING NEURAL NETWORKS

By Kunjan Mhaske and Mayur Jawalkar

Table of Contents

Introduction:

Modern cars are equipped with advanced driver-assist features such as lane detection, vehicle following, maintaining distance from vehicle in front. A lot of research work is going on to improvise the performance of the driver assistant and self-driving car algorithms. Following lane, discipline is one of the crucial tasks considering the safety aspect. Having a clear vision and understanding of the environment aspects like lanes, vehicles, poles etc. will make self- driving cars more robust. In this project, we propose to work on end-to-end lane detection with obstacle detection like cars, trucks, and other things. The end-to-end lane detection project will help cars in situations like speed maintaining, making smart decisions like changing lanes. Also, the obstacle detection will improvise the environmental awareness of the car. We will be working with multiple Artificial Neural Networks in general throughout this project. This will reduce the model dependency of algorithm as well as avoid training-testing phase. One should caliber the algorithm after certain period or the Neural Networks could learn thing as they work along. Use of already established network with unanimous training of all weather and lightning conditions could boost up the accuracy and dependency.

The crucial thing while driving is to take the decision before something happens and with quick response time. To take efficient decision, relevant data with intelligent filtering system is required. If data is too much and not processed, it will put extra burden on resources which will create the time lag in decision making process. The combination obstacle detection and lane detection make the efficient decision more accurate as they provide the necessary enough information to human driver or self-driving algorithm in real time i.e. very minimal process time. The implementation of this system requires the computer onboard with necessary memory, however, with embedded electronics and latest integrated circuit technology, one could make the hard-coded decision table prints on the ICs to enhance the final results. This project only deals with the algorithm of obstacle detection and lane detection using neural networks on computer and gives output by processing input video frame by frame.

Learning Objectives:

This project will expand our knowledge base regarding the applications of artificial neural networks in the computer vision field as well as artificial intelligence system. After successful implementation of this project, we expect us to learn and show the insights of computer vision field that could be beneficial for everyone who come to know this report and project. Some of the things are as follows:

1. Real world application of computer vision using artificial neural networks
2. Handle image and video stream data
3. Experimentation with pre-trained neural network's configurations and inputs
4. Understanding and implementation of image and video processing algorithms
5. Understand and implementation of object detection algorithm
6. Effects of multiple algorithms with respect to computational resources

The primal aim of this work is to develop the computer vision system based on artificial neural networks that utilizes the input video stream of the car's environment like road, vehicles, other obstacles from dash camera. This video input will be used to identify the lanes and obstacles or objects and improve the assistance provided to the driver while on the road in terms of lane detection and obstacle detection. This will help to improve the safety of the travelers as well as contribute towards the decision making of self-driving car algorithms.

Artificial Neural Networks in Computer Vision:

Artificial Neural Networks (ANNs) are currently major components of the most ingenious inventions like self-driving cars, object detections, image recreation, facial recognition and restructuring, etc. This incredible ability to learn from various types of data and environment makes them the first choice to develop any smart system in computer intelligence field. Neural networks lie in the core of such products that are highly adaptive to various data types and extends application of it in a great scale.

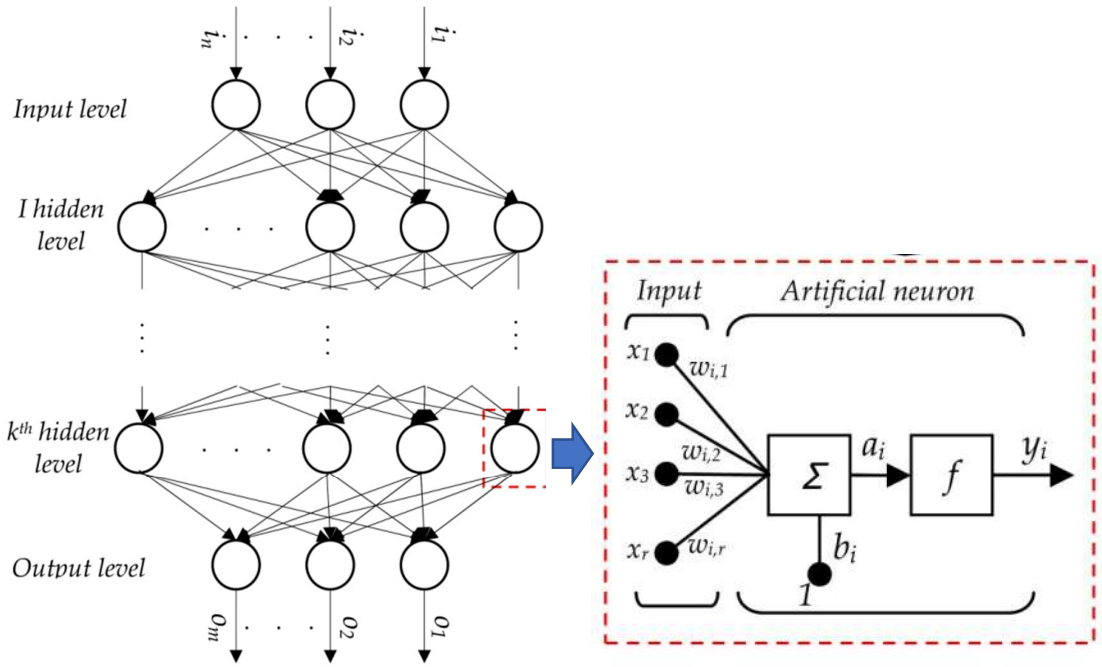
What is Artificial Neural Network (ANN)?

It is artificial representation of working of human being's nervous system that is made of network of multiple neurons. Just like human nervous system, an artificial neuron collates all the inputs and performs an operation on them. In the end, it transmits the output to all other neurons of the next layer to which it is connected. The neural network is divided into layer of 3 types, using those, it gives results based on the information provided in data.

Input Layer: Training data is fed through this

Hidden Layer: Intermediate layers that learns complex relationships and pattern in data

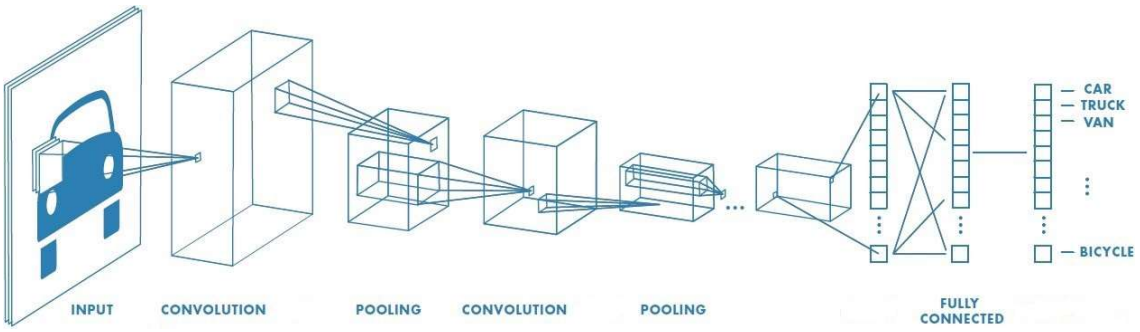
Output Layer: Gives out best possible outputs extracted from previous layers



Neural Network Neuron

The more specific versions of the artificial neural networks are CNN, RNN and other networks with different capacity of handling complex relations in different types of data. One of the most advancement in computer vision is Convolution Neural Networks (CNNs). Object detection with some enhancements are the basic developments achieved through computer vision that contains variations in viewpoints in which objects can have different positions and angles in images, difference in illumination, incomplete view of object in images, background clutter etc. However, the traditional methods of object detection like KNN, Linear Classifiers has certain drawbacks like they are unable to cater to the vast amount of variations in images. Here, CNN comes into light where it can consecutively model the small pieces of information and combine them in network. We can say that the first layer will try to detect the edges, contours and form templates for edge detection. Then consecutive layers will try to combine them into simpler shapes and eventually merge them into template of multiple object positions, illumination, scales, etc. The final layer will match the input image with all templates and predict the best possible case that is the weighted sum of all templates. Hence, CNN are able to model the complex variations with more accurate predictions.

CNN typically contains 3 types of layers. Convolution layer, Pooling layer and Fully Connected layer. Convolution layers form the core of the network. They can be visualized as 3-dimensional cuboid blocks where they can map all dimensions to colors, another block with filter that runs from top left corner of input image and sweep it till the bottom left corner. At each position, it updates the weighted sum of pixel. Pooling layer are used to reduce the size of image after the use of padding in convolution layers. It works by sampling in each layer using filtering. Generally, max-pooling is used over others like average pooling. Fully connected layer is at the end of convolution and pooling layers in which each pixel is considered as separate neuron. This last layer may contain may neurons as the number of classes available for prediction. Hence, the output of each neuron will give its prediction confidence of respective class.



With different configuration and additional layers, basic architecture of CNN can be used to do deep learning in computer vision. Advanced CNNs variants like AlexNet, GoogleNet, VGGNet, ResNet, ImageNet, etc. In this work, we are focusing on the use of YOLO for object detection and encoder-decoder network ENet based LaneNet for lane detection.

Obstacle / Object Detection using YOLO:

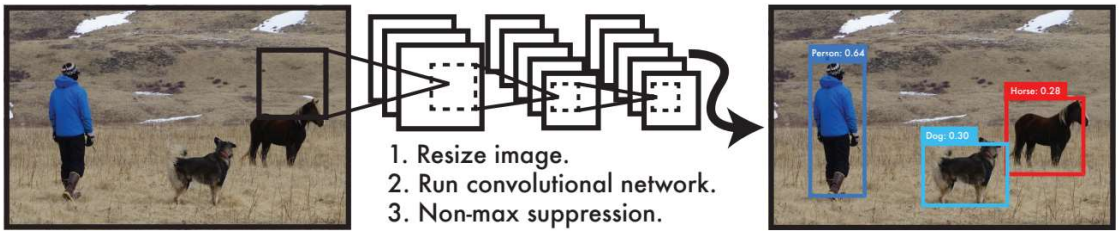
➢ Object Detection:

Human glance at any scene or image can instantly recognize what objects are in it, where they are, and how they interact. It is fast and accurate that allow us to perform complex tasks like driving, running, etc. Object detection is the most abundantly used technique in computer vision and image processing that deals with detecting the instances of semantics objects of a certain class like humans, cars, trucks, buildings, etc. in digital images and videos. Each object class has its own special features that can be exploited in object recognition and classification methods. For instance, all circles are round, humans have certain shape, etc. When looking for circles, objects that are at particular distance from a point (Center of circle) are sought. Similarly, when looking for cars and humans, objects that are in specific shape with respect to its basic geometry (rectangle in case of cars) are needed. [8]

To detect such features without defining them in prior, one can use the deep learning techniques based on convolutional neural networks (CNNs). Most recent approaches like Region-CNN use region-based methods to generate potential bounding boxes in the frame and classify them according to their feature accuracies. After classification, post processing is required to refine the boxes and eliminate erroneous detections. These tasks are slow and could not be optimized without losing substantial accuracy. To overcome this problem, a new approach called YOLO is proposed that uses single neural network to predict the boxes as well as class probabilities directly from full images in one evaluation.

➢ What is YOLO?

You Only Look Once – YOLO is an extremely fast state-of-the-art object detection approach that reframe the object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. It trains on full images and directly optimized detection performance. This architecture is based on a single convolutional neural network and has several benefits over traditional methods of object detection. It does not need the complex pipeline like traditional methods like DRM and Region-CNN. It is highly generalizable as it learns very generalized representations of the objects. As a result, there are multiple versions of YOLO like base YOLO than performs on 45 fps rate, fast YOLO that performs at 155 fps, YOLO 9000 that is trained to predict the 9000 classes, etc.



From the image above, we can see that image processing with YOLO is simple and straightforward. First it resizes the input images to 448 x 448 and then runs a single convolutional network on the image. In the end, it thresholds the resulting detections by the model's confidence.

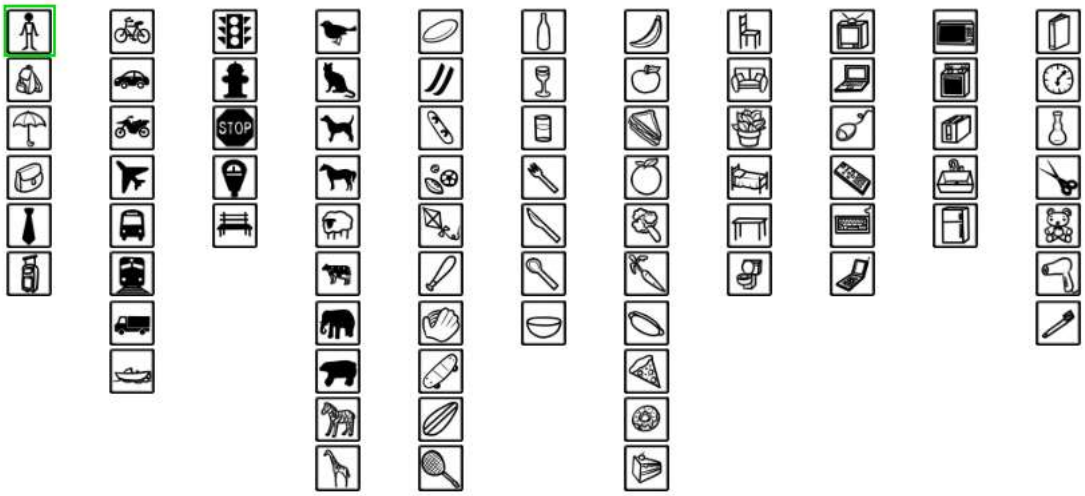
➢ Strategy to use YOLO

The YOLO network used in this system is pretrained model. It is trained on COCO dataset (Common Objects in Context) which is a large-scale object detection, segmentation and captioning dataset. It has 330K images with >200K labelled of 1.5 million object instances. This big dataset can be used to train the model for 80 object categories. [5]

While training, a single neural network is applied to the full image. This network divides the image into regions and predicts bounding boxes and probabilities for each region. These resulting bounding boxes are weighted by the predicted probabilities. Similarly, we apply this pretrained model in our system. While reading the input video stream frame by frame, YOLO gets the frame and identifies objects and their confidence. [6]

➢ Challenge

Applying the pretrained model to input video stream is basic part of the system. YOLO does its job to detect the object in very proper manner. Fully unleashed, it can get all the objects in the frame that it is pretrained to detect. These objects contain all the categories it was trained on. Some of them are given in figure below.



However, if our system goes on predicting every object in the frame with all the resources available to it, it will surely slow down the process as well as the purpose of the system. During driving any vehicle, driving assisting system or the driving system itself should be robust and extremely fast with minimum lag in the response. YOLO is such a big neural network with that many categories of objects, it is challenging to use this network on robust response system. We need to come up with predefined assumptions about the parameters and process outcome that can help it to minimize the response time as possible it can.

➢ Solution

YOLO provides the very general object detection of things that we generally see around us. However, as the environment of car is limited to few objects like different types of vehicles, human pedestrian walking with some animal or cart, animal or person crossing the road, unwanted objects like waste tire on road, other people with vehicle and passengers, signal lights and road signs, etc. Below are some results of YOLO.

From above images, we can see that YOLO is able to detect these relevant objects from image frame. Out of those, we can ignore some categories like bags, and other accessories that people wear. This will also sum up in the optimization of the process.

Clearly it is obvious that to improve the performance, we must reduce the detecting categories to limited number of classes that are relevant to the environment of vehicle and could possibly affect the normal driving of vehicle. To tackle this problem, we can make YOLO find for only limited list of classes that include all necessary object categories like above-mentioned things.

Lane Detection using LaneNet:

- Lane Detection

Modern cars are incorporating driver assistance feature which helps driver to drive safely and allows the car to position itself properly within the road lanes. Generally, a human glance can sense the lane width, how long the lane is and how much space is there to move forward or change the lane. Broadly speaking, this is crucial for any vehicle on the road. Lane detection is the most abundantly used in driver assisting system with advanced vehicle following task, to alert the driver if the vehicle begins to move out of its lane unless the turn signal is on (lane departure warning system LDWS), align the vehicle at the center of the lane (lane centering) or guiding when vehicle is parking and un-parking in lot. Lane detection looks easy because of insights like white lane markers that are already present on the dark road. However, it is very difficult to determine the markings on various types of the road by computer. These difficulties arrive from the lighting arrangements, shadows of trees or building like structures, occlusion by another vehicles, surface of road and sometimes the different marking system of lanes as well.

Traditionally, lane detection methods rely on the combination of hand-picked features and heuristics, Hough transform and canny edge detector to detect lane lines from real time camera images fed from the dash camera of vehicle. It pre-processes the images by perspective transformation, then consider the region of interest (viz. trapezoidal shape in front of hood in image) and convert that to grayscale and apply Canny edge detection and Hough transformation. After all such processes, we can make computer see the lanes only till the road markings are visible and these methods are computationally expensive and prone to scalability. Some modern deep learning methods like CNN with RANSAC algorithm can detect the end to end lanes that are pre-defined and fixed number like ego-lanes for particular road and could not cope up with lane changes. To overcome these limitations, the new method is proposed that can cast the lane detection problem as an instance segmentation problem. In this, each lane forms its own instance – that can be trained end-to-end. Further, apply the learned perspective transformation to fit the lane in robust way against the road plane variation and cope up with lane change by vehicle.

➢ What is LaneNet and H-Net?

The LaneNet is the branched, multi-task architecture to cast the lane detection problem as an instance segmentation task, that handles lane changes and allow the inference of an arbitrary number of lanes. It combines the benefits of binary lane segmentation with clustering loss function designed for one-shot instance segmentation. At the end, it outputs dense, pixel by pixel lane segments that disentangles into different lane instances.

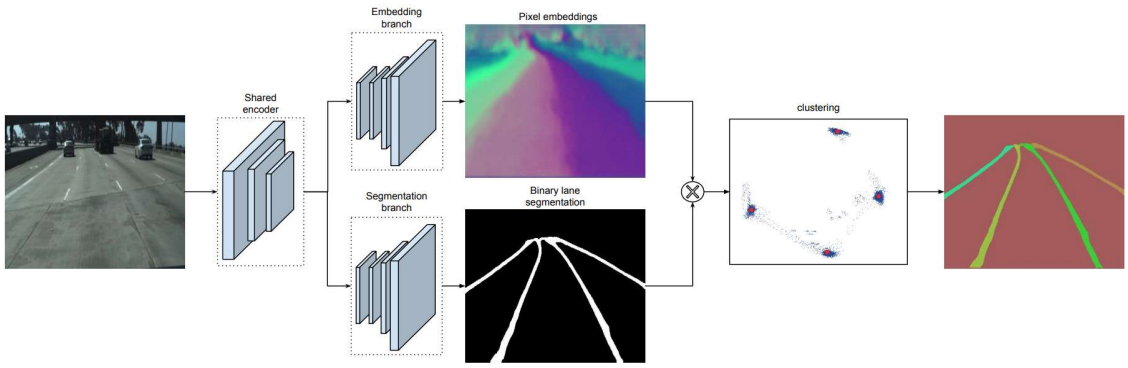
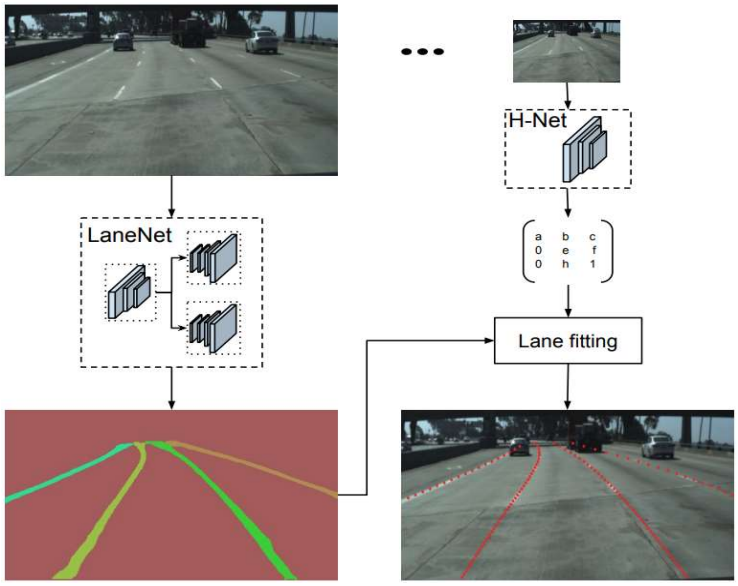


Fig. 2. LaneNet architecture. It consists of two branches. The segmentation branch (bottom) is trained to produce a binary lane mask. The embedding branch (top) generates an N-dimensional embedding per lane pixel, so that embeddings from the same lane are close together and those from different lanes are far in the manifold. For simplicity we show a 2-dimensional embedding per pixel, which is visualized both as a color map (all pixels) and as points (only lane pixels) in a xy grid. After masking out the background pixels using the binary segmentation map from the segmentation branch, the lane embeddings (blue dots) are clustered together and assigned to their cluster centers (red dots).



From above image: LaneNet outputs a lane instance map and these lane pixels are transformed using transformation matrix outputted by H-Net which learns a perspective transformation conditioned on the input image.

H-Net is the network with custom loss function that given the input image estimates the parameters of a perspective transformation H (matrix) that allows for the lane fitting robust against road plane change either uphill or downhill slope. It estimates the parameters of an ideal perspective transformation, conditioned on the input image which is the transformation in which the lane can be optimally fitted with a low-order polynomial (2^{nd} or 3^{rd} order). H has 6 degree of freedom: H = [[a b c], [0 d e], [0 f 1]] where 0 are placed to enforce the constraint that horizontal lines remains horizontal under the transformation.

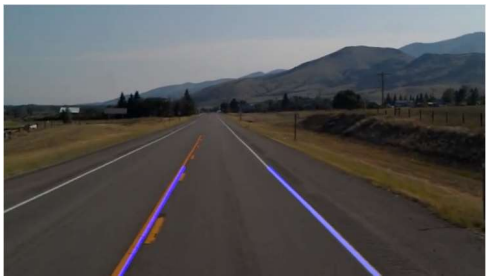
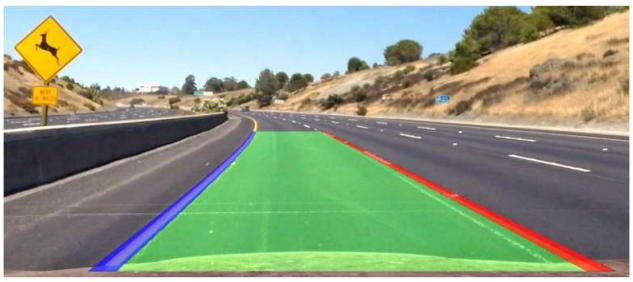
➢ Strategy to use LaneNet and H-Net

The LaneNet and H-Net networks used in this system is pretrained model. It is trained on the tuSimple (self-driving truck startup) lane dataset that contains video data with good and medium weather conditions, different daytime, 2-lanes to 4-lanes highway roads, and different traffic conditions. On each image, the ego lanes and left/right lanes are annotated. When changing the lanes, a 5^{th} lane can be added to avoid the confusion. It contains 3626 video clips and annotated (polylines on lane) frames for training and 2782 video clips for testing. [7]

The LaneNet is trained with an embedding dimension of 4 with Variance term Lvar=0.5 and Distance term Ldist=3. It is used while pixel embedding of the same lane cluster together, forming unique clusters per lane. Here, Lvar applies a pull force on each embedding towards the mean embedding of a lane and Ldist pushes the cluster centers away from each other. The input images are rescaled to 512*256 and the network is trained using Adam with batch size of 8 and learning rate of 5e-4 until convergence. H-Net is trained for a 3^{rd} ordered polynomial fit with image size of 128*64. It is trained using Adam with a batch size of 10 and learning rate of 5e-5 until convergence. Similarly, we apply this pretrained model in our system. While reading the input video stream frame by frame, LaneNet and H-Net system gets the frame and identifies the lanes from vehicle to the end of the road.

➢ Challenge

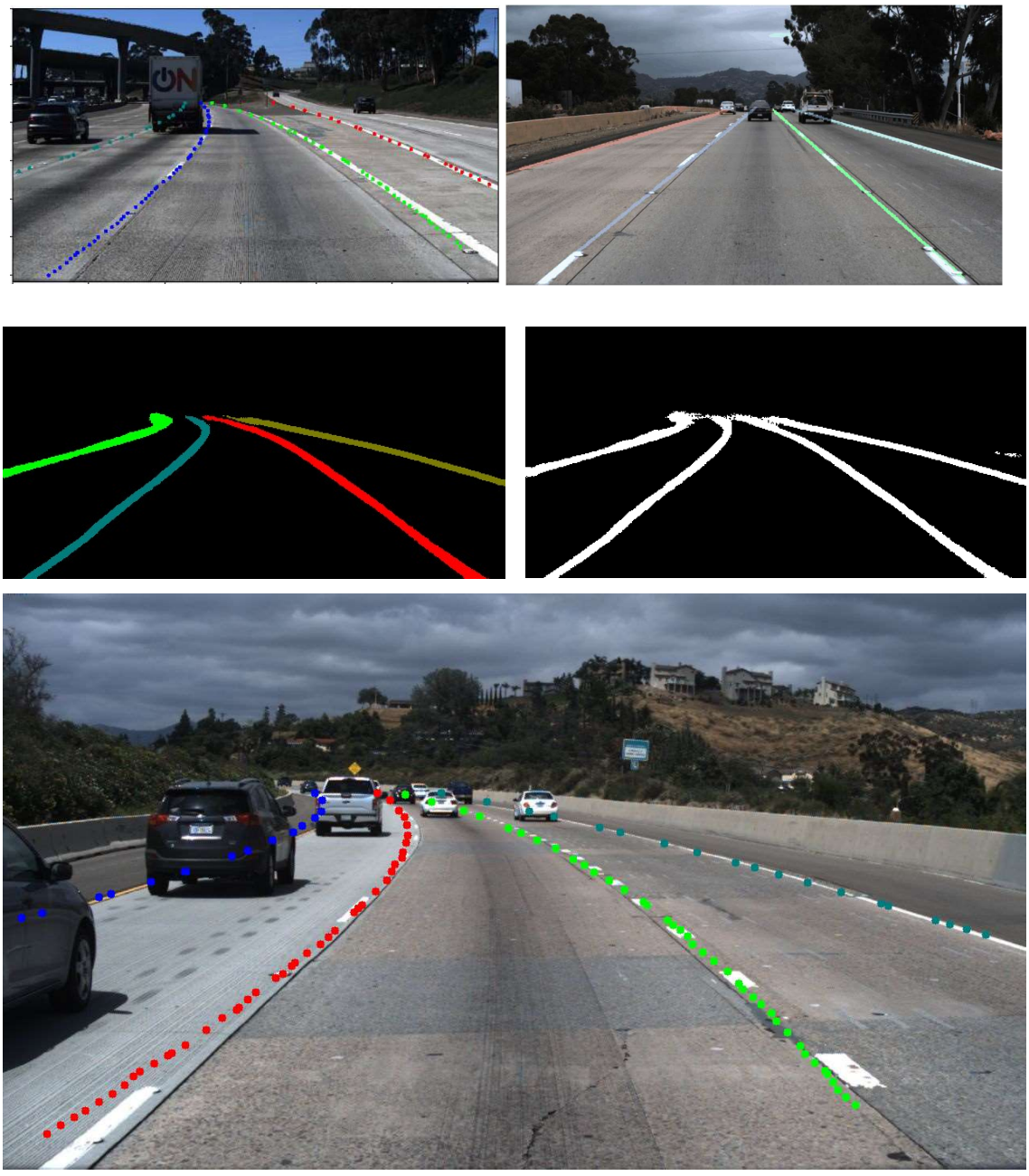
The traditional systems only detect the safe or danger zone behind or front of car up to some meters and could not be used to predict the whole environment front or back side of the vehicle. Some new deep learning systems detect the end to end lane, but their spread is limited to ego lane only which is not good for the driver assisting system which rely on the entire environment of the road, especially number of neighbor lanes and empty spaces on them. Some outputs of traditional lane detection are shown in images below.

On the other hand, to fully assists the driver to decide whether to change the lane or not, to speed up or speed down, to take over or not, we need the end to end lane detection technique that predicts the current lane from the vehicle till the road ends as well how many lanes are there.

➢ Solution

LaneNet provides the end to end lane detection from the scene that we generally get from dash camera of the vehicles. The span of lanes and number of lanes obtained in the output of LaneNet helps to take the good decision of dynamic lane changing and maintaining the speed based on the curve of road. This system only cares about the lanes and no other clutter in the output that makes it elegant to combine with other systems in driving assistance. Below are some results of LaneNet.



From above images, we can see that LaneNet is able to detect all lanes from the image frame. From those, we can further take decision to analyze the empty lane where the vehicle could move or speed up or slow down. The minimum and clean outcome helps in the process optimization.

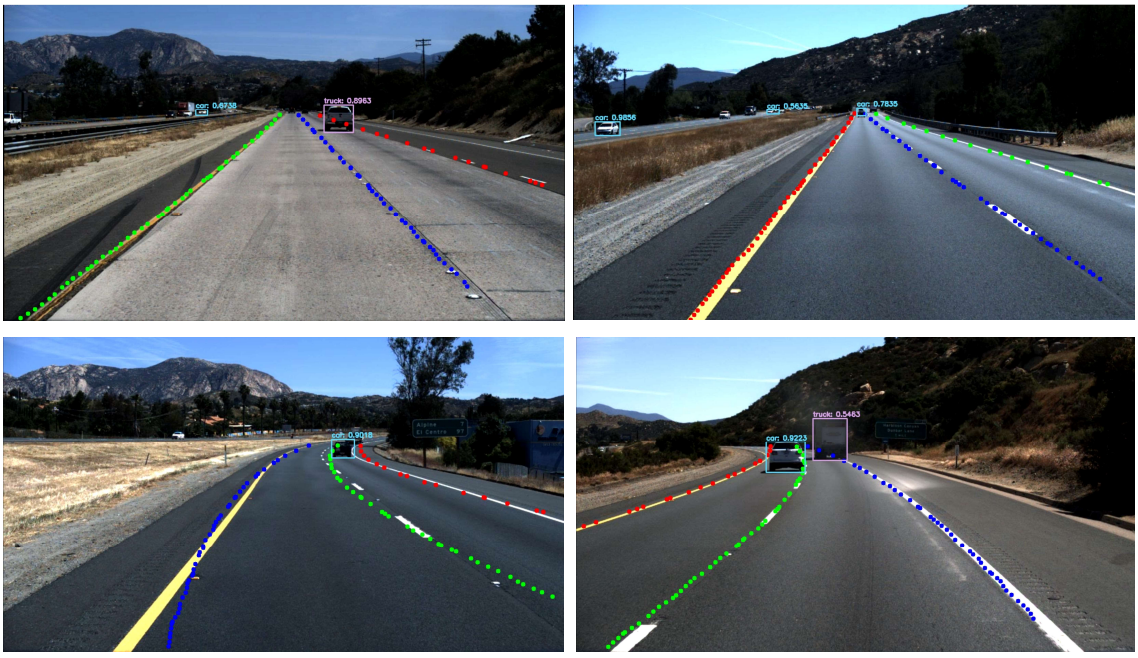
Combining YOLO and LaneNet:

We have tried to combine these two models to identify the lanes along with the obstacles in the current frame. The combined result of these two models will benefit in the driver assist technique in order to provide lane assistance to the driver. It will also be useful to identify the empty lanes from the current frame using which we can make predictions about the faster lanes to improve the efficiency of the roads.

The LaneNet and YOLO models downloaded from the GitHub repository mentioned in the references were able to handle the image input only. Whereas, considering the real-world scenario, it is a better idea to pass the video input to the model and analyze the results. Hence, we have created a short video from the using the images from the TuSimple Dataset. This dataset consists of multiple short clips containing 20 frames each. We have used this short video of length 2.46 minutes, containing 5000 frames, as an input for our model.

We have written a python program to read the video file frame-by-frame. Because of the limitations in the resources, we have decided to test our model on the first 300 frames of the video. In this python program, each video frame is passed as an input to the lane detection model. This model returns an image with the identified lanes in the current frame. Similarly, we pass the current input frame as an input to the YOLO object detection model. Afterwards, we write the results of the YOLO model on the output image of the lane detection model. In this way we combine the results of both the models. Finally, we write this image, containing detected objects and lanes, on the video output stream.

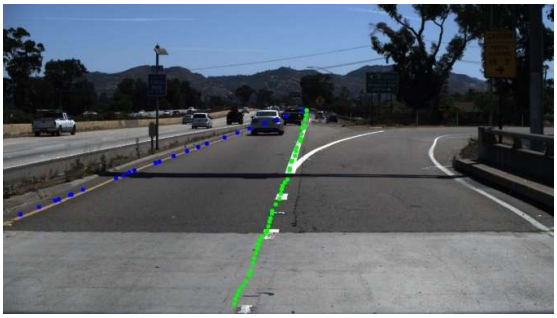
We are sharing the input video used for the testing purpose along with the generated output video in the submission folder. The input video and output video can be found in the test_video and OUT directories respectively. Few of the screenshots of our final model are as shown below.



Applications and Future Scope:

This project has a direct application in the driver assistance techniques. Using the insights about the lanes and the objects in front of the car, obtained the model, autonomous driving cars can behave in a more robust way in the diverse traffic scenarios. It gives the end to end lanes mapping and it also identifies the obstacles in the frames. This knowledge can be used to make better decisions considering the safety and efficiency of the travelers. Hence, it would be helpful to make the road travelling safer. This model can also be used for detecting the faster lanes and providing an assistance about the lane changes.

Although the performance of the model seems to be good in most of the scenarios, the model seems to be failing in some of the cases. The lane detection algorithm seems to be failing in the situations where the lane gets split in two. In this kind of situation, the model seems to struggle to identify the lane clusters and fit the curve through the identified clusters. The images shown below demonstrates this issue. In the first image, the model ignores the lane turning left. In the second figure, model struggles to fit the curve.




The functionality should be added to the model to handle this kind of functionality. As this is one of the complicated scenarios, it requires special attention. Also, the model seems to struggle to identify the lanes on the images in which camera angle is different. i.e., model identifies the lanes correctly if the tusimple dataset is used in which camera is focused completely on the roads and there is no other detail present in an image. But it fails to produce results on images in which camera angle is different, other details like car dashboard is present in the image, etc. The model needs more robust training in order to handle such cases.

Conclusion:

In this project, we worked on an image dataset containing various images of the roads. From these images we tried to identify the lanes and objects. While doing this, we learnt about the YOLO model for object detection in depth, which is one of the fastest performing object detection models. Also, we learnt about the LaneNet model to identify the lanes in the image. We studied the renowned papers published on the YOLO model and End-To-End Lane Detection Using Image Segmentation to understand the concept in depth. We practiced our skills to pass the video input to these models, perform operations on each frame of the video and produce a video output. We tried to understand the codes for the lane detection and object detection model. We explored the tensorflow library in order to understand the code for the lane detection model. Finally, we combined the results of these two models to make a complete system for lane and obstacle detection (in our case any object in the road view can be considered as an obstacle). Overall, upon completion of this project, we are more confident about working on the computer vision project.

Finally, we would like to thank Dr. Thomas Kinsman to provide us guidance and insights for multiple image and video processing methods and computer vision paradigm. It was a great opportunity to learn about neural network aspects in the computer vision field.

References:

[1] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, You Only Look Once: Unified, Real-Time Object Detection: https://arxiv.org/pdf/1506.02640v5.pdf

[2] Davy Neven, Bert De Brabandere, Stamatios Georgoulis, Marc Proesmans, Luc Van GoolTowards, Towards End-to-End Lane Detection: An Instance Segmentation Approach: https://arxiv.org/pdf/1802.05591.pdf

[3] Github link to object detection using YOLO: https://github.com/kalyco/yolo_detector

[4] Github link to lane detection code: https://github.com/MaybeShewill-CV/lanenet-lanedetection

[5] COCO Dataset: http://cocodataset.org/#home

[6] YOLO strategy: https://pjreddie.com/darknet/yolo/

[7] tuSimple Lane Detection Challenge: http://github.com/TuSimple/tusimple-benchmark/

[8] Object Detection Wiki: https://en.wikipedia.org/wiki/Object_detection