Published by :
http://www.ijert.org
International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 05, May-2022

Lane and Object Detection for Driver Alertness System using Deep Learning Algorithm

Mrs. Snehal Date-Thube
Department of Technology, SPPU, Pune,
Maharashtra.

Abstract: **Lane identification is a challenge. For many decades, it has drawn the attention of the computer-vision world. Lane detection is a multifunctional detection problem which has become a major challenge for computer vision and ML techniques. While several ML methods are used to detect lanes, they are used primarily for classification rather than feature detection. But modern methods of ML can be used to recognize features that are high in recognition and have been effective in feature detection tests. However, in the reliability and precision of lane detection, these methods have not been completely implemented. In this method, we are suggesting a new way of approaching it. We introduce a new DL-based approach to detect the lane. In addition to lane detection, a lane departure detection system is also implemented. The vehicle detection algorithm using YOLO is also implemented to keep away from the accident for the autonomous vehicle system. The proposed approach achieved promising results for urban as well as rural roads.**

Index Terms**:** ***Advance driver assistance system, Deep learning, Convolution Neural network.***

I. INTRODUCTION

Human beings, when fully attentive, do quite well at identifying lane line markings under most driving conditions. Inherently the machines are not good at doing the same. However, humans have a drawback that they are not always vigilant (whether it's because of adjusting the radio, talking to another person, being sleepy, being under the influence, etc.). In contrast, a computer is not subject to this downfall. As such, if we can train a computer to be as good as a human when it comes to detecting lane lines as it is already considerably easier to pay full-time attention, the machine will take over this job from the human driver. Using DL, we can train a model that can be more robust and faster than the original computer vision-based model. The model will be based on a neural network architecture called a "Convolutional Neural Network," which is known to perform well on image data. It is a great architecture candidate since we will feed the model frames from driving videos to train it. CNN's work well with images as they look first for patterns at the pixel level (groups of pixels around each other), progressing to larger and larger patterns in more expanded areas of the image.

Many researchers have worked and are working on creating and improving several strategies with advanced driving assistance systems in intelligent transportation systems that can ensure road safety and congested traffic conditions. Road accidents are the world's principal causes of sudden death. Even though we have many good and advanced techniques in this world, we are leftover with something to make it better than before. There are chances from different angles. The road lane detection and object detection is also the other important way that we can improve safety inroads.
So in our research, we provide a way to improve lane detection and object detection in vehicles than then rest of the other categories that may avoid the many road accidents. Lane should have to be detected even with the external factors in consideration. The object detection will provide driving person confidence even in the different lighting and different environments situations by improved techniques to detect the objects. I thought you could provide safety on the roads to achieve a safer environment and in traffic-congested conditions.

II. RELATED WORK

In this section, Lane detection and tracking algorithms are discussed. This section also investigates the best lane detection and tracking algorithms that can be selected for specific road conditions.

Yim and Oh [1] developed a three feature-based lane detection algorithm. The features used are starting position, orientation, and intensity value. In the initial step, a Sobel operator is applied to get the edge information. The lane boundary is represented as a vector comprising of the three features. The current lane vector is calculated based on the input image and the previous lane model vector. Two windows, one for each, are used for left and right boundaries. Assuming N pixel in each horizontal line, N lane vector candidates are generated. The best candidate is selected based on the minimum distance from the previous lane vector using a weighted distance metric. For equalization, each feature is assigned a different weight. Then a lane inference system is used to predict the new lane vector. If the road width changes abruptly, the current vector calculated is discarded and the previous one is taken as the current vector.

A lane detection approach for the urban environment is proposed by Sehestedt et al. [4]. Since the lane markers are not visible due to wear and tear occlusions, and due to complex road geometry, a weak model is used for detecting lane markers. In the inverse perspective mapped image, the particle filter is applied from the bottom row to the top. The filter is tuned in such a way to track multiple lanes.

Kim[6] developed an algorithm for lane detection and monitoring that can handle difficult scenarios such as fading lane markers, lane curvatures, and split lanes. Initially, gradient detector and an intensity bump detector are used to remove non-lane markers. The remaining samples are added to Artificial Neural Networks (ANN) for lane detection. The pixels found in the lane markers are grouped using cubic splines. Hypotheses are generated from segments of a random set of lines. The RANSAC algorithm assists in invalidating the assumptions. Lane monitoring is used for particle filtering.

Daigavane et al. [12] developed a lane detection method based on ant colony optimization (ACO) is proposed. Initially, the input image is resized to 255 X 255 pixels for reducing the computation time. The three-channel RGB image is converted to a single-channel grayscale image. Then, the median filter is used to filter out the noise and to preserve the edges. Next, the edges are detected using a Canny edge detector. The output binary image is given as input to ant colony optimization. It generates additional edge information that is lost in Canny edge detection.

III. METHODOLOGY

The Methodology is using deep learning, we can train a model that can more robust, and faster, than the original computer vision-based model. The model will be based on a neural network architecture called a “Convolutional Neural Network (CNN)" which is known to perform well on image data. This is a great architecture candidate since we will feed the model frames from driving videos to train it. CNN's work well with images as they look first for patterns at the pixel level (groups of pixels around each other), progressing to larger and larger patterns in more expanded areas of the image.

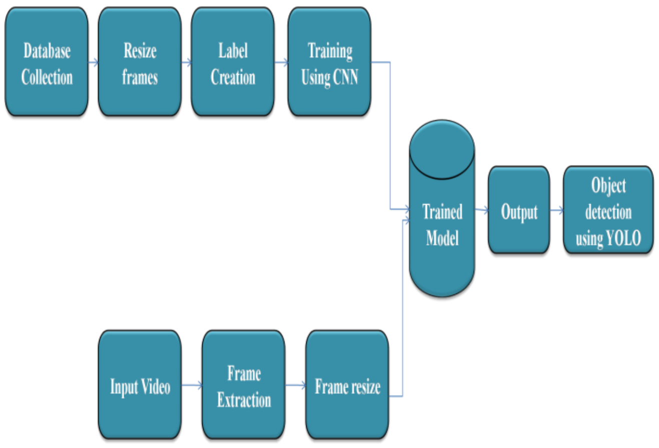


Fig. Block Diagram of Lane Departure Detection

A. Datasets and Inputs

The datasets plan to use will be image frames from the video. The smartphone creates the database. The database is constructed in 720p horizontal / landscape mode with 720 pixels on the y-axis and 1280 pixels on the x-axis, 30 frames a second. The frames are resized to 25 percent of the original size to reduce the total processing needed. To train my model, it will require some manual pre-processing. As such, the polynomial coefficients for a second-order polynomial will be determined for each line (the three outputs per line, plus two lines, meaning that each image will have six "labels") in each image. It is also the intention to use the undistorting and perspective transform techniques of Computer Vision (CV) to create these labels for image training. The hope is that in the fully trained model, the model would not need such techniques for new pictures. The dataset would likely end up very large, depending on the size of the neural network used. At 30 fps, 9,000 photos will be just 5 minutes of the film, which would be a lot of labeling. With our labeling strategy, we will first try a subset of the data to ensure that the method appears to work before labeling on the full dataset. We will try to use a combination of both straight lines and different curved lines, as well as different conditions (night vs. day, shadows, rain vs. sunshine) to help generalize the model overall. We hope these would help to address more of the actual problems drivers see every day.

B. Training using CNN

Before the training process, the basic of CNN needs to understand, and it is explained below.

a. Convolutional Layer

The convolutional layer performs the central building block of a Convolutional Network, performing much of the heavy computational lifting. The Convolution layer's primary objective is to extract features from the input data, which is an image. Using a collection of learnable neurons convoluted the input image. It generates a feature map or activation map in the output image, and afterward, the feature maps are fed into the next Convolutional layer as input data. It is interpreted in mathematical terms as

$$G[m,n] = (f * h)[m,n] = \sum_j \sum_k h[j,k] f[m-j, n-k]$$

where the input image is denoted by f and our kernel by h.

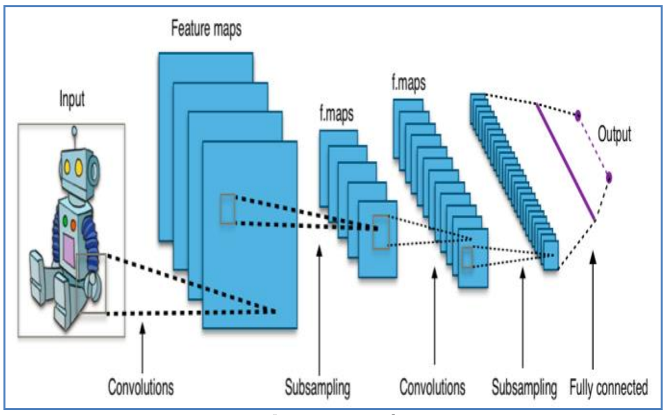


Fig .Architecture of CNN

b. ReLU Layer

ReLU is a non-linear operation which involves the rectifier-employing components. It is an element-wise operation, which means it is applied per pixel and reconstitutes all negative values by zero in the characteristic diagram. To understand how the ReLU works, we suppose there is an input of neurons given as x, and the rectifier is defined as

$$f(x) = max(0, x)$$

c. Pooling Layer

The pooling layer decreases the dimensionality of each activation map but still has the most relevant information available. The image's input is broken down into a collection of non-overlapping rectangles. Each area is down-sampled such as average or maximum by a non-linear operation. This layer achieves stronger generalization, quicker convergence, stable translation, and distortion.

The layers of max Pooling are fairly easy and do not learn themselves. They simply take a certain region of k×kk×k and generate a single value, which is the limit in that region. For example, if their input layer is an N×NN×N layer, they will then output an Nk×NkNk×Nk layer because each k×kk×k block is reduced to just one value via the max function.

C. Training details

Although we had made a CNN previously that ended in fully-connected layers, we had never before made a fully CNN, and there were some challenges in getting the underlying math to work for my layers. Unlike in the forward pass in normal Convolution layers, Keras's Deconvolution layers flip around the backpropagation of the neural network to face the opposite way and therefore need to be more carefully curated to arrive at the correct size (including the need to specify the output size). We chose to make my new model a mirror of itself, with Convolutional layers and Pooling in slowly decreasing in size layers, with the midpoint switching to Upsampling (reverse-pooling) and Deconvolution layers of the same dimensions. The final deconvolution layer ends with one filter, which is because only wanted a returned image in the 'G' color channel, as it was drawing predicted lanes in green (it later is stacked up with zeroed-out 'R' and 'B' channels to merge with the original road image). Choosing to input 80x160x3 images (smaller images were substantially less accurate in their output, likely due to the model being unable to identify the lane off in the distance very well) without grayscaling (which tended to hide yellow lines on light pavement), we also normalized the incoming labels by just dividing by 255 (such that the labels were from 0 to 1 for 'G' pixel values). The CNN architecture used in the proposed system is shown in Fig. 3.8. It consists of one layer batch normalization initially, seven convolutional layers with ReLu activation function, three pooling layers, seven deconvolutional layers with ReLU, and three upsampling.

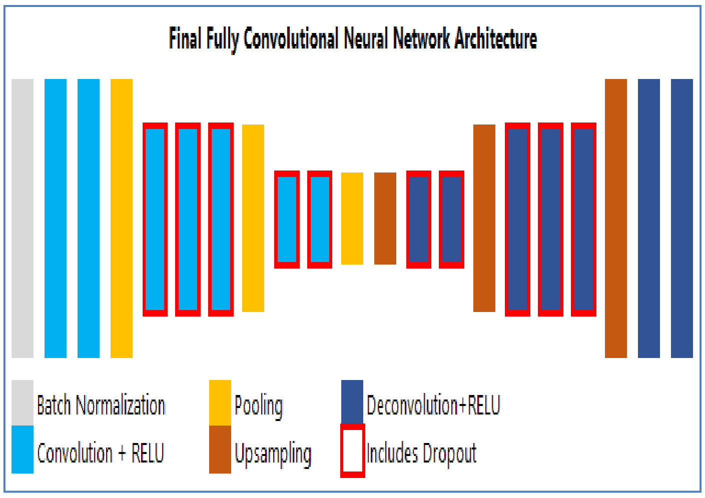


Fig. Convolution Neural Network architecture of the proposed system

After 20 epochs, the model finished with MSE for the training of 0.0046 and validation of 0.0048, which was significantly lower than any previous model's had tried. First tried the trained model against one of our videos, one of the hilly and curved roads for which the model had potentially seen up to 20% of the images for, although likely much less from the image statistics earlier, had to throw out a large portion of the images from these videos, so even though ran it on one in five images, the model probably only saw 5-10% of them. Fascinatingly, the model performed great across the entire image, only losing the right side of the lane at one point when the line became completely obscured by leaves. The model performed near-perfectly even on a lot of the areas knew had previously had to throw out because my CV-based model could not appropriately make labels for them.

IV. TESTING

In the testing phase, the testing video is provided through employing recorded video or live feed from a web camera. The frames are extracted from the video and resize into 640x480 size. Then the image is feed to the training model and gets a result of lane detection. The detected lane is highlighted with green color.

V. OBJECT DETECTION USING YOLO ALGORITHM

The Block diagram of Vehicle detection from the scene is shown in Fig.

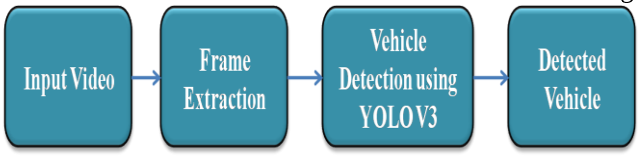


Fig. Block diagram of the vehicle detection using YOLO v3 algorithm

This section describes the object detection methods used in this study. The design of the system for the identification of highway vehicles utilized the YOLOv3 network. The YOLOv3 algorithm continues the basic concept of the YOLO algorithms from the first two generations. The input image features are extracted using CNN. The input image is split into 13*13 grids according to the scale of the function diagram, such as 13x13. The object label box centre is in a grid unit which is responsible for predicting the object is in charge. The Network Configuration adopted by YOLOv3 is called Darknet-53. This structure improve the maximum convolution approach and replaces the previous version of the directly linked neural network with the residual one. The branch is used for the direct connection of the input to the network's deep layer. Direct residual learning guarantees the quality of information about image characteristics, simplifies the difficulty of training, and increases the network's overall detection performance. In YOLOv3, for one item, each grid unit will have three bounding boxes of various scales. The final predictive result will be the candidate box, which has the highest overlap area with the annotated box. The YOLOv3 network also has three performance scales, and the three branches of the system are finally combined into one another. Shallow features are used to detect small items, and deep features are used to detect large objects; this enables the network to identify objects with differences in size. The detection speed is small, and the accuracy of the detection is high. Traffic scenes taken from highway surveillance footage have excellent YOLOv3 network adaptability. Finally, the network outputs the object's position, trust, and category.

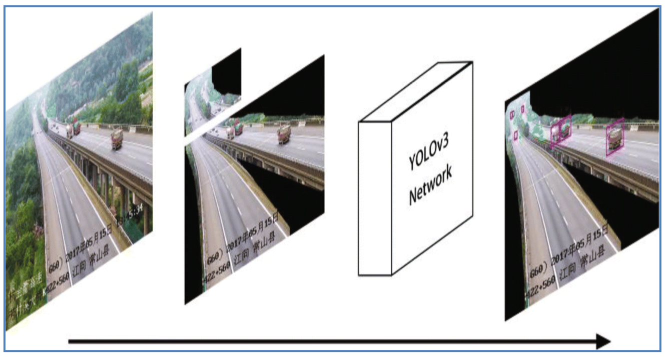


Fig.A segmented image was sent to the detection network, and detected results are merging. (Green, blue, and fuchsia boxes are labeled to indicate the "car," "bus," and "truck" regions, respectively.

The project is to develop and test a computer vision system for real-time vehicle detection for giving notification to the system, a level 2 of automation for cars was used to verify the result of the developed algorithm and the system.

Data collection: The data will be collected, and will dictate how accurate the model is based on the quantity and quality. The first step will be to start to look for annotated object detection datasets for specific objects like vehicles and pedestrians, and this is where RQ1.2 will be answered. The Open Images Dataset V4 is one of the datasets that will be selected because of the huge amount of data it contains, 15.4M bounding boxes for 600 object classes. After downloading the images, the data will be analyzed to see if there exists any negative data, and maybe images that are not labeled correctly.

Data preparation: After the collection training data, the next step will be to proceed to data preparation, loading the data to the appropriate location, and then preparing for ML training. First, the data will put together; then, the order is randomized because the order of the data should not affect what the model learns. If the dataset consists of negative data, it will be discarded as it can impact the performance.

Training: When starting an ML process, several steps need to be familiar before the start. First of all, it is choosing an algorithm that is suited to solve a specific problem. Then the data used to train the model must be labeled correctly, as it is very common that the objects can be falsely identified. Overfitting is an issue that can occur when a model that trains the training data too; well, that slowly gives negative impacts on the performance of the model. A way to work against overfitting is by adding more data. When training a model based on the new data, training aims to increase the performance, and a way to do that is to use transfer learning; for instance, if a model is trained to detect trucks, it could be useful to help train another model on detecting cars. We even improved the dataset for this project as the dataset that was downloaded considered images that were taken from different angles and bounding boxes that did not catch the object well.

Parameter tuning: The input layer of the model will be altered as well as the batch sizes which will be set to 64, and the number of training steps present in one batch will affect the model performance. Momentum will be set to 0.9, learning rate will be set to 0.001, and it is how quickly the network replaces the concepts it has learned up until now with the new ones, training steps will be (8000, 9000), and this is the number of steps based on the batch size it takes to complete the processing of the whole dataset. It is to the tuned model for improved performance of accuracy and faster training.

The output samples of some real-time scenarios are shown in Fig.

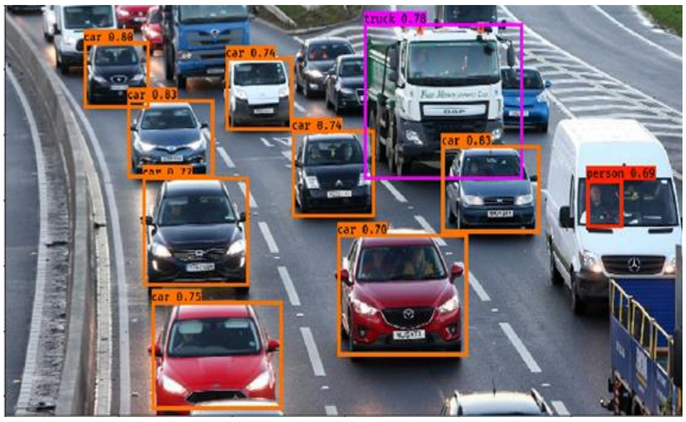


Fig. The output of vehicle detection using YOLO V3.

VI. RESULTS

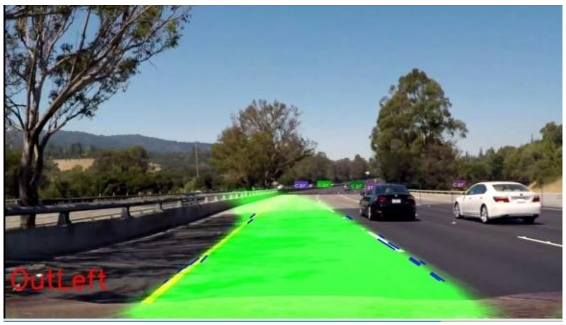
The proposed Lane departure detection system, is implemented using the OpenCV library with the python language. The Keras and TensorFlow libraries are used to implement DL algorithms. The proposed algorithm is tested on real-time videos. The results are presented in qualitative and quantitative ways.

i. Qualitative Analysis

The aim of this analysis is a comprehensive description. There is no effort to assign frequencies to the linguistic features recognized in the data and unusual phenomena get the same amount of coverage as more common phenomena. Qualitative analysis allows excellent distinctions to be made, because shoehorning the data into a limited number of classifications is not important. Ambiguities, which are intrinsic in human language, can be recognized in the analysis. Qualitative analysis is the pictorial and non-statistical demonstration of the research. The results of the proposed system implementations have been shown below in Fig. 3, and Fig. 4.

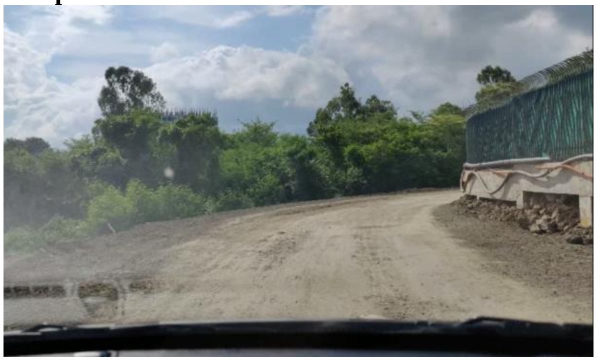
1. Urban Roads
Sample:

(a)

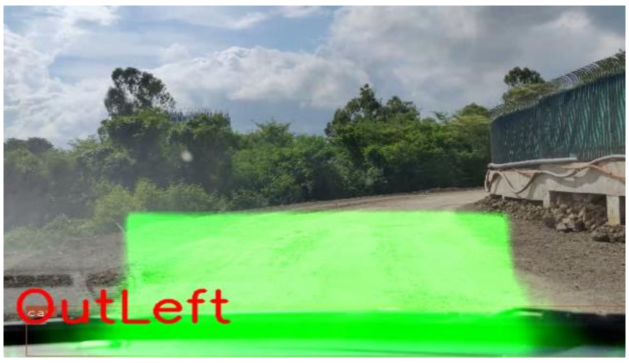


(b)

2. Rural Roads
Sample:



(a)

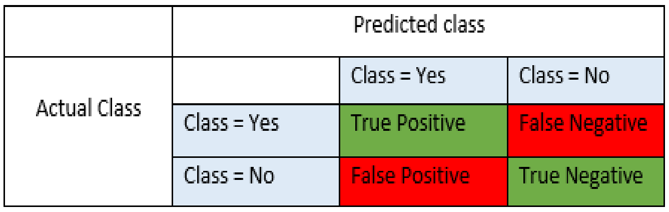


(b)

Fig. Qualitative Analysis (a) Input Image (b) Output Image

ii. Quantitative Analysis

In quantitative research, the approach is used to categorize features, count them up, and even build more complex arithmetical models to explain what is observed. Findings can be comprehensive to a larger population, and straight comparisons can be analyzed between the two classes, so long as valid sampling and implication techniques are used. Thus, this analysis allows us to determine the concept of the real reflections of the variety in the classes.
T.P. and T.N.s are the observations that are correctly predicted and therefore shown in green. We want to minimize F.P.s and F.N.s, so they are shown in red color. These terms are a bit confusing. So let's take each term one by one and understand it fully.

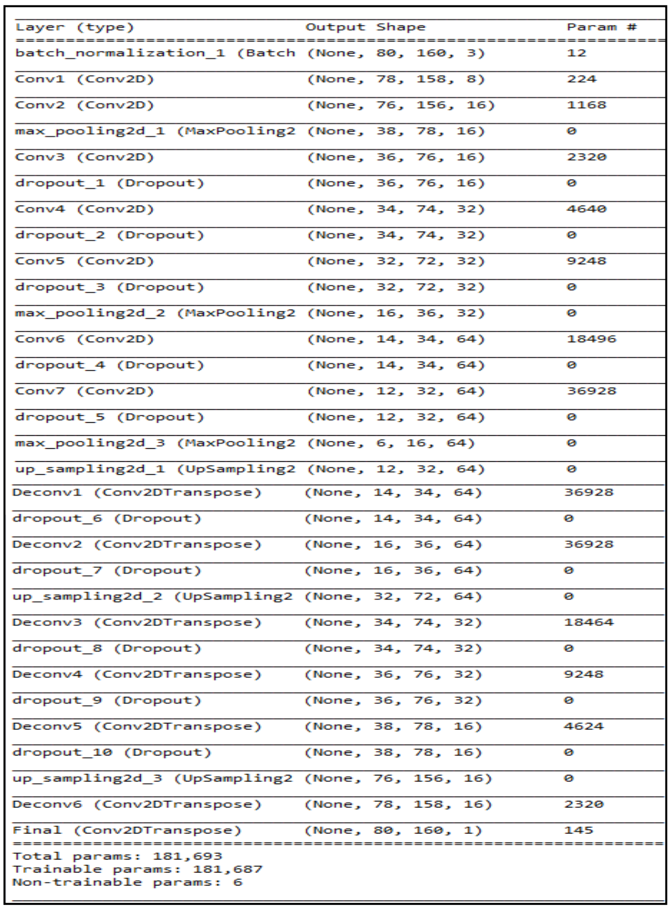


		Predicted class	
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

Fig Results of shot boundary detected

T.P. and T.N.s are the observations that are correctly predicted and therefore shown in green. We want to minimize F.P.s and F.N.s, so they are shown in red color

The Network Parameters for the CNN based training algorithm is shown in Fig.



```
Layer (type)                     Output Shape              Param #
==================================================================
batch_normalization_1 (Batch (None, 80, 160, 3)         12
Conv1 (Conv2D)                   (None, 78, 158, 8)        224
Conv2 (Conv2D)                   (None, 76, 156, 16)       1168
max_pooling2d_1 (MaxPooling2 (None, 38, 78, 16)         0
Conv3 (Conv2D)                   (None, 36, 76, 16)        2320
dropout_1 (Dropout)              (None, 36, 76, 16)        0
Conv4 (Conv2D)                   (None, 34, 74, 32)        4640
dropout_2 (Dropout)              (None, 34, 74, 32)        0
Conv5 (Conv2D)                   (None, 32, 72, 32)        9248
dropout_3 (Dropout)              (None, 32, 72, 32)        0
max_pooling2d_2 (MaxPooling2 (None, 16, 36, 32)         0
Conv6 (Conv2D)                   (None, 14, 34, 64)        18496
dropout_4 (Dropout)              (None, 14, 34, 64)        0
Conv7 (Conv2D)                   (None, 12, 32, 64)        36928
dropout_5 (Dropout)              (None, 12, 32, 64)        0
max_pooling2d_3 (MaxPooling2 (None, 6, 16, 64)          0
up_sampling2d_1 (UpSampling2 (None, 12, 32, 64)         0
Deconv1 (Conv2DTranspose)        (None, 14, 34, 64)        36928
dropout_6 (Dropout)              (None, 14, 34, 64)        0
Deconv2 (Conv2DTranspose)        (None, 16, 36, 64)        36928
dropout_7 (Dropout)              (None, 16, 36, 64)        0
up_sampling2d_2 (UpSampling2 (None, 32, 72, 64)         0
Deconv3 (Conv2DTranspose)        (None, 34, 74, 32)        18464
dropout_8 (Dropout)              (None, 34, 74, 32)        0
Deconv4 (Conv2DTranspose)        (None, 36, 76, 32)        9248
dropout_9 (Dropout)              (None, 36, 76, 32)        0
Deconv5 (Conv2DTranspose)        (None, 38, 78, 16)        4624
dropout_10 (Dropout)             (None, 38, 78, 16)        0
up_sampling2d_3 (UpSampling2 (None, 76, 156, 16)        0
Deconv6 (Conv2DTranspose)        (None, 78, 158, 16)       2320
Final (Conv2DTranspose)          (None, 80, 160, 1)        145
==================================================================
Total params: 181,693
Trainable params: 181,687
Non-trainable params: 6
```

Fig. Network Parameter

After 20 epochs, the CNN model finished with MSE for the training of 0.0046 and validation of 0.0048, which was significantly lower than any previous model's we had tried. We first tried the trained model against one of our videos, one of the hilly and curved roads for which the model had potentially seen up to 20% of the images for, although likely much less from the image statistics earlier, we had to throw out a large portion of the images from these videos, so even though we ran it on one in five images, the model probably only saw 5-10% of them. Fascinatingly, the model performed great across the entire image, only losing the right side of the lane at one point when the line became completely obscured by leaves.

VII. CONCLUSION

In this approach, a lane detection algorithm for intelligent vehicles in urban road conditions and rural road conditions has been presented. Firstly the database is selected for rural and urban roads and its respective lane binary output. Then CNN model is used to train the image by hyper tuning the parameters. The proposed algorithm achieved the MSE for the training of 0.0046 and validation of 0.0048. In other parts of the project, vehicle detection from the real scene has been presented. The YOLOV3 algorithm is used to detect the vehicles on the road. The experiments show that the promising result for vehicle detection as well. The vehicle detection is helpful to avoid the accident of the autonomous vehicle.

One potential improvement to the model could be the use of a recurrent neural network (RNN). The current version of my model uses an averaging across five frames to smooth out any issues on a single frame detection, outside of the actual neural network itself. On the other hand, an RNN would be able to directly look at previous frames to learn that what was detected in a previous frame matters to the current frame. By doing so, it would potentially lose any of the more erratic predictions entirely. In the future, this algorithm can be implemented to improve the performance of the system.

VIII. REFERENCES

[1] Yim, Y. U., Oh, S. Y. (2003). Three-feature based automatic lane detection algorithm (TFALDA) for autonomous driving. IEEE Transactions on Intelligent Transportation Systems, 4(4), 219-225.

[2] E. Bellis and J. Page, National motor vehicle crash causation survey (NMVCCS), SAS analytical users manual, U.S. Department of Transportation, National Highway Traffic Safety Administration, Washington, DC, USA, Tech. Rep. No. HS-811 053, Dec. 2008.

[3] Y.U. Yim and S.- Y. Oh, "Three-feature based automatic lane detection algorithm (TFALDA) for autonomous driving," IEEE Trans. Intell. Transp. Syst. , vol. 4, no. 4, pp. 219-225, Dec. 2003.

[4] S. Sehestedt, S. Kodagoda, A. Alempijevic, and G. Dissanayake, "Robust lane detection in urban environments," in Proc. IEEE Intell. Robots Syst., Oct. 2007, pp. 123-128.

[5] M. Aly, "Real time detection of lane markers in urban streets," in Proc. IEEE Intell. Veh. Symp., Jun. 4–6, 2008, pp. 7-12.

[6] Z. Kim, "Robust lane detection and tracking in challenging scenarios," IEEE Trans. Intell. Transp. Syst. , vol. 9, no. 1, pp. 16-26, Mar. 2008.

[7] H.Y. Cheng, C.C. Yu, C. C. Tseng, K. C. Fan, J. N. Hwang, and B. S. Jeng, "Hierarchical lane detection for different types of roads," In Acoustics, Speech and Signal Processing (ICASSP), 2008 IEEE International Conference on pp. 1349-1352. IEEE, 2008.

[8] A. Assidiq, O. Khalifa, R. Islam, and S. Khan, "Real time lane detection for autonomous vehicles," in Computer and Communication Engineering, 2008. ICCCE 2008. International Conference on. IEEE,2008, pp. 82-88.

[9] A. Borkar, M. Hayes, and M. T. Smith, "Robust lane detection and tracking with RANSAC and Kalman Filter," Proceedings of the IEEE International Conference on Image Processing, pp. 3261-3264, 2009.

[10] C. W. Lin, H. Y. Wang, D. C. Tseng, "A robust lane detection and verification method for intelligent vehicles," In Intelligent Information Technology Application, 2009. IITA 2009. Third International Symposium on vol. 1, pp. 521-524.

[11] A. Borkar, M. Hayes, M. Smith, and S. Pankanti, "A layered approach to robust lane detection at night," in Proc. IEEE Workshop Comput. Intell. Vehicles Veh. Syst., 2009, pp. 51-57.

[12] P. Daigavane and P. Bajaj, "Road Lane Detection with Improved Canny Edges Using Ant Colony Optimization," in 3rd International Conference on Emerging Trends in Engineering and Technology (ICETET), pp. 76-80, 2010.
[13] C. Guo, S. Mita, and D. McAllester, "Lane detection and tracking in challenging environments based on a weighted graph and integrated cues," in Proc. Int. Conf. on IEEE/RSJ Intelligent Robots and Systems, Taipei, Taiwan, Oct. 2010, pp. 6643-6650.
[14] Y. C. Leng and C.-L. Chen, "Vision-based lane departure detection system in urban traffic scenes," in Control Automation Robotics Vision (ICARCV), 2010 11th International Conference on, 2010, pp. 1875- 1880.
[15] A. Borkar, M. Hayes, and M. T. Smith, "Polar randomized hough transform for lane detection using loose constraints of parallel lines." Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on. IEEE, 2011.
[16] V. Gaikwad and S. Lokhande, "Lane Departure Identification for Advanced Driver Assistance", IEEE Transactions on Intelligent Transportation Systems, pp. 1-9, 2014. Available: 10.1109/tits.2014.2347400.

AUTHORS PROFILE

Mrs.Snehal Date-Thube received her Bachelor of Technology (B.Tech.) from Walchand College of Engineering, Sangli, Maharashtra, India. Completed Master of Technology(M.Tech) from Department of technology, Savitribai Phule Pune University,Pune,Maharashtra,India.