Vehicle Speed Detection Using Multi-Branch Networks From Temporal Image Pairs

1st Rudramani Gyawali Singha
Dept. of Information Technology
RGIT, Mumbai University
Mumbai,India
rudramani.singha@gmail.com

2nd Manan Lad
Dept. of Information Technology
RGIT, Mumbai University
Mumbai,India
m616e616e.lad@gmail.com

3rd Gaurav Mukesh Shipurkar
Dept. of Computer Engineering
MPSTME, NMIMS University
Mumbai,India
gaurav.shipurkar4@gmail.com

4th Anish Rohekar
Dept. of Information Technology
RGIT, Mumbai University
Mumbai,India
rohekar.anish@gmail.com

5th Chaitanya Thombare
Dept. of Information Technology
RGIT, Mumbai University
Mumbai,India
indcthombare@gmail.com

6th Sunil Wankhade
Dept. of Information Technology
RGIT, Mumbai University
Mumbai,India
sunil.wankhade@mctrgit.ac.in

***Abstract*—Vehicle Speed Detection is important for traffic monitoring and surveillance. But very little research has been done on techniques that don't require specialist hardware or hand-crafted image processing method on video streams. This paper presents a novel end-to-end learning-based system that utilizes a multi-branched network architecture to estimate the speed of any vehicle from only two time-varied, unprocessed, images. The system also propose an optimized approach for generating a ground truth dataset by creating an estimation method for a 3D Rendering Engine. And further, propose a very computationally cheap method of extracting license plates by custom training a YOLOv4 tiny architecture in tflite format and Tesseract OCR. The proposed low-cost system is easily implementable in the existing infrastructure and provides a lightweight performance with computation time of $\leq$ 101.58 ms on average and accuracy of 97.3% in a wide range of applications.**

***Index Terms*—Vehicle Speed Detection, Multi-Branched Networks, YOLOv4 tiny custom training, OCR system, End-to-End Learning, Object Detection, Computer Vision, Deep Learning.**

I. Introduction

Population growth has resulted in the increased use of vehicles which has over time augmented the number of vehicle accidents in the real world because of over-speeding. Hence, vehicle speed monitoring is very essential and important for reducing these accidents. In order to advocate traffic monitoring and surveillance, the system proposes a multi-branched neural network for expeditiously and meticulously detecting the speed of the moving vehicles. To more simplify the work of the traffic monitoring control and to identify the car that is over-speeding, The system proposes a custom-trained YOLOv4 model[24] that specializes in detecting the bounding box of the license plate.

Presently, radar technology is used by traffic monitoring and surveillance systems. Radar technology employs radio waves for detecting the speed of the moving vehicle. An electromagnetic pulse is produced by the device that travels towards the moving vehicle and reflects off. The reflection results in a slight frequency shift that is the Doppler Shift. This shift in frequency is scrutinized which results in the actual speed of the moving vehicle. Other signals can cause an interruption in the radar signals giving rise to false positives, if there are several objects and mediums in the air the signal can be interfered leading to false results.

Contemplating the drawbacks of the previous methods, a method is designed for detecting the velocity and license plate of the vehicle that is over-speeding. For the data set, a 3D Render Engine is used to simulate a scene for generating high-resolution images. Each data point should have at least two images, one before and one after. Over 40,000 high-resolution images of 2048 x 1080 pixels are generated so that this model trains with high accuracy and efficiency. A powerful model is created utilizing the optimal combination of ResNet50, convolution layers, and LSTM layers. ResNet50 is a highly efficient pre-trained image classifier. Using transfer learning, we have custom trained the binaries that have been pre-trained on the ImageNet dataset to specialize for this experiment. This combination of networks becomes each branch of this Multi-Branched Network Architecture. Since LSTM's networks are considered prominent in finding the temporal variance in a given time series data, it is applied in the system to extrapolate vehicle speed from each branch's abstracted encodings. As the system is also detecting the license plate of the over-speeding car, YOLOv4 tiny is implemented for object detection after extensive experimentation and TesseractOCR for extracting the text out of the bounding box.

The system is available in the following GitHub repository-
https://github.com/monacotime/Vehicle-Speed-Detection-Using-Multi-Branch-Networks-from-Temporal-Image-Pairs

II. Related Work

The paper [1] uses an optical flow algorithm, specifically the Lucas-Kanade algorithm [2] for object tracking, speed detec-

tion, motion, and direction of motion detection. It combines Kalman filters [3] for occlusion reduction and optical flow for delivering accurate speed results. Optical flow is applied on clustered object data from DB-SCAN [4] performed on foreground subtraction. Foreground subtraction is performed using Gaussian Mixture Models [5] with dynamic background, it's role is to classify whether pixels belong to foreground or background. The speed detection algorithm used has its constraints like video should have the same brightness levels throughout, temporal persistence and points in the same context should belong to the same surface and have coherent motion. This paper [1] focuses more on vehicle detection/ object detection from video stream trained on the constant background context.

Another method proposed in the paper [6] for foreground extraction is the CVS (combination of saturation and value) which is based on the frame difference. The mean filter method [7] is used for the detection of foreground and then CVS is applied to extract the background along with the background color particularly with low light settings and fixed camera positions. Vehicle location is considered as 2d, which is mapped to real-time 3d vehicle positions considered to have 2d motion finally achieving 2d-to-2d mapping. Image blob is made out of binary image generated after CVS, performing edge detection, noise and dilation removal, and object labeling. Finally, speed detection is performed using blob centroid and results are calibrated. CVS method applies a constant threshold to all the pixels and in low vehicle speed settings, a low threshold will detect imperfect background spots and a high threshold generates imperfect foreground.

To detect vehicle speed using computer vision, object detection and tracking are the initial steps to be performed. The paper [8] proposes a system with WaldBoost detector [9] used along with modified TLD trackers [10]. WaldBoost detector is based on the cascaded classifier made by Viola and Jones [13]. This method creates fine-grained detection cascades similar to that of Viola and Jones. 80,000 positive samples were generated and trained on WaldBoost from 5000 car samples with a negative background sample count exceeding 1 billion. WaldBoost[9] uses a sliding window approach to successfully detect vehicle positions. Successful detections are passed to FOT trackers[11] which internally use the Lucas-Kanade algorithm [2] to accurately track the vehicle position. LrD (Learning and re-detection) [12] corrects the positional errors from FOT and contributes inaccurate position tracking of the vehicle.

Some approaches [14] utilize camera optics and digital image processing for vehicle speed estimation. In this method, coordinates of the vehicle in the video images which are considered as 2d are mapped into real-world 3d coordinates. The camera is positioned with its optical axis tilted at an angle from the road, utilizing geometrical optics a mapping function is developed between the real-world domain and image domain. The mapping function is used to calibrate results that utilize the vanishing point and size of the vehicle. The cross-section of parallel mark lines is used to calculate the vanishing point. For vehicle detection on video images, they are using a method analogous to Chris Stauffer [18] which over the period provides adaptive background. For tracing the vehicle positions they have employed method homogenous to Cucchiara and Freund[17]. The method proposed is inexpensive as it requisites only a single camera and a computer for vehicle speed detection in multiple lanes.

Another method considering vehicle speed detection utilizing image and video processing is proposed in this paper [15]. The method designed commences with video segmentation where they have sampled the frames to get only 3 frames per second instead of 30 frames per second to avoid unwanted redundancies and to reduce the time complexity. Median filtering [19] is employed in this method to remove the high-frequency noises like salt-pepper noise and convolution noise (blurring) for preprocessing. For background subtraction, they have employed a reference frame for detaching the background of the image. In this method, edge detection is implemented by employing a couple of methods like smoothing, gradients, non-maximum suppression, double thresholding, edge tracking by hysteresis. For the further enhancement of the edges, they have executed morphological operations like dilation and erosion. To reduce the static interference and heavy background, vehicle segmentation is implemented with the help of masks that vary according to the background, and segmented images are obtained using a logic AND operator. In this paper [15], speed detection has been implemented by using a corner detection method i.e Moravec operator [21]. Moravec operator has a high-performance rate, better localization and sturdiness considering noise. By employing the Moravec approach all the corners of the frames are detected and then by utilizing the distance formula speed is calculated considering the corners of both the frames. However, despite using such convoluted methods, the efficiency lacks when this method is tried on multiple vehicles on cloudy days resulting in false positives.

Speed detection using GMR sensors is one of a kind method that is developed in this domain. In this paper [16], method employed requisites the use of two GMR sensors for speed detection which is a stable method considering the parameters like temperature and noise. The complete circuit designed has GMR sensors, a low pass filter, high pass filter, an instrumentation amplifier, a micro-controller with A/D converter, a radio module and a fast ram. This system has to be positioned on the road, when a vehicle passes above this system, the micro-controller with A/D converter activates the signal of the sensors to detect the speed of the vehicle. In this way, this system designed can assist the traffic monitoring systems.

III. Proposed Work

A. Ground Truth Data generation with a 3D Render Engine

With the nature of the dataset being that each data point must have at least two images, namely one before and one after with a time gap of exactly 0.25 seconds and an associated ground-truth value of the correct speed read by a speedometer,

the decision was taken to collect the data through an automated 3D simulation.

The above fig.1 shows a 3D Render Engine simulating a scene for generating high-resolution images. We have programmatically written a controller script for the Render Engine such that we have complete control over all the minute hyperparameters of each data point.

While rendering, this method enables to create 3D simulations and animations of any car driving down the road in a more robust and efficient manner while also providing complete control over its critical hyperparameters ie. speed, color, license plates, distance, time variance, etc.

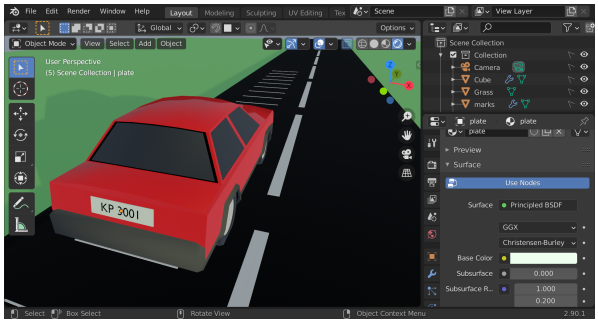

Fig. 1. 3D Animation and Render Engine

Varying the hyperparameters helps produce a dataset that is more identical to real-world applications. Mathematical estimation methods are written that estimate the position and the angle of the car after the given delta time and this is heavily used by the controller for extreme optimization during dataset rendering and reduce the time complexity by more than 1000% from an average of 2000 seconds to 2 seconds. And now because with the controller, there is no need to re-animate the whole scene for each variation in the hyperparameter but rather just render 2 shots, one for before and one for after the delta time ie. 0.25 seconds and all the in-between calculations can be estimated mathematically.

The dataset has train, test and validation split of 80:10:10. Which enables us to conduct detailed analysis of the model's performance

With this method, over 40,000 high-quality images with a resolution of 2,048 x 1,080 pixels are generated, that enables to create the dataset with over 20,000 pairs of before and after images and their co-responding mathematically estimated speeds as the data points.

B. System Architectures

The System includes three main blocks, Multi-Branched network block, License plate detection block, and Tesseract OCR block fig.2. Two consecutive images of a car moving on a straight road are captured with the marking grids visible in the camera set at a pre-defined offset angle. Two sets of images are taken one when the car is first completely visible in the frame and the other image is taken after a set amount of time - to calculate the speed of the car. In this experiment, time is kept as the constraint.

These images from the input of the Multi-Branched network block. The block utilizes a combination of ResNet50 layers and custom-made convolution layers feeding forward to the LSTM block whose output is then used to summarize and evaluate the output speed. The speed estimations made by the model are checked to single out vehicles offending a certain speed threshold and their respective images are sent forward for further processing.

To generate the ground truth for training YOLOv4 tiny [24], each car image is manually labelled for license plate and transfer learn it on pre-trained YOLOv4 [24] weights on CoCo dataset. CoCo dataset consists of 330,000 plus labeled training images with 80 object categories. pre-trained YOLOv4 weights contain the features for more than 15,000,000 object instances and their respective contexts. With the experiments, it is concluded that these well-generalized weights give higher accuracy as compared to other methods like WaldBoost [5] when custom trained and therefore is selected for the system. This Custom trained YOLOv4 tiny model is then converted into a TensorFlow Lite model to slightly push the performance level of the model for license plate detection system and be suitable for even low compute mobile devices.

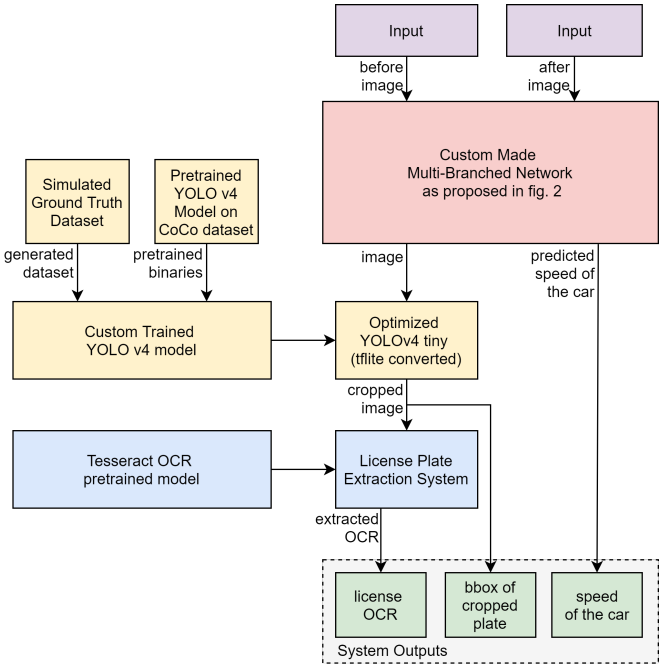

Fig. 2. System Architecture

Vehicle images taken during the speed estimation process in the Multi-Branched Network block which violates determining speed limits are captured and passed through YOLOv4 tiny[10] which is already trained and converted into the TensorFlow lite model. The detected number plate is cropped using the bounding box coordinates using computer vision techniques and passed on to the OCR block for text extraction. In the OCR block, Tesseract OCR implementation in python namely pytesseract is implemented. License plate images are

pre-processed and enhanced to suit the OCR process.

Outputs from the system, evaluated speed, License plate number, and the image of the vehicle, can be then used by the enforcement authorities to get the account of the vehicle in question and to hold the owner accountable.

C. Multi-Branched Network

In order to adapt and specialize the LSTM [23] block to differentiate the temporal variance between the image pairs and to ensure that each image's encoded representation is independently factored into the regressive estimation of the speed in the final neuron of the dense network, the system proposes a multi-branched approach. This novel approach eliminates various issues and challenges faced by this [15] and this [16] approach. Double detection and double encoding of each temporal pair diminish the possibility of the model over-fitting the attributes of any single image out of the pair during the process of creating their representation. For their optimization, the ADAM optimizer is used.

$$w_{t+1} = w_t - \alpha m_t \tag{1}$$

$$m_t = \beta m_{t-1} + (1-\beta)\begin{bmatrix}\delta L \\ \delta w_t\end{bmatrix} \tag{2}$$

The loss is calculated using the Root Mean Squared loss function which is described in (3)

$$L(y,\widehat{y}) = \frac{1}{N}\sum_{i=0}^{N}(y,\widehat{y})^2 \tag{3}$$

As a result, the LSTMs are presented with only the extracted higher-level abstraction of the temporal variance and must distinguish and generalize to the difference in the patterns of abstraction.

It is discovered in the experiments that, this end to end approach of teaching the model to generalize the speed of the car distinctly outperforms other data-driven statistical approaches like edge-based histogram flow [20] detection and handcrafted feature engineered approach in terms of robustness in the variation of the application, speed, and accuracy. This adaptive characteristic of the model has a direct correlation with the architecture of the proposed model.

The input to the proposed model will be segmented equally to ensure no bias are inherently present in the data points even before the image has entered the model's input layer. The slicing function will ensure that the images are partitioned NumPy vector arrays of equal size and equal dimensions. Then transfer the 2D vectors to the input layers of each branch independently. No data will be tampered with during this process other than the slicing and resizing to ensure it fits the input dimension of the ResNet50 model.

ResNet50 [22] is a pre-trained model architecture that has been trained on different datasets compared to own use case ie. car speed prediction, but it can provide a critical starting point since the new task can make use of the features that have been learned and abstracted while training from the old previous dataset. For the specific proposed model, it is opted to utilize the ResNet50 architecture that has been trained on the ImageNet dataset that is over 14,000,000 images large categorized into 1,000 classes.

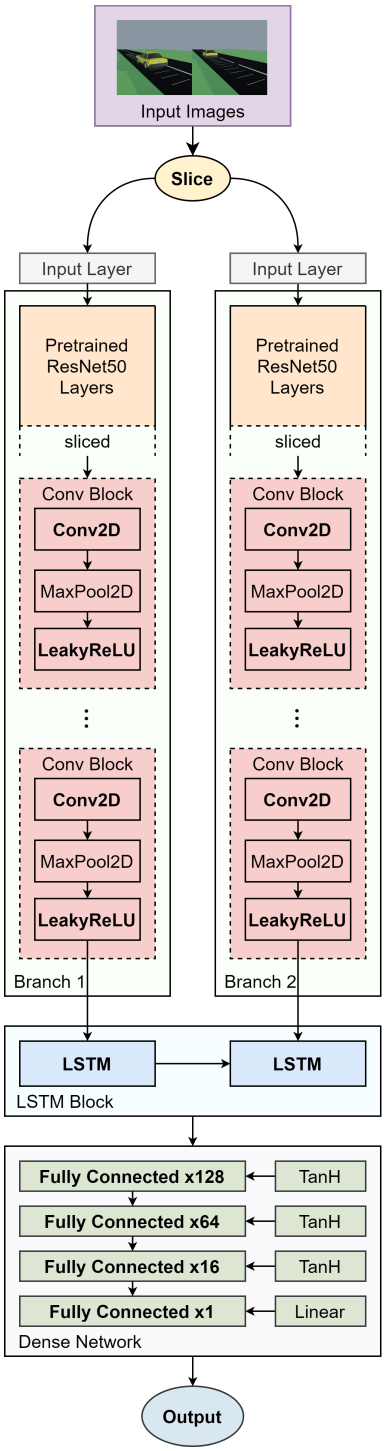

Fig. 3. Multi-Branched Network Architecture

The ResNet50 model used has 5 stages each with a convolution and identity block with over 23,000,000 trainable parameters which constitutes a total of 48 convolution layers. This enables it to design the branched network with having a

very deep neural network without having to tackle the problem of vanishing gradient that is prevalent at such deep layers.

By utilizing the unique advantage of transfer learning on ResNet50 [22] and then custom training the last 2 deleted layers, the abstraction that the ResNet50 provides is only relevant to the Cars and it only encodes the representation of the cars. These last few layers are replaced with the chain of Convolution Blocks as shown in fig. 3. Each output of the convolution layer has to pass through the LeakyReLU [30] activation function for each of the filters. The LeakyReLU function computes

$$f(x) = 1(x < 0)(\alpha x) + 1(x >= 0)(x) \tag{4}$$

for each input, where α is a very small constant. With this, train them both to specifically generalize the abstraction of cars that are present in the custom dataset are generated through the Animation and 3D Render Engine.

These abstractions are done independently for each branch separately and the encoded representations are flattened and passed on as inputs to individual LSTM [23] neurons. Long Short-Term Memory (LSTM) networks are specialists in finding temporal variance in a given time series data.

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i) \tag{5}$$

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f) \tag{6}$$

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o) \tag{7}$$

Each LSTM block has three gates namely forget gate (f_t), update/input gate (i_t), output gate (o_t). And in this specific case, the time series data is the two very high-level abstract representations of each car in sequence from each independent branch. Each LSTM neuron passes the output to the next layer as well as the neuron adjacent to it in the same layer.

Once the LSTM block has calculated an output after factoring the encoded representation from each branch, the output is passed onto the next block which is a series of layers of Densely or Fully connected neurons. Meaning that each neuron in these layers is connected to all the neurons in the adjacent layers. This block narrows down the output of all the other blocks and branches and funnels them into smaller and smaller-sized layers. This increase in restriction forces the neurons to calculate a representation of the whole data point in terms of speed. Especially in the most restricted layer which is the last and the final layer with only one neuron. This neuron's output is passed through a linear activation function which then finally is considered as the output of the whole model.

D. YOLOv4 tiny for number plate detection

YOLOv4 model[24] is the fastest object detection model with the tiny variants performing at 155 frames per second and giving mAP(mean average precision) rates almost double as compared to other real-time object detectors. It sees object detection as a regression problem and is trained to learn the general representation of an object from the entire image as a result of encoding the context information about the classes and their representations. By looking at an entire image it predicts bounding boxes and confidence for the object (number plate in our case) as seen in the fig.4.

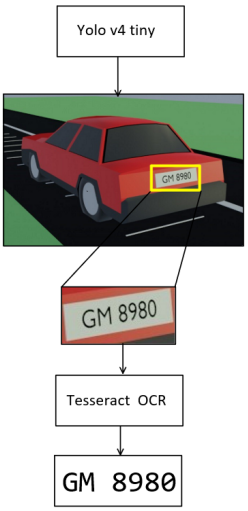

Fig. 4. Number Plate Detection Flow

The prediction consists of x, y, w, h, confidence - x, y represents the center of the bounding box, confidence is calculated by taking a product of object probability and object ground truth (IOU, intersection over union). High-resolution images, greater than 448 * 448 are used as the tiniest bit of information from the image is used for training. Faster rates and higher performance of the YOLOv4 tiny model on mobile IoT devices make it an excellent choice for the number plate detection problem. Transfer learning on YOLOv4 tiny model is used in the system to detect number plates using annotated car images taken during the experimentation for 8 hours on NVIDIA Tesla K80 GPU.

E. Tesseract OCR for number extraction from number plate

Tesseract OCR [25] is one of the notable OCR engines available for text extraction after its development in 1984. It detects the outline of the text with the help of connected component analysis and stores the outlines to detect inverse text - black-on-white tests are very easy to detect and extract however that is not the case with white-on-black text, tesseract OCR detects the inverse text easily. Text outlines are further broken into word outlines and an attempt is made to detect words out of it. The detections with satisfactory confidences are passed to adaptive classifiers for getting accurate detections. word outlines are detected using spaces and fuzzy spaces which are then changed to x-heights in the final stage to detect small captioned words. To extract the number out of the number plates by tesseract, a python implementation of tesseract OCR is used for number extraction, fig.4. with the high accuracy of tesseract OCR numbers from the number, plates are almost always extracted 99% of the times.

IV. Results Analysis

A. Comparative Analysis

The implemented method overcomes the variable background problem faced due to the application of the Lucas-Kanade algorithm [4]. Since the method's only requirement is a marking grid, as a context, visible on the input image, The system is immune to variable conditions like lightings, seasons change, variable backgrounds and other image occlusion problems faced in [4]. On the special comparative validation dataset of 3050 images (1280 x 720 pixels) with difficult background segmentation, this [4] previous method only achieved an accuracy of 34.6% whereas the implemented model was able to outperform it by correctly predict with 87.7% validation accuracy on the same dataset.

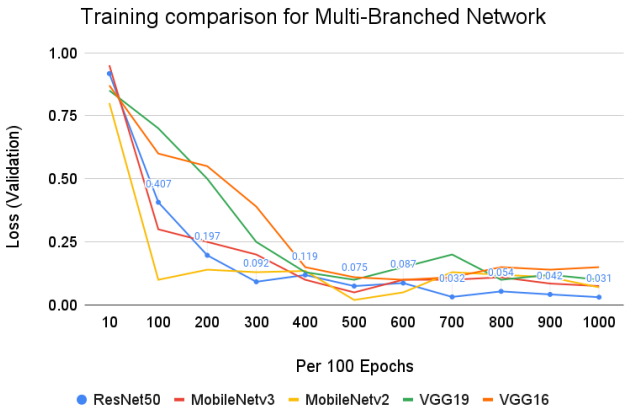

Fig. 5. A comparison between different pre-trained binaries for Multi-Branched Networks

The method proposed in the system is based on end-to-end learning and therefore does not require any handcrafted correction algorithms like Lrd[5], Kalman Filters[5] or any other intermediate steps like foreground-extraction, contour detection, and centroid estimation as proposed in [5]. This method also foregoes the need for object tracking and its heavy computational demands as proposed on [5], since it only needs to apply image classification twice. Therefore, the mean execution time taken by this [5] proposal ,when run iteratively over 200,000 predictions, is 1440s milliseconds while this method averages about 67 milliseconds over the same iterations. Making the model over 21.5 times faster and better optimized in time complexity to the previous proposal[5].

Although methods like GMR tracking [3] utilizes GMR sensors, filters, and microcontrollers to detect vehicle speed with very high accuracy, the fact that it has to be placed inside the road, introduces new vulnerabilities like unauthorized tempering and loss of visual context. It also has an added risk of component failures and high maintenance since there are many intricate and delicate parts. These shortcomings can be easily avoided using the proposed method which only uses a camera from a high vantage point making it difficult to tamper without being seen and also connected to already secure networks. The system consists of minimum moving parts and easy implementation with current infrastructure making it far more economically viable compared to this [3] proposed system.

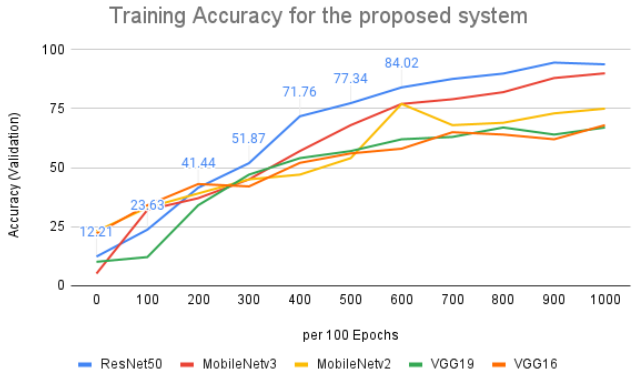

Fig. 6. A comparison between accuracy of different pre-trained binaries for Multi-Branched Networks

The components proposed in the method [3] cost between $250-$500 depending upon the commercial availability, whereas, the proposed system, with only minimum requirements, costs a nominal charge of $75-$100 with commercial over-the-shelf components. It makes the system, in the worst case scenario, 60% cheaper and, in the best case, 85% cheaper as compared to this [3] previous method.

For the implementation, the system's multi-branched network is tested with other famous image classification networks like MobileNets and VGG. ResNet50, MobileNetv3, MobileNetv2, VGG19, VGG16 are trained for 1000 epochs each on NVIDIA tesla K80. Validation loss after every 100 epochs were gathered and compared. Each model was at least trained for 2.5 hours on a dataset consisting of 40,000 high-quality, 3D rendered images.

As per the comparison fig.5, After almost 100 epochs the models start generalizing to the training data. Although MobilenetV2 [28] shows good performance for the initial 100 epochs it gradually starts over-fitting training data resulting in a performance drop on the validation set. VGG16 [26] and VGG19 [27] show similar downward trends but finally stabilize at 0.05, 0.06. Although deviating at places, ResNet50 shows a gradual increase in accuracy overall pointing towards a well-fitted model and therefore is selected for the experimentation. Mobilenetv3 [29] also shows a gradual downward trend finally finishing along with the Mobilenetv2 variant around 0.03. A model is detected with the best performance on validation data for the Multi-Branched Network.

Licence plate detection is done using yolov4 tiny for real time performance rate and application on lite devices. Yolov4 tiny and Yolov4 gives good validation accuracies as per fig.7 fig.8 but tiny variant is selected due to its lite nature. Other detectors tested for the licence detection include Mobilenetv1

and WildBoost which give decent results on licence plate dataset from imagenet.

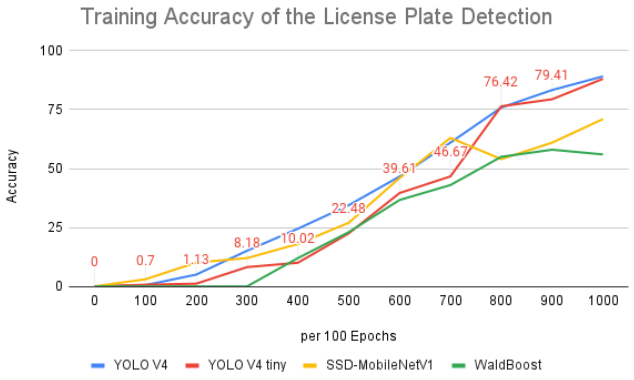

Fig. 7. Training Accuracy of the License Plate Detection

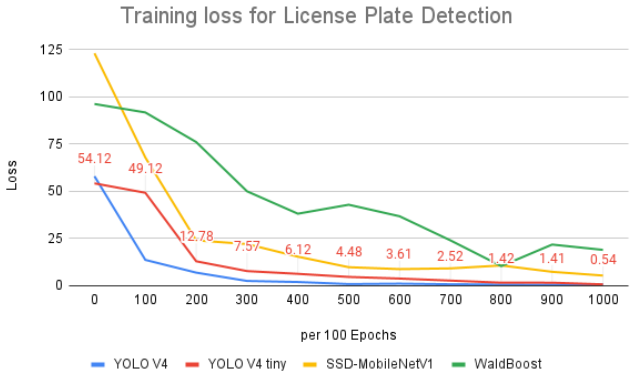

Fig. 8. Training loss for License Plate Detection

The computational time required is tested by different system components like image classifiers used in Multi-branched models, Object detectors, and OCR techniques for text extraction. Different versions of the models used are compared in the system and also similar models to make time-efficient decisions. Two sets of images are tested with low and high resolutions of 640*480 and 2048*1080 respectively. Time required to make predictions are calculated to the nearest millisecond.

TABLE I
MULTI-BRANCHED MODEL

Networks	***Image Resolution***	
	640x480	***2048x1080***
ResNet50	34.56	68.95
MobileNetv3	42.37	96.68
MobileNetv2	58.92	172.72
VGG19	103.25	215.12
VGG16	89.28	193.13

TABLE II
LICENSE DETECTION

Networks	***Image Resolution***	
	640x480	***2048x1080***
YOLO V4	21.97	46.73
YOLO V4 tiny	9.57	16.85
WaldBoost	17.18	43.47
SSD-MobileNetV1	34.39	52.58

TABLE III
OCR

Networks	***Image Resolution***	
	640x480	***2048x1080***
TesseractOCR	57.45	75.24
CloudVisionAPI(Google)	46.91	62.18

For the Multi-Branched model, ResNet50 is selected as the image classifier but still check the other candidates for time efficiency. ResNet50. Although VGG19 has got higher execution time than MobileNetv3, VGG19 outperforms MobileNet in terms of accuracy but ResNet50 outperforms all of them with respect to time as well as accuracy and is therefore selected. In order to perform object detection on the license plate, different models are compared based on different methodologies. WaldBoost model based on image cascading is known for its high accuracies but still, YOLOv4 tiny variation is used. Although YOLOv4 tiny lacks in terms of performance but in terms of computational speed to make up for its lost efficiency. CloudVisionAPI from Google performs better compared to the selected TesseractOCR but still use it due to ease of implementation and network independence.

B. Limitations

The work is trained and tested in a controlled simulated environment and therefore will have some real time limitations. System context includes a single lane road assuming one vehicle at a time. Also the distance of the markings on the road are important and part of the ground truth for the model. Roads are assumed to be straight. These conditions are necessary for the model to give accurate results. However, in order to overcome some of the limitations, like detecting speeds for multiple vehicles on a single lane road and also to focus the model on a particular vehicle, object tracking can be implemented. System will require pre-training on the practical environment ground truth in order for it to work efficiently.

V. CONCLUSION

With advancements in machine learning and deep learning technologies and the availability of state-of-the-art models with faster performance rates than the previously available technologies, it is possible to implement faster and lighter systems for vehicle speed detections combining different techniques without losing on the efficiency and the overall system

performance. The ability to make the system lightweight and deployable on mobile devices has opened up many real-time possibilities. This opens up countless possibilities for augmenting and improving the current monitoring and surveillance infrastructure. With our process, a novel and highly adaptive method has been presented with a very open-ended implementable architecture. It has a high potential for adoption since there are only 2 low-cost components. This method can be further improved for multi-lane speed detection by varying and re-training the presented model on new multi-lane ground truth.

REFERENCES

[1] J. Gerát, D. Sopiak, M. Oravec and J. Pavlovicová, "Vehicle speed detection from camera stream using image processing methods," 2017 International Symposium ELMAR, 2017, pp. 201-204, doi: 10.23919/ELMAR.2017.8124468.

[2] S. Baker and I. Matthews, "Lucas-Kanade 20 years on: A unifying framework," Int. J. Comput. Vis., vol. 56, no. 3, pp. 221–255, Feb. 2004, doi: 10.1023/B:VISI.0000011205.11775.fd.

[3] R. Chen and J. S. Liu, "Mixture Kalman filters," J. R. Stat. Soc. Ser. B Stat. Methodol., vol. 62, no. 3, pp. 493–508, Jan. 2000, doi: 10.1111/1467-9868.00246.

[4] D. Birant and A. Kut, "ST-DBSCAN: An algorithm for clustering spatial-temporal data," Data Knowl. Eng., vol. 60, no. 1, pp. 208–221, Jan. 2007, doi: 10.1016/j.datak.2006.01.013.

[5] "(PDF) Gaussian Mixture Model - method and application." https://www.researchgate.net/publication/321245699_Gaussian_Mixture_Model_-_method_and_application (accessed Jun. 24, 2021).

[6] A. Gholami Rad, A. Dehghani, and M. Rehan Karim, "Vehicle speed detection in video image sequences using CVS method," Academic Journals, Dec. 2010. doi: 10.5897/IJPS.9000618.

[7] B. Fu, X. Xiong and G. Sun, "An efficient mean filter algorithm," The 2011 IEEE/ICME International Conference on Complex Medical Engineering, 2011, pp. 466-470, doi: 10.1109/ICCME.2011.5876785.

[8] C. Caraffi, T. Vojíř, J. Trefný, J. Šochman and J. Matas, "A system for real-time detection and tracking of vehicles from a single car-mounted camera," 2012 15th International IEEE Conference on Intelligent Transportation Systems, 2012, pp. 975-982, doi: 10.1109/ITSC.2012.6338748.

[9] J. Sochman and J. Matas, "WaldBoost - learning for time constrained sequential detection," 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), 2005, pp. 150-156 vol. 2, doi: 10.1109/CVPR.2005.373.

[10] Z. Kalal, K. Mikolajczyk and J. Matas, "Face-TLD: Tracking-Learning-Detection applied to faces," 2010 IEEE International Conference on Image Processing, 2010, pp. 3789-3792, doi: 10.1109/ICIP.2010.5653525.

[11] T. Vojíř and J. Matas, "The enhanced flock of trackers," Stud. Comput. Intell., vol. 532, pp. 113–136, 2014, doi: 10.1007/978-3-642-44907-9_6.

[12] N. Wang, W. Zhou and H. Li, "Reliable Re-Detection for Long-Term Tracking," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 29, no. 3, pp. 730-743, March 2019, doi: 10.1109/TCSVT.2018.2816570.

[13] Q. Li, U. Niaz and B. Merialdo, "An improved algorithm on Viola-Jones object detector," 2012 10th International Workshop on Content-Based Multimedia Indexing (CBMI), 2012, pp. 1-6, doi: 10.1109/CBMI.2012.6269796.

[14] Jianping Wu, Zhaobin Liu, Jinxiang Li, Caidong Gu, Maoxin Si and Fangyong Tan, "An algorithm for automatic vehicle speed detection using video camera," 2009 4th International Conference on Computer Science & Education, 2009, pp. 193-196, doi: 10.1109/ICCSE.2009.5228496.

[15] K. V. K. Kumar, P. Chandrakant, S. Kumar and K. J. Kushal, "Vehicle Speed Detection Using Corner Detection," 2014 Fifth International Conference on Signal and Image Processing, 2014, pp. 253-258, doi: 10.1109/ICSIP.2014.46.

[16] J. Pelegri, J. Alberola and V. Llario, "Vehicle detection and car speed monitoring system using GMR magnetic sensors," IEEE 2002 28th Annual Conference of the Industrial Electronics Society. IECON 02, 2002, pp. 1693-1695 vol.2, doi: 10.1109/IECON.2002.1185535.

[17] N. Peterfreund, "Robust tracking of position and velocity with Kalman snakes," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 21, no. 6, pp. 564-569, June 1999, doi: 10.1109/34.771328.

[18] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2, pp. 246–252, 1999, doi: 10.1109/cvpr.1999.784637.

[19] G. Devarajan, V. K. Aatre and C. S. Sridhar, "Analysis of median filter," ACE '90. Proceedings of [XVI Annual Convention and Exhibition of the IEEE In India], 1990, pp. 274-276, doi: 10.1109/ACE.1990.762694.

[20] Shih-Kuan Liao and Baug-Yu Liu, "An edge-based approach to improve optical flow algorithm," 2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE), 2010, pp. V6-45-V6-51, doi: 10.1109/ICACTE.2010.5579363.

[21] M. S. Kumari and B. H. Shekar, "On the use of Moravec operator for text detection in document images and video frames," 2011 International Conference on Recent Trends in Information Technology (ICRTIT), 2011, pp. 910-914, doi: 10.1109/ICRTIT.2011.5972394.

[22] A. T. Sasongko and M. Ivan Fanany, "Indonesia Toll Road Vehicle Classification Using Transfer Learning with Pre-trained Resnet Models," 2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), 2019, pp. 373-378, doi: 10.1109/ISRITI48646.2019.9034590.

[23] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS 1 LSTM: A Search Space Odyssey," doi: 10.1109/TNNLS.2016.2582924.

[24] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.

[25] R. Smith, "An Overview of the Tesseract OCR Engine," Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), 2007, pp. 629-633, doi: 10.1109/ICDAR.2007.4376991.

[26] H. Qassim, A. Verma and D. Feinzimer, "Compressed residual-VGG16 CNN model for big data places image recognition," 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), 2018, pp. 169-175, doi: 10.1109/CCWC.2018.8301729.

[27] T. Carvalho, E. R. S. de Rezende, M. T. P. Alves, F. K. C. Balieiro and R. B. Sovat, "Exposing Computer Generated Images by Eye's Region Classification via Transfer Learning of VGG19 CNN," 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), 2017, pp. 866-870, doi: 10.1109/ICMLA.2017.00-47.

[28] B. Koonce, "MobileNetV3," in Convolutional Neural Networks with Swift for Tensorflow, Apress, 2021, pp. 125–144.

[29] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," 2018.

[30] J. Xu, Z. Li, B. Du, M. Zhang and J. Liu, "Reluplex made more practical: Leaky ReLU," 2020 IEEE Symposium on Computers and Communications (ISCC), 2020, pp. 1-7, doi: 10.1109/ISCC50000.2020.9219587.

[31] J. Zong Chen, "Automatic Vehicle License Plate Detection using K-Means Clustering Algorithm and CNN", March 2021, vol. 3, no. 1, pp. 15-23, 2021. Available: 10.36548/jeea.2021.1.002.

[32] S. S and H. Wang, "Security Enhancement in Smart Vehicle Using Blockchain-based Architectural Framework", June 2021, vol. 3, no. 2, pp. 90-100, 2021. Available: 10.36548/jaicn.2021.2.002.

[33] R. Bestak, "INTELLIGENT TRAFFIC CONTROL DEVICE MODEL USING AD HOC NETWORK", December 2019, vol. 01, no. 02, pp. 68-76, 2019. Available: 10.36548/jitdw.2019.2.002.

[34] D. R and K. R, "Bus-Based VANET using ACO Multipath Routing Algorithm", March 2021, vol. 3, no. 1, pp. 40-48, 2021. Available: 10.36548/jtcsst.2021.1.004.