ISSN No:-2456-2165

Complete Panoptic Traffic Recognition System with Ensemble of YOLO Family Models

Deniz Sen
Service Application DC Huawei Turkiye R&D Center Istanbul, Turkiye

Abstract:- Erroneous driver behavior is one of the most dangerous aspects of driving in traffic, which can be prevented if the dangerous actions are detected immediately and the necessary actions are taken. However, in order to be able to implement a hazard detection framework, we first need a model that can perceive the related objects in the traffic scene, which is not a trivial task considering the diversity of the outside conditions. In this paper, we propose a computer vision-based system that only requires dash camera footage to compute binary drivable area segmentation, dashed and straight lane line segmentation, and traffic object detection. We utilize two YOLOv5s and YOLOP, a multitask architecture, to create a complete panoptic perception model and train the member models using the public BDD100K, traffic sign dataset, and our private dataset. To reduce the significant annotation overhead of the segmentation tasks, we use semi-supervised learning techniques and a different annotation approach for lane line labeling. We also present 2 lane violation detection algorithms and temporal smoothing techniques for the segmentation tasks. We managed to achieve remarkable results in all 3 of our tasks and showed the usability of our system under real-world scenarios.

Keywords:- *YOLOP; Lane Detection; Traffic Object Detection; Drivable Area Segmentation; Traffic Violation.*

I. INTRODUCTION

The ever-accelerating progress of Artificial Intelligence (AI) has illuminated a transformative path for the automotive industry, offering an array of opportunities to enhance both vehicular safety and the quality of the driving experience. In an era marked by data-driven innovations and technological convergence, the role of AI in reshaping the dynamics of road safety and driver behavior analysis has become increasingly prominent.

This study embarks on an exploration of the noteworthy advancements in AI technology, with a specific focus on its application within vehicles. Unlike the pursuit of fully autonomous driving, our primary objective is to develop a robust and versatile system that will lay the groundwork for ongoing research aimed at detecting and mitigating unsafe driver behaviors. This system relies on computer vision as the fundamental technology, harnessing visual data captured by a dashboard camera (dash cam) as its sole sensory input.

The core aim of this paper is to introduce a comprehensive computer vision-based system that functions independently, without additional sensory data, to segment drivable areas and lanes while simultaneously detecting the presence of pedestrians, vehicles, traffic lights, and signs. Furthermore, our research seeks to push the boundaries of road safety analysis by proposing several innovative rule-based techniques designed specifically for the detection of lane violations, a prevalent and hazardous unsafe behavior exhibited by drivers.

This holistic approach to driver behavior analysis not only aligns with current trends in the AI-driven automotive industry but also serves as a foundation for further research. The developed system, combined with rule-based methodologies, offers an adaptable platform for researchers and safety experts to delve into the complexities of driver behavior, identify potential risks, and develop interventions to reduce accidents and enhance road safety.

In the subsequent sections of this paper, we will provide a comprehensive overview of our computer vision-based system, its functionalities, and the rule-based techniques we have devised for detecting lane violations. Our contributions can be listed as follows:

- Provide a simpler method of lane annotation technique that can reduce the labeling overhead.
- Train a multitask network that is capable of drivable area and lane segmentation with the distinction of dashed and straight lines, and traffic object detection, using two large datasets and a semi-supervised learning method.
- Train separate object detectors that can detect people and traffic lights/signs.
- Propose two simple yet effective lane violation techniques that have a small computational cost.

II. LITERATURE SURVEY

There have been countless works on lane detection, which is a popular problem that requires to be addressed to make AI utilizable in the automotive industry. Lane detection can generally be separated into feature-based and model-based methods. One common approach in lane detection is the use of inverse perspective mapping (IPM) to transform the image into a top-down view, which simplifies the detection process [1]. Random sample consensus (RANSAC) is a popular algorithm used for fitting lane models to the transformed image [2]. Other techniques, such as Gabor filters and Hough transform, have also been

employed for lane detection [3].

Recent advancements in deep learning have led to the development of end-to-end lane detection algorithms that can directly predict lane markings from raw image data [4]. These models eliminate the need for hand-crafted features and post-processing techniques, making them more scalable and computationally efficient. Instance segmentation approaches have also been proposed, where lane markings are treated as individual instances and segmented accordingly [5, 8, 10]. The segmentation methods also include current lane classification (left, mid, right) [7]. Besides, vanishing point detection is also a related problem in the field therefore there have also been works that addressed it using deep learning methods [6].

In addition to lane detection, there has been research on multitask AI models for traffic data analysis. These models aim to perform multiple tasks simultaneously, such as lane detection, object identification, and traffic sign recognition [9]. By integrating multiple tasks into a single model, these multitask AI models can improve the performance and efficiency of traffic data analysis systems. In this paper, we utilize YOLOP, which is also a multitask network that is capable of separately segmenting drivable areas and lane lines, along with detecting traffic objects [11]. Hybrid Nets and YOLOPv2 were introduced following YOLOP, completing the same tasks with small yet significant changes in the architecture and the training procedures [12, 13].

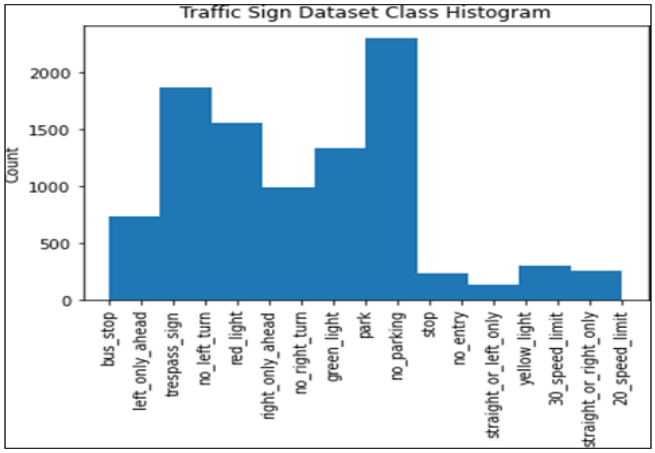

Fig 1 Combined BDD100K and Private Dataset Class Histogram

III. METHODOLOGY

A. Data Gathering

In this work, we utilized 3 datasets, 2 of which are publicly available and one is composed of private dashcam frames.

➢ *BDD100K*

BDD100K is a large-scale diverse driving video dataset that has been widely used in computer vision research, particularly in the field of autonomous driving [14]. It stands for Berkeley DeepDrive 100,000, indicating that it contains 100,000 high- quality video clips captured from diverse driving scenarios. The dataset is designed to address the limitations of existing driving datasets by providing a more comprehensive and diverse collection of real-world driving scenes. It covers a wide range of driving conditions, including different weather conditions, lighting conditions, road types, and traffic patterns. The videos are captured from a variety of camera viewpoints, including front-facing, rear-facing, and side-facing cameras, providing a rich source of visual information. BDD100K includes detailed annotations for various tasks, such as object detection, instance segmentation, and drivable area segmentation. The dataset also provides additional information, such as GPS coordinates, vehicle speed, and sensor information, which can be used for more advanced research and analysis. The detection classes are separated as 'person', 'rider', 'car', 'bus', 'truck', 'bike', 'motor', 'green light', 'red light', 'yellow light', 'undefined traffic light', 'traffic sign', 'train', however in our case, we densified the labels to be 'vehicle', 'person', and the traffic lights. On the other hand, the lane line information is not held in a segmentation mask as in regular cases, instead, the vertices of the lane lines are kept as data which then can be used to fit some 1st or 2nd-degree polynomial, based on the number of provided vertices.

It should be noted that the drivable area binary segments and the lane line annotations are only available for the first frame of each video, which led us to use only a small portion of the dataset; besides, 100,000 samples are arguably a sufficient amount of data for the trained CNN models with high inductive bias.

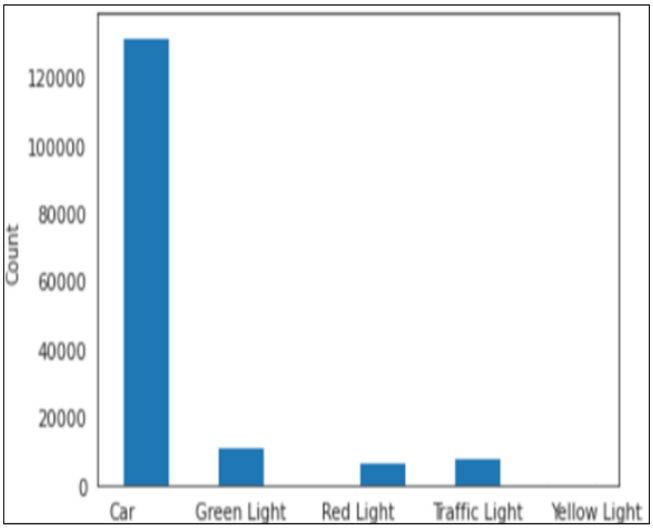

Fig 2 Combined BDD100K and Private Dataset Class Histogram

➢ *Private Dashcam Dataset*

Our private dataset was gathered from the dashcams installed on the windshield of the taxis of Istanbul by our partner. The frames are taken from everyday routes that the taxi drivers go through, thus the clips are highly natural and properly represent the real-world scenarios. On the other hand, unlike BDD100K, the frames of the private dataset are highly distorted, due to the low quality of the dashcams. The low- quality frames arguably make the task even more challenging. The dataset is composed of 15.2k frames and

each frame is annotated to hold the traffic light, vehicle, and lane information. The traffic light bounding boxes are separated based on their colors, i.e. red, green, and yellow. Furthermore, each vehicle is labeled as a 'car', similar to the densification performed for the BDD100K dataset.

For the annotation of the frames, we used a tool called Remo, which is a web application for managing and annotating raw data. While being relatively lightweight, the tool is short on functionality and only supports bounding box annotation for object detection tasks. To get around this issue, we annotated the corners of the lines as bounding boxes, and took the centroids of the bounding boxes as the vertices, to match the format of BDD100K lane lines. Note that the vertices of each line were labeled as the same number. Furthermore, as detecting the dashed and straight lines is a key requirement for our case, we set the labels of the straight lines as odd numbers, and the dashed ones as even numbers. We utilized this information in a posterior operation to generate the corresponding lane mask.

- *Traffic Sign Dataset*

The traffic sign dataset is an object detection dataset that contains annotations of the traffic signs and lights captured by high-quality dashcam frames. The resolution of each frame is 1920x1080, similar to BDD100K, but different from the private dataset. The dataset is composed of 4706 images, 3546 of which are separated for the train set, and 1060 of which are separated for the test set. The annotations are composed of 18 classes; 'trespass sign', 'straight or left only', 'straight or right only', 'left only ahead', '20 speed limit end', '30 speed limit', '20 speed limit', 'right only ahead', 'no right turn', 'no left turn', 'stop', 'no parking', 'park', 'bus stop', 'green light', 'yellow light' and 'red light'. The class instance histogram can be found in Fig. 1.

- *Dataset Combination*

To be able to use all the datasets in our possession, we utilized a basic yet effective method, which is annotating the unlabeled data with a model that was trained with the annotated portion of the dataset. In this case, the fully annotated part of the dataset is BDD100K; it includes the drivable area segments that are absent in the private dataset. For that reason, we first trained a multitask network with a drivable area segmentation head, then, annotated the frames coming from the private dataset. The class instance histogram of the combined dataset can be found in Fig. 2.

B. Utilized Models

In this section, the utilized methods will be presented in detail. These methods can be summarized as the model architectures and approaches for lane violation detection. The complete pipeline that will be presented can be found in Fig. 6.

- *YOLOP*

You Only Look Once for Panoptic Driving Perception (YOLOP) is a multitask CNN that is capable of doing binary drivable area and lane segmentation along with traffic object detection [11]. The high-level architecture comprises a multi- head autoencoder scheme, where the features are extracted through a shared encoder (backbone and neck), and then fed into 3 decoupled decoders(heads) that were trained to utilize the input features to their separate tasks. The backbone is taken from the popular CSPDarknet architecture, due to its real-time efficiency and proven impact on the propagated features [16]. Furthermore, Spatial Pyramid Pooling (SPP) module and Feature Pyramid Network (FPN) architecture compose the neck part of the network [19, 20]. As usual, the neck is utilized to couple and fuse the features coming from different semantic scales, by combining the bottom-up features from the backbone with the bottom-down features computed by upsampling. The detection head takes a similar approach to YOLOv4, in that the detections are selected via precomputed anchor boxes, which are computed based on the different scales that are computed in the neck [17].

On the other hand, the segmentation heads upsample the top- down features coming from the neck several times; notice that the upsampling method does not use deconvolution to reduce computational costs, instead, nearest interpolation is preferred. The upsampling is done until the size of the obtained 2D feature map matches the input's dimensions. In the original paper, both segmentation heads yield binary maps which signify the presence of either a lane or a drivable area; for our task, we modified the output layer of the lane segmentation head to also classify the lane's type, as whether straight or dashed. This classification feature is crucial for our requirement as we intend to detect lane violations that occur due to the passage through straight lines.

As the initial pretrained YOLOP model came with 2 binary segmentation heads to be suitable with the data coming from the BDD100K dataset, we had to modify the last layer of the lane segmentation head to output 3 channels (no line, dashed lane, straight lane) instead of the original 2.

We utilized a hybrid loss function which is the combination of 3 loss functions crafted for each individual task. The loss function of the detector head is the addition of the objectness loss, box, and classification losses. The objectness loss is the standard binary cross entropy function which indicates whether the detected area contains an object, furthermore, the box loss signifies how accurate the predicted bounding box is when compared to the ground truth. Finally, the classification loss is the standard focal loss that has seen common use in recent years to overcome the struggles raised by the class imbalance problem. On the other side, for both segmentation tasks, we use the focal Tversky loss which is a common method to overcome the severe class imbalance problem raised by the nature of the lane and drivable area segmentation tasks. Note in the overall data, the ratios of drivable area and lane pixels to the negative pixels are very small, as most of the frames do not contain either of the positive classes. The focal Tversky loss function was shown to be highly effective in forcing the models to tune the parameters more when encountered with hard samples. The Tversky Index, which is the main equation that aims to maximize true positive and minimize false negative predictions is given in (1).

ISSN No:-2456-2165

$$TI = \frac{TP}{TP + \alpha FN + \beta FP} \quad (1)$$

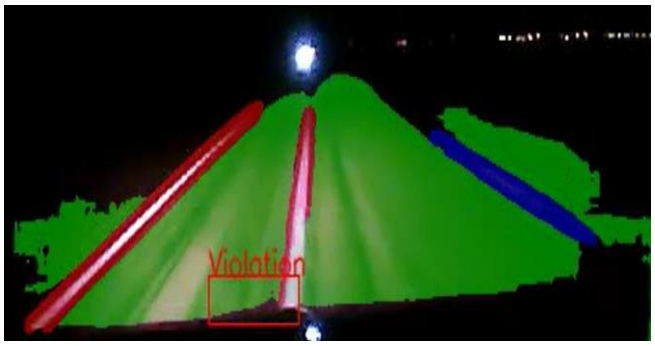

Fig 3 The Model Performance Under Dark Conditions

The focal Tversky loss which capitalizes more in hard examples thanks to the γ parameter, is given in (2).

$$FTL = (1 - TI) \quad (2)$$

➢ *YOLOv5*

YOLOv5, short for "You Only Look Once Version 5," is a state-of-the-art object detection framework that has gained prominence in computer vision research and applications [15]. It consists of several key components, each contributing to its exceptional performance. The backbone of YOLOv5 is a deep convolutional neural network, often based on architectures like CSPDarknet53 or EfficientNet, which extracts features from input images [17]. The neck component, typically PANet or PANet-Lite, then combines these features at different scales to enhance object localization [18]. YOLOv5's unique feature is the detection head, composed of several prediction layers responsible for bounding box coordinates, objectness scores, and class probabilities. The anchor boxes, a set of predefined bounding box priors, aid in predicting accurate object locations. Furthermore, YOLOv5 implements a novel loss function, CIoU (Complete Intersection over Union), which optimizes both localization accuracy and class prediction. Its versatility is evident through support for various input sizes, hardware accelerators like GPUs, and an efficient inference process that makes it suitable for real-time applications. With its robust components and superior performance, YOLOv5 has become a pivotal tool in the field of computer vision, advancing object detection capabilities.

We trained 2 Small YOLOv5(YOLOv5s) models to detect traffic lights/signs and people. We have empirically seen that training the YOLOP detection head to detect people along with the traffic objects yielded suboptimal performance in real- world situations, therefore we went on with training multiple smaller models, which can learn completely different features from each other.

C. Posterior Algorithms

In this section, the proposed algorithms will be applied after the model inference is made. These algorithms are crafted to enhance the model segmentation outputs, which will be used in the following operations.

➢ *Temporal Smoothing for Segmentation Outputs*

Temporal smoothing in computer vision refers to the process of reducing noise or fluctuations in a sequence of images or video frames over time. It involves applying a filtering technique to the temporal dimension of the data to create a more stable and visually coherent representation. The main goal of temporal smoothing is to improve the quality and consistency of the video sequence by reducing unwanted variations caused by factors such as camera shake, sensor noise, or rapid changes in the scene. It helps to create a more visually pleasing and easier-to-analyze video stream. We utilize this scheme in our pipeline as well both in lane and drivable area allows the smoothing process to take certain predictions more into account while deciding on the presence of drivable areas.

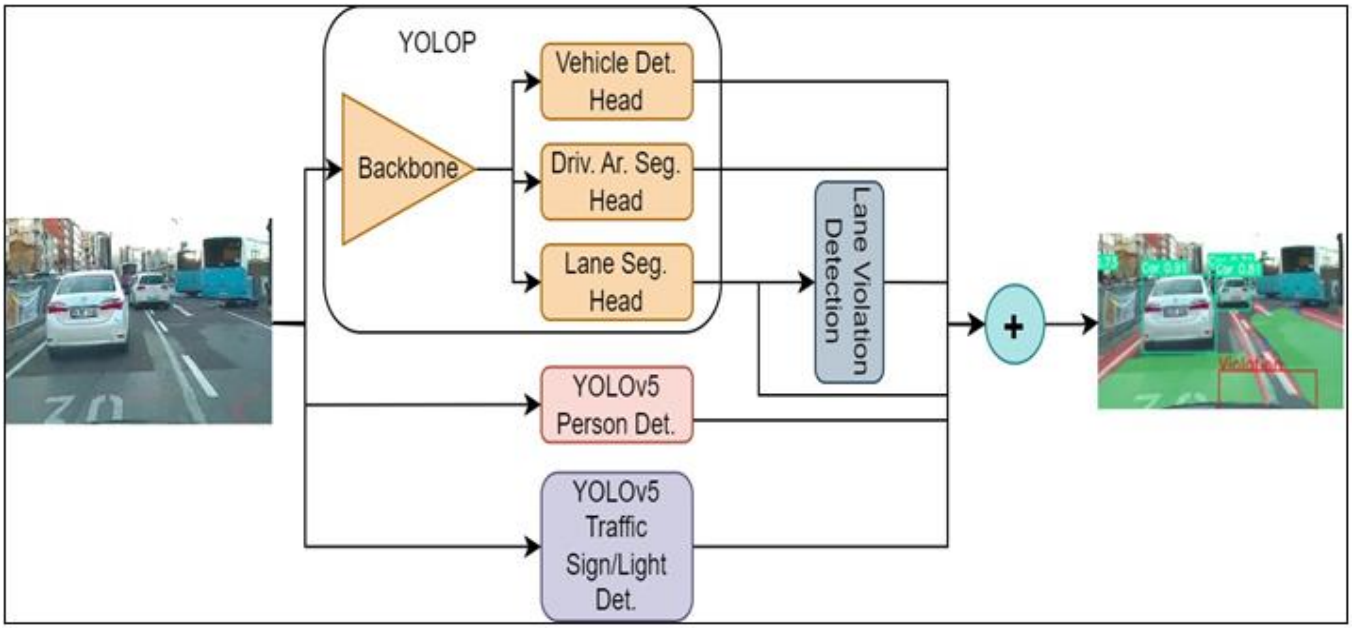

Fig 4 Panoptic Perception and Violation Detection Pipeline

ISSN No:-2456-2165

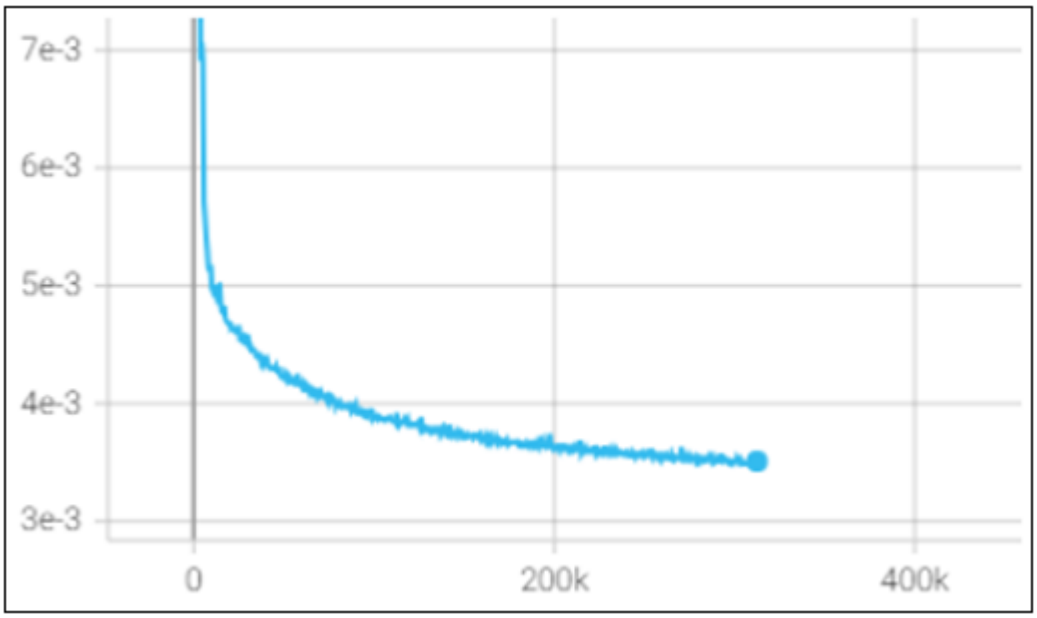

Fig 5 Validation Loss Curve of YOLOP Trained with BDD100K

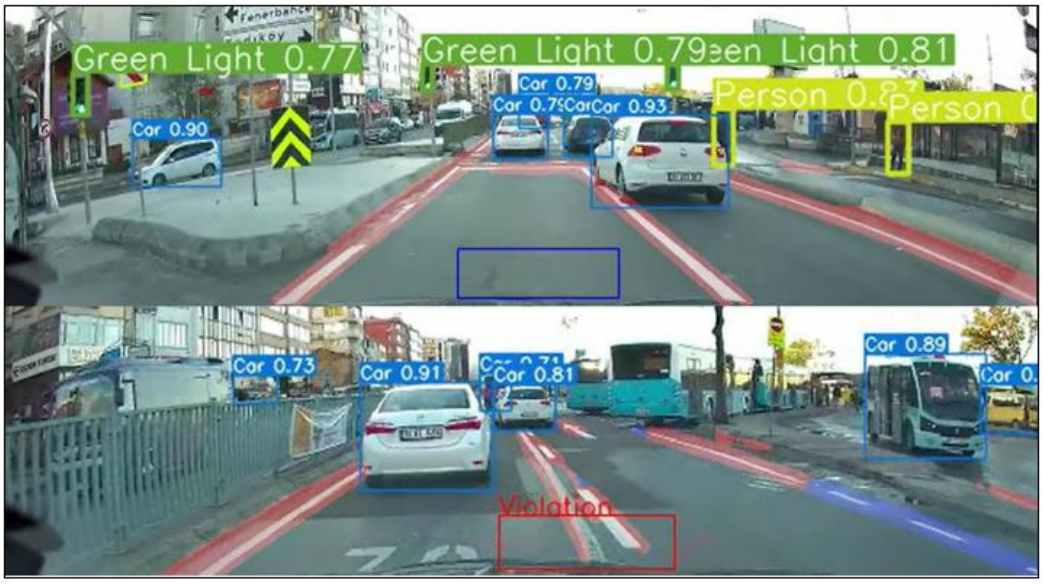

Fig 6 Example Output Frames Containing Lane Segmentation and Vehicle Detection from YOLOP, and Traffic Light and Person Detection from YOLOv5s Models

Segmentation tasks. We implemented 2 temporal smoothing methods that can be utilized in any situation depending on the preference.

➢ *Majority Voting*

Majority voting is a technique used to make decisions based on the majority opinion of a group of voters and is commonly used to make the decisions more reliable. We used the majority voting mechanism to smooth out the small meaningless segments obtained during the model inference. The majority voting mechanism adds up the binary drivable area segmentation results of the last n frames (preferably an odd number, we empirically set n to be 5 on 30FPS video streams), and outputs 1 for the pixels with an addition of more than n/2.

➢ *Probability Voting*

The probability voting mechanism is similar to the majority voting algorithm, however this time instead of the binary predictions, we add up the sigmoid outputs of the last n frames and pick the pixels with an addition more than n/2. This method.

D. Lane Violation Detection Algorithms

We describe a lane violation as the lane change over a straight line, which is a universally forbidden behavior. To detect the lane violations of the driver, we incorporated a methodology where the YOLOP lane segmentation head outputs the lane pixel map, then we put the new output through a rule-based mechanism, which is crafted and calibrated based on the empirical results gathered on unseen data. This section presents the basic algorithms that were crafted to detect lane violations under different situations.

We introduce 2 algorithms with different strengths and weaknesses depending on the outside conditions.

➢ *Center Crop*

For the center cropping method, we first need to define an ROI where under usual conditions, the occurrence of a vertical line segment means that there is a passage of lane. In that regard, we first select a suitable ROI, and then, in each frame, we check the area for the occurrence of straight-line segments, and if there is any, we compare the numbers of pixels coming from the dashed and straight-line segments. If the number of straight- line pixels is higher than some threshold, which is empirically set to be 5% of the RoI's area, and is higher than the number of dashed line pixels, the algorithm yields lane violation. This method is highly intuitive and effective when the input segments are generated properly; this phenomenon will be discussed in detail in the following sections of the paper.

➢ *Centroid Shift Tracking*

Centroid shift tracking was crafted to counteract the possible hard lane segmentation cases, where the line segment is not clear and continuous around the RoI. This method utilizes the temporal information alongside the spatial information.

Table 1 Segmentation Task Results

Model	Dataset	Task	Accuracy	IoU
YOLOP	BDD100K	Dri. Ar. Seg.	0.95	0.89
		Lane Seg.	0.71	0.39
YOLOP	BDD100K+Priv.	Dri. Ar. Seg.	**0.97**	**0.91**
		Lane Seg.	**0.85**	**0.49**

Table 2 Object Detection Task Results

Model	Dataset	Task	Precision	Recall	mAP0.5	mAP0.95
YOLO	BDD100K+Priv.	Car Detection	0.88	0.66	0.63	0.44
YOLOv5	BDD100K	Person Detection	0.80	0.64	0.68	0.45
YOLOv5	Traffic Sign Dataset	Traffic Light/Sign Detection	0.96	0.88	0.91	0.75

Coming from each line segment. The main idea of the algorithm is to track the lane centroid movements in both ego lanes (left and right) which are detected by computing the horizontal position of the centroid inside the predefined RoIs. These regions are selected to best cover the lane segments when the vehicle is going perfectly forward inside the lanes, therefore any major movement of lanes inside these RoIs can be considered as an attempt at a lane change.

With this in mind, the algorithm puts the centroid locations of the lane segments found on the ego lane RoIs and differentiates the horizontal centroid locations, which gives a rough idea about the movement of the vehicle. Especially if the movements of 2 lane centroids are similar, this can imply the movement of the car in either direction; for instance, if both of the centroids move towards the right, it can be assumed that the vehicle moves towards the left, and at some point, either or both lane segments can disappear as the vehicle's position can sit perfectly on the left lane, in which case neither region can enclose a lane segment. The algorithm iteratively checks for such a phenomenon and throws a lane violation event when it captures one.

The algorithm is designed to overcome the possible problems that can occur from the utilization of the Center Crop method, in that the line segments can be inconsistent around the middle of the images, yet this method will not be affected by such inconsistency as the focal points of the line segments are chosen to be original ego line positions. However, as this algorithm utilizes the small centroid movements between consecutive frames, it requires the frames to be as close to each other as possible, i.e. high FPS. However, as the operation is done in the CPU the execution time of the algorithm is rather high therefore it is not suitable for real-time processing problems; on the other hand, if the requirements are suitable for offline processing, meaning that the dashcam videos can be analyzed after the driving session, this algorithm can yield more reliable results when the lane segmentation model underperforms in real-world scenarios.

Finally, it should be noted that to overcome the issue of high processing overhead, we also added a movement detection scheme that is crafted to reduce the amount of redundant computation when the vehicle is stationary. Note that in this case, we assume that our rule-based scheme does not take any sensory data as input that can indicate the movement of the vehicle, therefore we need to perceive the movement through the camera frames also. For that reason, we compute the Structural Similarity Index Measure (SSIM) on another RoI, which is chosen to be the union of the regions selected in the Center Crop and Centroid Shift algorithm, i.e. the front part of the vehicle. Computing SSIM between 2 whole frames is not a viable approach as even when the vehicle is stationary, the surroundings are likely to move, forcing the algorithm to mispredict the occurrence of a movement. The SSIM between the RoIs from 2 consecutive frames is thresholded with a value of 0.9 and if the measurement is less than the threshold, the algorithm decides that there is a significant change between the frames and notifies the occurrence of a movement. Note that even though the SSIM measurement is highly optimized in the package we are using(sci-kit-learn), to reduce the amount of computation, we run the movement detection algorithm once every 10 seconds, and if we do not detect a movement, we do not make any model inference or run any lane detection algorithm.

ISSN No:-2456-2165

E. Experimental Design

As mentioned earlier, we made the experimental setup such that we can use multiple datasets at once and create a model with enhanced perception capability in different datasets and scenarios.

In the first step, we trained a YOLOP model with semantic lane segmentation, drivable area segmentation, and vehicle detection using the BDD100K dataset. Furthermore, we used the focal loss for the class prediction with α and γ parameters being 0.25 and 1.5 respectively, to generate more loss values from harder examples during training. Finally, for the segmentation tasks, the parameters of the focal Tversky loss α, β and γ were set to be 0.5, 0.5, and 0.75 in order. The model was trained for 500 epochs with a batch size of 16. The critical point for the training of a multitask network is the separation of the training procedures; this paradigm has been highly popular in the related fields, therefore in our case, we use this method as well. Firstly, only the encoder and the detection heads are trained over the detection data of BDD100K, therefore we freeze all the parameters of the two segmentation heads. In the following step, we unfreeze the segmentation parameters and freeze the encoder along with the detection head parameters. These two steps utilize separate Adam optimizers with initial learning rates of 0.001, β_1 of 0.937, and β_2 of 0.999. Note that solely freezing the model parameters is not enough in this case, as some of the statistics are learned during the forward pass of the model, such as the mean and standard deviation information of the batch normalization layers. Therefore, unless we freeze these forward statistics along with the model parameters, the outputs received before and after the training can differ, even though technically the parameters of the corresponding layers were not changed. Finally, we unfreeze all the parameters and forward statistics of the model and train the model while lowering the initial learning rate of the Adam optimizer to 0.0001. This can be considered a final fine tune of the overall multitask model. The smoothed validation loss graph of the final training procedure can be seen in Fig. 5.

Furthermore, we annotate the unlabeled data coming from the private dataset using the learned model, more specifically the drivable area segments. The same strategies are also applied during the fine-tuning of the new combined dataset. We also extensively applied data augmentation during the training with the combined dataset to make the model robust against the hard situations that may be encountered in real-world examples. The augmentations used during the training are as follows; random horizontal flip (probability(p)=0.5), hue, saturation, value change (p=0.5), blur (p=0.25), grayscale (p=0.25), histogram equalization (p=0.1), brightness, contrast, gamma change(p=0.2), 1-75% image compression (p=0.75), motion blur(p=0.1), and shift, scale, rotate(p=0.3). We chose the augmentation techniques to simulate the possible situations that can be encountered during a standard traffic session. Fig. 3. shows an example of a frame with very dark lighting, and the performance of the model that was trained on the augmented data. For instance, the camera's light exposure can be simulated by increasing the gamma of the image. In particular, we frequently utilized image compression augmentation to match the possible inputs that can be gathered from low-quality dash cams. We should also point out the intentional usage of the shifting augmentation. We empirically found that when the lane segmentation head is trained over BDD100K, the model grows a noticeable bias about the positions of the lanes, since most of the time the vehicle stays between the lanes, therefore there are very few samples where the lane segments are around the middle of the frame. This phenomenon makes the model struggle to find lane pixels when footage shows a lane change activity. To overcome this issue, we set the shifting augmentation parameters such that a standard frame can imitate a lane change after the transformation. Note that we used the Albumentations package to apply the mentioned augmentations and Torchvision for the normalization.

For the training procedure of the YOLOv5s object detectors, we used the pre-trained weight that is available in the model repository and fine-tuned it with our datasets. We trained both object detectors for 300 epochs with early stopping regularization with patience of 30 epochs. We set the batch size of the training as 32 as the model is relatively small. Different from the training of YOLOP, we used a Stochastic Gradient Descent optimizer with a learning rate of 0.001 and momentum of 0.98, and cosine annealing to overcome overfitting on a relatively small dataset.

IV. RESULTS AND DISCUSSION

The resultant metrics of the object detection models are presented in Table I, which gives the precision, recall, and mean average precisions with intersection over union (IoU) thresholds being 0.5 and 0.95(mAP0.5 and mAP0.95). The car detection head, trained on the combination of BDD100K and our private datasets yields precision and recall values of 0.88 and 0.66 respectively. A high precision and a relatively low recall value usually signify that the model gives predictions relatively rarely, yet the predictions are correct for the most part. This situation translates into high false negative but low false positive rates; this situation can usually be balanced out by reducing the confidence threshold of the non-maximum suppression postprocess (NMS). However, in most real-world AI-based systems, false positives are perceived to be more problematic when compared to false negatives, therefore we increased the NMS's confidence threshold to 0.6 to make sure that the predictions of the model are reliable. In terms of the box regression scores, mAP0.5 and mAP0.95 values are received as 0.63 and 0.44.

Person detection with YOLOv5s on BDD100K gives results very close to car detection. The model again yields high precision and low recall, which is also a consequence of the confidence threshold being set relatively high to reduce false positives on real-world use. The traffic sign and light detection model yields very high results on every metric, as the dataset is relatively simple. The precision value of 0.96 is exceptionally high, pointing out the effectiveness of the method on the target dataset.

The segmentation results are given in Table II, where it is possible to see the pixel classification accuracy and IoU scores of the 2 YOLOP segmentation heads. As mentioned earlier we trained the first model on BDD100K to be able to annotate the drivable area segments of the private dataset. The drivable area pixel classification accuracy of the annotator model is 0.95 and its IoU score is 0.89; these values are rather high and can be considered enough to be used for the annotation of a similar dataset. On the other hand, the lane segmentation scores of the first model are rather low, which makes it unsuitable for an annotation task, however, note that our private dataset's lane segments were annotated by real people. The results of the final finetuned YOLOP model are naturally more promising than its predecessor. In terms of drivable area segmentation, the pixel classification accuracy has increased by 2 percentage points; this situation shows that the model has not only learned the automatically annotated drivable area data but also kept on learning the data coming from BDD100K. Note that this may be a consequence of the separate tasked training paradigm we have incorporated; it is possible that during the final phase of the training procedure, which is the fine tune of all the model parameters, the loss coming from the detection head had more effect on the backbone and the neck, which could hypothetically have harmed the segmentation heads.

The most remarkable improvement is seen in the lane segmentation task, where both the pixel accuracy and the IoU score have increased significantly after the fine tune with the private dataset. The pixel classification accuracy and the IoU score reached values of 0.85 and 0.49 respectively. The total output of lane segmentation and traffic object detection is given in Fig. 4.

V. CONCLUSION

In this work, we have utilized some models from the YOLO family to meet the requirement of building a complete visual perception system on the traffic data, gathered by the installation of dashcams on the windshields of the vehicles. We used YOLOP, which is a multitask network that was inspired by the more popular YOLOv4, for binary drivable area segmentation, semantic lane segmentation for straight and dashed lines, and vehicle detection on the traffic. We started with a state-of-the-art panoptic traffic dataset, BDD100K, and used semi-supervised learning techniques to automatically annotate our private dataset. Furthermore, we used 2 YOLOv5s object detectors for person and traffic light/sign detection. As our eventual aim is to build a system that can detect the erroneous behavior of the drivers, we also introduced two rule- based lane change violation detection algorithms that can work under different conditions and situations. The first algorithm is simple and effective; however, it assumes consistent lane segmentation output in every part of the frame. We also introduced a less efficient algorithm that takes advantage of the frequent positions of the ego lanes, thus more consistent.

This work can be majorly improved in the future by using newer and more capable multitask networks. These models can implicitly utilize the temporal aspect of the video frames to further improve the detection and segmentation performance. We implemented lane violation as a rule-based algorithm as a proof of concept, however, the violations can also be implicitly learned by a machine learning model if the data is annotated to meet such requirements, even though the explainability of the decisions made by the AI models is crucial in these cases.

REFERENCES

[1]. C. Kuo, Y. Lu, & S. Yang, "On the image sensor processing for lane detection and control in vehicle lane keeping systems", Sensors, vol. 19, no. 7, p. 1665, 2019. https://doi.org/10.3390/s19071665J

[2]. J. Shin, E. Lee, K. Kwon, & S. Lee, "Lane detection algorithm based on top-view image using random sample consensus algorithm and curve road model", 2014. https://doi.org/10.1109/icufn.2014.6876735

[3]. C. Low, H. Zamzuri, & S. Mazlan, "Simple robust road lane detection algorithm", 2014. https://doi.org/10.1109/icias.2014.6869550

[4]. D. Neven, B. Brabandere, S. Georgoulis, M. Proesmans, & L. Gool, "Towards end-to-end lane detection: an instance segmentation approach",, 2018. https://doi.org/10.1109/ivs.2018.8500547

[5]. Cheng, W., Wang, X., & Mao, B. (2023). Research on lane line detection algorithm based on instance segmentation. Sensors, 23(2), 789. https://doi.org/10.3390/s23020789

[6]. S. Lee, J. Kim, J. S. Yoon, S. Shin, O. Bailo, N. Kim, T.-H. Lee, H. S. Hong, S.-H. Han, and I. S. Kweon, "VPGNet: Vanishing Point Guided Network for Lane and Road Marking Detection and Recognition," 2017. [Online]. Available: arXiv:1710.06288 [cs.CV].

[7]. R. Bhandari, A. U. Nambi, V. N. Padmanabhan, and B. Raman, "DeepLane: Camera-Assisted GPS for Driving Lane Detection," in Proceedings of the 5th Conference on Systems for Built Environments, BuildSys '18, Shenzen, China, 2018, pp. 73-82, doi: 10.1145/3276774.3276791.

[8]. P. Wang, J. Xue, J. Dou, D. Wang and H. Zhao, "PCRLaneNet: Lane Marking Detection via Point Coordinate Regression," 2021 IEEE Intelligent Vehicles Symposium (IV), Nagoya, Japan, 2021, pp. 1332- 1338, doi: 10.1109/IV48863.2021.9575625.

[9]. H. Lin, J. Dai, L. Wu, & L. Chen, "A vision-based driver assistance system with forward collision and overtaking detection", Sensors, vol. 20, no. 18, p. 5139, 2020. https://doi.org/10.3390/s20185139

[10]. L. Liu, X. Chen, S. Zhu, and P. Tan, "CondLaneNet: a Top-to-down Lane Detection Framework Based on Conditional Convolution," 2023. [Online]. Available: arXiv:2105.05003 [cs.CV].

[11]. D. Wu, M.-W. Liao, W.-T. Zhang, X.-G. Wang, X. Bai, W.-Q. Cheng, and W.-Y. Liu, "YOLOP: You Only Look Once for Panoptic Driving Perception," Machine Intelligence Research, vol. 19, no. 6, pp. 550-562, Nov. 2022. doi: 10.1007/s11633-022-1339-y.
[12]. D. Vu, B. Ngo, and H. Phan, "HybridNets: End-to-End Perception Network," 2022. [Online]. Available: arXiv:2203.09035 [cs.CV].
[13]. C. Han, Q. Zhao, S. Zhang, Y. Chen, Z. Zhang, and J. Yuan, "YOLOPv2: Better, Faster, Stronger for Panoptic Driving Perception," 2022. [Online]. Available: arXiv:2208.11434 [cs.CV].
[14]. F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning," 2020. [Online]. Available: arXiv:1805.04687 [cs.CV].
[15]. G. Jocher, "YOLOv5 by Ultralytics," May 2020. [Online]. Available: https://github.com/ultralytics/yolov5. doi: 10.5281/zenodo.3908559. Version: 7.0. License: AGPL-3.0.
[16]. C.-Y. Wang, H.-Y. M. Liao, I.-H. Yeh, Y.-H. Wu, P.-Y. Chen, and J.-W. Hsieh, "CSPNet: A New Backbone that can Enhance Learning Capability of CNN," 2019. [Online]. Available: arXiv:1911.11929 [cs.CV].
[17]. A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," 2020. [Online]. Available: arXiv:2004.10934 [cs.CV].
[18]. S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path Aggregation Network for Instance Segmentation," 2018. [Online]. Available: arXiv:1803.01534 [cs.CV].
[19]. K. He, X. Zhang, S. Ren, and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," in Computer Vision – ECCV 2014, Springer International Publishing, 2014, pp. 346-361. doi: 10.1007/978-3-319-10578-9_23.
[20]. T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," 2017. [Online]. Available: arXiv:1612.03144 [cs.CV]