01

REALTIME OBJECT & LANE DETECTION FOR AUTONOMOUS VEHICLES

Sai Tarun Sathyan

ABSTRACT

Autonomous driving has become an increasingly popular topic in recent years due to its potential to revolutionize the transportation industry. One of the key challenges in autonomous driving is real-time object detection and lane detection, which require accurate and efficient algorithms to ensure safety on the road.

This project focuses on real-time object detection and lane detection for autonomous cars using YOLO (You Only Look Once) and ultra-fast lane detection techniques. The project utilizes several libraries such as OpenCV, TensorFlow, PyTorch, and CUDA, and was successfully implemented on a GTX 1660Ti. The system can handle video resolutions up to 720p and also works well with webcam and dashcam footage. The primary goal of this project is to find the optimal YOLO model and ultra-fast lane detection method that can operate in real-time. This is an application/reproducibility study that highlights the feasibility of the proposed method for real-time autonomous driving.

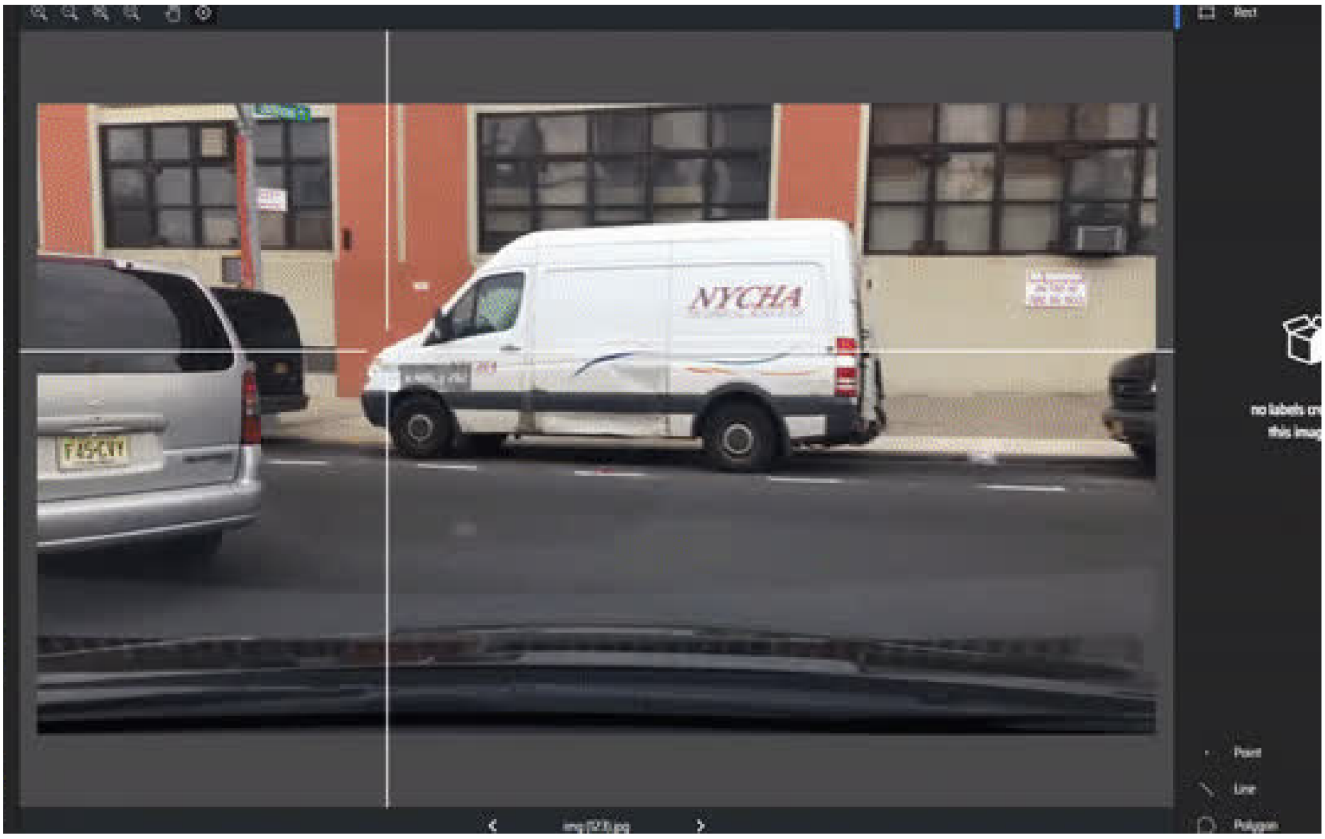
INTRODUCTION

One of the key challenges in autonomous driving is real-time object detection and lane detection, which require accurate and efficient algorithms to ensure safety on the road. This project addresses these challenges by developing a real-time object detection and lane detection system using YOLO and ultra-fast lane detection techniques.

In this project I managed to:

- Create custom labels for my dataset
- Test various yolo models to find which is best for real time
- Test a robust lane detection model that works irl with GTX 1660ti
- Try to combine both and make it work in real time

MANUALLY LABELLING DATASET



The dataset was downloaded from Berkely Deep Drive repository. They have over a 1000 images in their dataset but I used only 500 images. Since I had to manually label the dataset. Yolo models above version 5 have inbuilt data augmentation methods so each image is manupilated multiple times and trained over and over thus ariificially boosting the training set size. The labels I used are as follows:

- Car
- Van
- Truck
- Bus
- Road Sign
- Traffic Signal
- Person

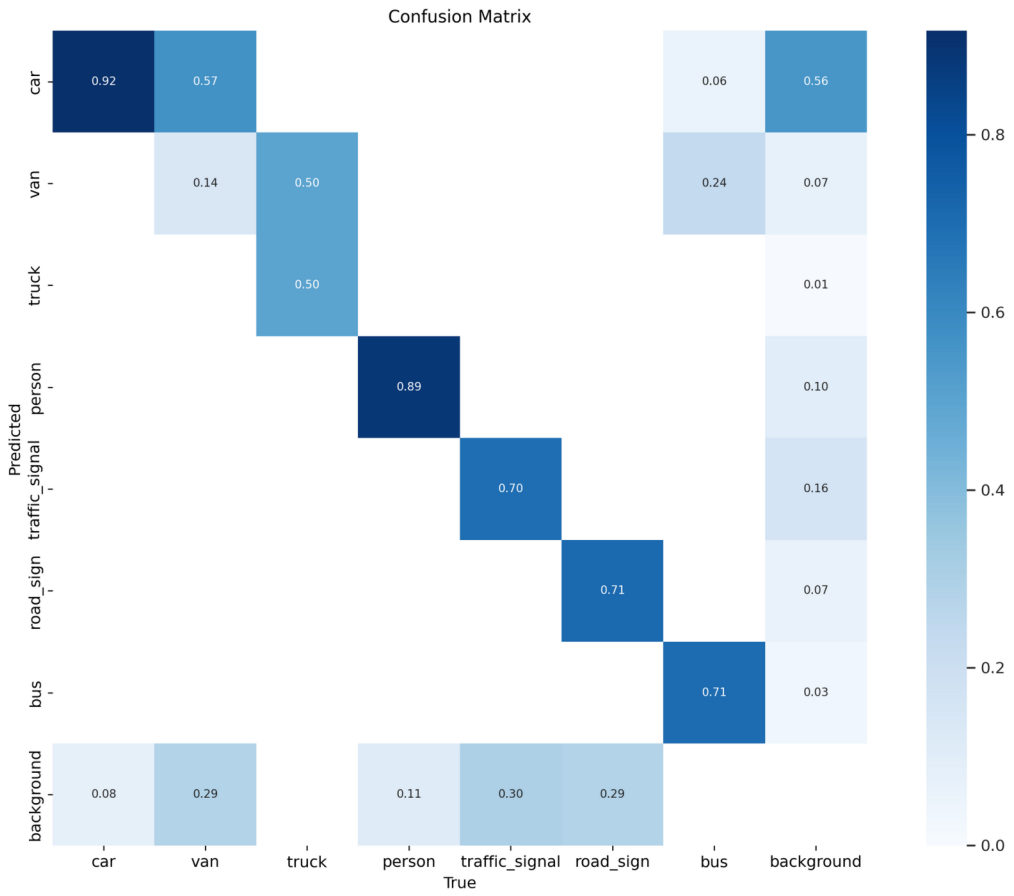
The dataset was labelled on makesense.com which can export your labels in yolo format.

TESTING VARIOUS YOLO MODELS

So the yolo models that caught my attention and I wanted to test out were the following:

- Yolov5
- Scaled Yolov4
- Yolov7
- Tiny Yolov7

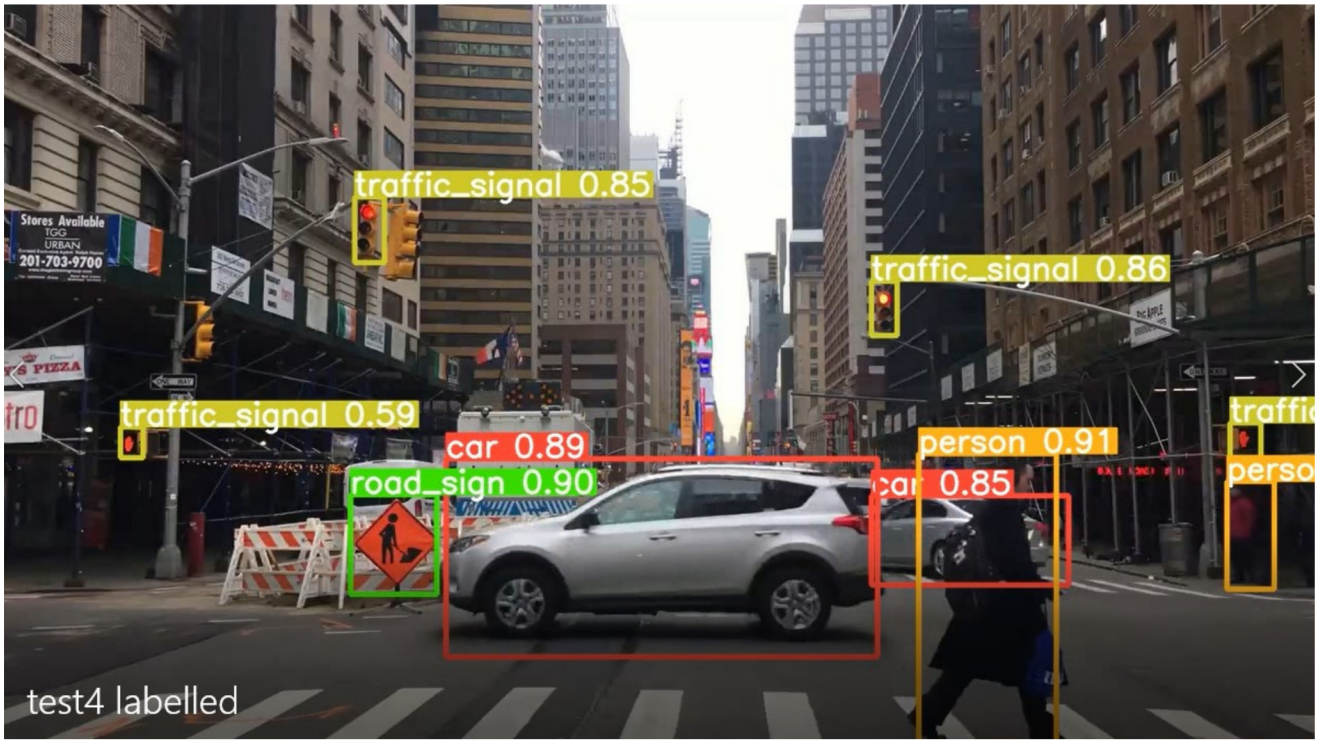
Starting with Yolov5 I trained the model testing various epochs and batch sizes. The most accuracte model was the yolov5s6 trained using 16 batch size and 100 epochs. Producing an MAP50 accuracy of 70%. The yolov5 is known for using less resources compared to Yolov4 but still being more accurate. Which is why I tested this out first but the yolov5 model was not able to keep up when it comes to real time performance and could not even make it to 30FPS.



TESTING VARIOUS YOLO MODELS

YOLOv5 is a popular object detection model developed by Ultralytics. It is an improvement over the previous versions of YOLO, such as YOLOv3 and YOLOv4, with significantly faster inference times and improved accuracy. YOLOv5 is a one-stage object detection model that uses anchor boxes to predict the bounding boxes and class probabilities of objects in an image.

One of the main improvements in YOLOv5 is the use of a novel architecture based on a CSP (Cross-Stage Partial) backbone that enables efficient feature extraction and improves the model's accuracy. YOLOv5 also introduces a new technique called "anchor-free" object detection, which replaces the use of anchor boxes with an offset regression approach to predict object positions. This technique reduces the number of hyperparameters needed for training and simplifies the model architecture.Out of all the models i've trained Yolov5 was the most accurate while using really low resources to train.



Yolov5 Model Detection

TESTING VARIOUS YOLO MODELS

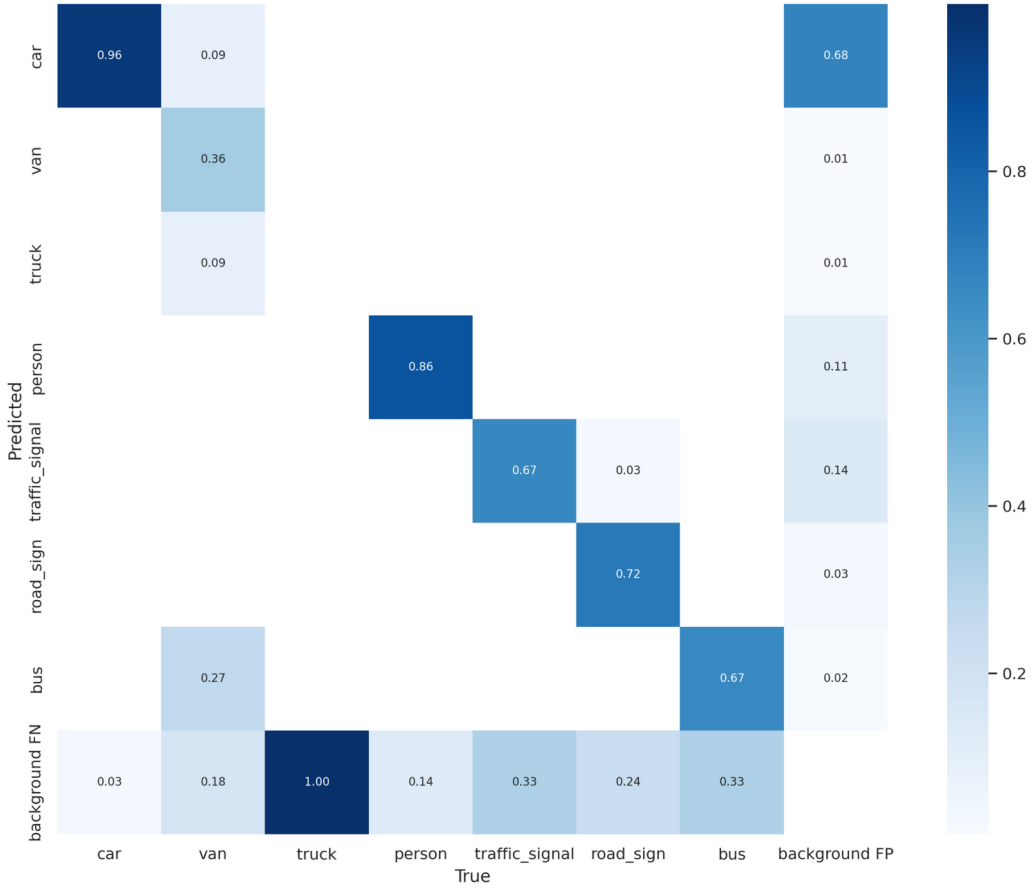
I've decided to skip over scaled yolov4 testing since it has been proven by other researches that Yolov7 beats scaled Yolov4.

Model	#Param.	FLOPs	Size	AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
YOLOv4 [3]	64.4M	142.8G	640	49.7%	68.2%	54.3%	32.9%	54.8%	63.7%
YOLOR-u5 (r6.1) [81]	46.5M	109.1G	640	50.2%	68.7%	54.6%	33.2%	55.5%	63.7%
YOLOv4-CSP [79]	52.9M	120.4G	640	50.3%	68.6%	54.9%	34.2%	55.6%	65.1%
YOLOR-CSP [81]	52.9M	120.4G	640	50.8%	69.5%	55.3%	33.7%	56.0%	65.4%
YOLOv7	36.9M	104.7G	640	**51.2%**	**69.7%**	**55.5%**	**35.2%**	**56.0%**	**66.7%**
improvement	-43%	-15%	-	+0.4	+0.2	+0.2	+1.5	=	+1.3
YOLOR-CSP-X [81]	96.9M	226.8G	640	52.7%	**71.3%**	57.4%	36.3%	57.5%	68.3%
YOLOv7-X	71.3M	189.9G	640	**52.9%**	71.1%	**57.5%**	**36.9%**	**57.7%**	**68.6%**
improvement	-36%	-19%	-	+0.2	-0.2	+0.1	+0.6	+0.2	+0.3
YOLOv4-tiny [79]	6.1	6.9	416	24.9%	42.1%	25.7%	8.7%	28.4%	39.2%
YOLOv7-tiny	6.2	5.8	416	**35.2%**	**52.8%**	**37.3%**	**15.7%**	**38.0%**	**53.4%**
improvement	+2%	-19%	-	+10.3	+10.7	+11.6	+7.0	+9.6	+14.2
YOLOv4-tiny-3l [79]	8.7	5.2	320	30.8%	47.3%	32.2%	**10.9%**	31.9%	51.5%
YOLOv7-tiny	6.2	3.5	320	**30.8%**	**47.3%**	**32.2%**	10.0%	**31.9%**	**52.2%**
improvement	-39%	-49%	-	=	=	=	-0.9	=	+0.7
YOLOR-E6 [81]	115.8M	683.2G	1280	55.7%	73.2%	60.7%	40.1%	**60.4%**	69.2%
YOLOv7-E6	97.2M	515.2G	1280	**55.9%**	**73.5%**	**61.1%**	**40.6%**	60.3%	**70.0%**
improvement	-19%	-33%	-	+0.2	+0.3	+0.4	+0.5	-0.1	+0.8
YOLOR-D6 [81]	151.7M	935.6G	1280	56.1%	73.9%	61.2%	**42.4%**	60.5%	69.9%
YOLOv7-D6	154.7M	806.8G	1280	56.3%	73.8%	61.4%	41.3%	60.6%	70.1%
YOLOv7-E6E	151.7M	843.2G	1280	**56.8%**	**74.4%**	**62.1%**	40.8%	**62.1%**	**70.6%**
improvement	=	-11%	-	+0.7	+0.5	+0.9	-1.6	+1.6	+0.7

In comparison with YOLOv4, YOLOv7 reduces the number of parameters by 75%, requires 36% less computation, and achieves 1.5% higher AP (average precision). Compared to the edge-optimized version YOLOv4-tiny, YOLOv7-tiny reduces the number of parameters by 39%, while also reducing computation by 49%, while achieving the same AP.

TESTING VARIOUS YOLO MODELS

The most accuracte Yolov7 model was trained using a batch size of 8 and 100 epochs. It MAP50 accuracy of 60%.



In comparison with YOLOv4, YOLOv7 reduces the number of parameters by 75%, requires 36% less computation, and achieves 1.5% higher AP (average precision). Compared to the edge-optimized version YOLOv4-tiny, YOLOv7-tiny reduces the number of parameters by 39%, while also reducing computation by 49%, while achieving the same AP.

Still this was not capable of performing in realtime, It was slow and could not function well on my Laptop.

TESTING VARIOUS YOLO MODELS

Finally I moved over to Tiny Yolov7 which is specifically made for performance, but the accuracy metric was questionable. So I tested it with various hyperparameters for training. Finally I found that having 28 batch size and 100 epochs gives the best accuracy for Tiny Yolov7. It MAP50 accuracy of 58%.

Tiny Yolov7's performance was comparable with the accuracy of the regular Yolov7 here. There were a few miscalssifications but when testing in real time it worked well. The FPS was over 30FPS on my computer which had a GTX 1660ti and Intel i7.

REAL TIME LANE DETECTION

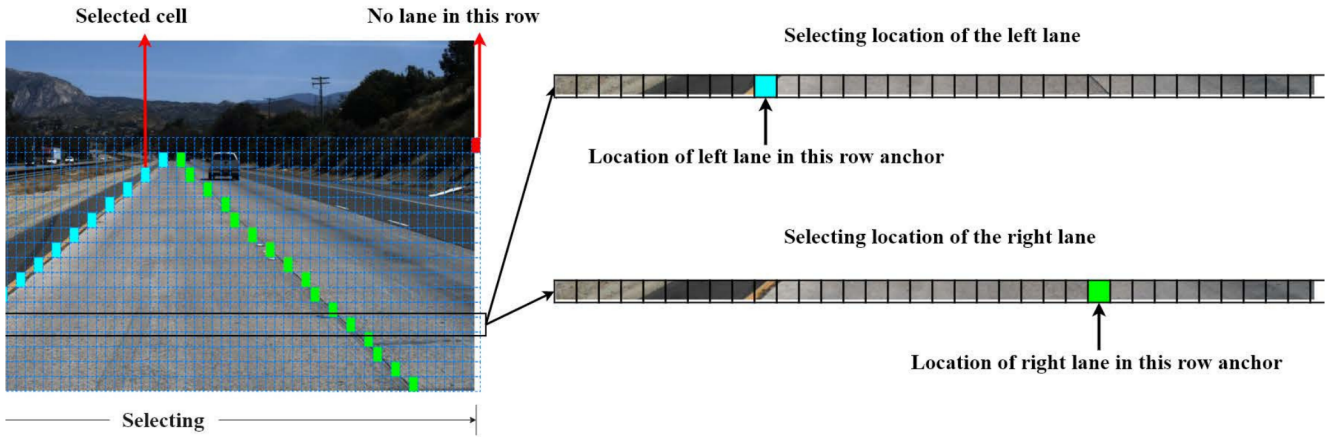
traditional computer vision techniques such as Hough transforms or edge detection to identify lane boundaries. These methods are less computationally intensive than CNNs but may require more manual tuning of parameters to achieve accurate results.

But the problem with hough transform is that it breaks even when theres a slight bend or curve in the road. Which is a big issue when working in real world since most roads are not perfectly straight. So I had to use a ML algorithm that is capable of working in realtime and being accurate as well.

REAL TIME LANE DETECTION

For the real time lane detection I used the model proposed and trained by this paper : https://arxiv.org/abs/2004.11757. Ultra Fast Structure-aware Deep Lane Detection is a paper which proposed method that is based on a deep convolutional neural network (CNN) that is trained to predict lane boundaries in images. The network architecture consists of an encoder-decoder structure that uses a series of convolutional and deconvolutional layers to extract features from the input image and generate a pixel-wise lane boundary prediction. The model proposed does the following:-

- The method incorporates structural priors that capture the expected geometry of lane boundaries, such as their shape, width, and position relative to each other, to guide the network's predictions.
- The proposed method uses a sparse set of anchor points to represent the lane boundaries, which reduces the amount of data needed for training and inference and enables real-time performance on low-power embedded devices.
- The authors also use a lightweight network architecture and optimize the implementation for parallel processing on GPUs to further improve the speed of the method.



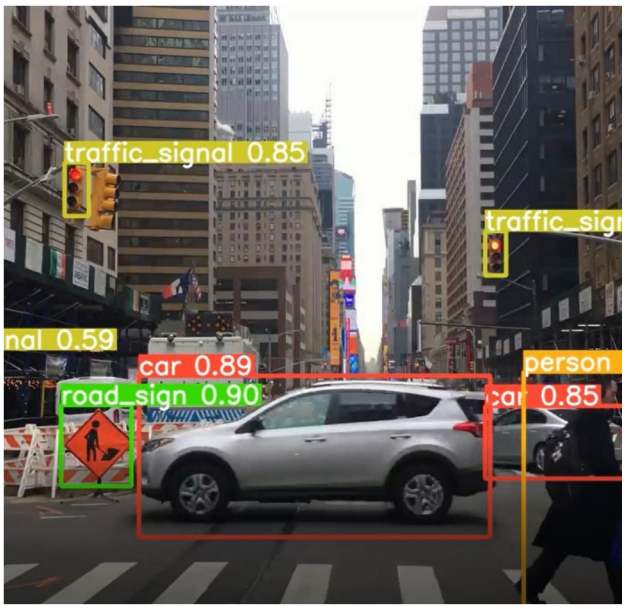
This method breaks down the road into a grid and marks both sides of the lanes with points. So we can use these points to fill in this polygon and produce a output image marking the lane the car is on right now.

REAL TIME LANE DETECTION

This method is not only capable of working in real time. But it also does not really hinder the performance of the object detection model. Thus it is capable of maintaing the FPS over 30. This model uses the GPU as well. I am still working on fine tuning the models parameters to be more accuracte but so far it is doing a pretty good job of dynamically tracking the lane.

CONCLUSION

This project demonstrates the successful implementation of real-time object detection and lane detection for autonomous cars using a custom-trained tiny YOLOv7 model and the Ultra Fast Structure-aware Deep Lane Detection method. The models were chosen after a thorough evaluation and comparison with other models based on various metrics. The project achieved a performance of 30fps+ on 720p resolution video, showcasing the potential for cost-effective implementation of autonomous driving systems and improved road safety through optimized model selection and implementation.



FUTURE SCOPE

01 Real-time performance on higher resolutions

02 Improving accuracy

03 Dynamic object detection

04 Adaptation to different environments

05 Hardware optimization