SELF DRIVING CARS
USING COMPUTER VISION

PROBLEM STATEMENT:

Self-driving vehicles are a cutting-edge technology that is set to revolutionize mobility as we know it. These self-driving cars have the potential to provide a more secure and effective means of transit. However, the growth of self-driving vehicles requires careful assessment and examination of the technology's various components. There are several components of self-driving cars. The problem statement of the project was to examine four critical components of self-driving cars, namely, vehicle detection, lane line detection, traffic light classification, and road segmentation using Classic Computer vision techniques, Machine learning methods and Deep Learning models. Our main goal was to learn how computer vision works, how it can be used with Machine Learning, Deep Learning and how a single problem, for example: a Segmentation or Detection problem, is divided into multiple sub problems. Although each group member had a specific role to fulfill, the completion of the project was a collective endeavor.

ABOUT THE DATASET:

a) Vehicle Detection:

This code makes use of a video called 'vehicle_movement.mp4', which includes moving vehicles that must be identified. The algorithm recognizes the cars in the footage and draws boxes around them. The output is a new video file called 'output_new.mp4' that contains the detected cars highlighted.

b) Lane Line Detection:

The dataset was referred to from various sources like Wikipedia, GitHub links and a total of 30 test images were taken and stored in a folder named test_images. These images are that of roads with white and yellow lane lines and have been made use in code for purposes like finding region of interest, Hough transform, etc.

c) Traffic Light Classification:

The dataset was taken from GitHub. The dataset comprised train and test images. In train and test images, there were sub folders on green, blue and red. These subfolders furthermore comprised images of traffic light with the corresponding traffic color (named as per the file). There was a total of 949 samples in the training file, the dimension of each image was 64 x 64 pixels of 3 color channels. The test set had 238 samples; the dimension of each image was 64 x 64 pixels of 3 color channels.

d) Road Segmentation:

The dataset was taken from Kaggle. This dataset contains pictures of data and labelled semantic segmentations taken by the CARLA self-driving vehicle simulator. The information was gathered as part of the Lyft Udacity Challenge. This dataset can be used to teach ML systems to recognize semantic segmentation of vehicles, highways, and other objects in images. The data consists of five groups of 1000 pictures with associated labels.

OUR APPROACH:

Methodologies followed-

a) Vehicle Detection (Basic Level):

A .mp4 video file was given as input to the code. Pre-trained YOLO4 model was used to perform the task. First, we imported the necessary libraries and loaded the pre-trained YOLO4 model for vehicle detection. Weights and configuration files were used for this purpose. OpenCV's VideoCapture function read the input video file. An output video file was created using OpenCV's VideoWriter function. Information like frame rate, height, width and input video length were obtained. A region of interest was then defined using the width and height of the video. Then we initialized a counter to count the number of vehicles detected. Each frame of the input video was read using the while loop and then preprocessed using OpenCV's dnn.blobFromImage function.

b) Lane Line Detection (Basic Level):

In this case, we used image processing methods such as color filtering, image thresholding, and edge detection. To receive, manipulate, and display images, packages and tools such as matplotlib, numpy, opencv, and glob were used. A collection of test images was first loaded and displayed. To separate white and yellow lane lines from the pictures, a color filter based on HSV and HSL color spaces was used. The resulting masked images were obtained in the result. Following that, we attempted to identify lane lines in the masked images by converting them to grayscale and using Gaussian blur to reduce noise. The Canny edge recognition algorithm was used to identify edges. Lane lines were extracted again.

c) Traffic Light Classifier (Basic Level):

Here, we performed traffic classification using machine learning models. Images of traffic lights were taken as input and classified into three categories, namely green, red, and yellow. The code was been divided into different sections, each of which performed a specific task. The required libraries such as OpenCV, os, numpy, PIL, scikit-learn, and Machine learning algorithms for classification were imported.The images present in the subfolders of the folder "traffic_light_images" were read. The subfolders were "training" and "test" respectively, and within each of these subfolders, three folders were present. They were based on the traffic light color and were named green, red, and yellow. The image size was set to (64, 64) for the models to be trained. The images are resized to this size and then converted into numpy arrays. The images were appended to the list 'data', and the respective labels are appended to the list 'labels'. The number of images in each category is printed to check the balance in the dataset. The data was split into training and testing sets using train_test_split from sklearn. We did exploratory data analysis and got the shapes of the training and testing sets and the labels present in them. Then, the models to be trained and tested were defined in a dictionary, and the code trains and tests each model in the dictionary. The models are SVM, Random Forest, K-NN, and Naive Bayes.The models are then tested on the testing data, and their accuracies are printed. Then, the code read the test images and uses one of the trained models (in this case, Naive Bayes) to predict the category of each test image. We got in result the prediction and the name of the test image file.

Similarly, we tried to predict the category of the test images using SVM and K-NN models and got the correct prediction in output. Overall, the code uses machine learning models to classify traffic light images into three categories based on the color of the light. In other words, we took the input images, developed and refined a model based on these images, assessed the accuracy of the model on a test dataset, and ultimately used the trained model to classify new images into the correct traffic light category. We tried covering one aspect of the traffic light classification, that is, detecting the color of the traffic light correctly given the training and test samples. These samples were in the form of .jpg images.

d) Road Segmentation (Basic Level):

We had a dataset of images and their corresponding segmentation masks. The segmentation masks masked the pixels of the image that correspond to the objects we were interested in. At first, necessary libraries were imported for preprocessing the images. Then, the dataset of images and segmentation masks were read and stored in separate lists. Data was preprocessed as a part of preparation phase before the model was trained. The values of pixel were normalized in the range [0,1] and the images & their corresponding masks were resized to a fixed width and height. The dataset was split into training, test and validation sets. U-Net model architecture was defined using the keras library. The model had two paths: contracting and expanding. The contracting path was used to capture the content and create a more abstract representation of the image while the expanding path was used to recover details of objects which may be lost during the contracting path. The output of the model was a probability map where every pixel was assigned value between 0 and 1. At last, the model was compiled and trained on the training set using Adam Optimizer and binary cross-entropy loss function. The training process was monitored using early stopping and learning rate reduction. The challenge we faced here was there were limitations in GPU and compute power resources. The model was supposed to be trained for 100 epochs with a batch size 16 but it took a considerable amount of time even for one epoch to run. The code stopped running, if it was not monitored every 3-4 minutes. Due to the long duration of training, it was challenging to monitor the progress of the training process.

Models used:

Portion of the project	**Models/ Algorithms/ Computer Vision techniques used**
Vehicle Detection	YOLOv4 for Vehicle Detection
Lane Line Detection	Computer vision concepts including color filtering, image masking, image thresholding, and edge detection
Traffic Light Classifier	Machine Learning models like SVM, Naïve bayes, Random Forests, kNN
Road Segmentation	Deep learning techniques like Convolutional Neural Networks and Computer vision techniques like Image augmentation

Results:

a) Vehicle Detection (Basic Level):

Excerpt from Input video –

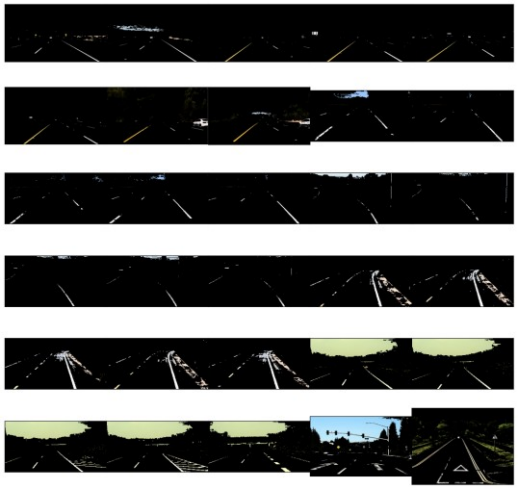
Excerpt from Output Video-

b) Lane Line Detection (Basic Level) –

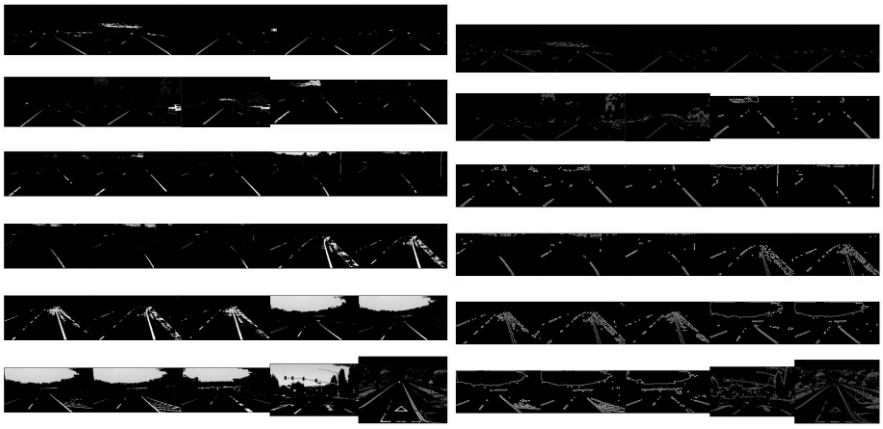
Input Images

HSL select lane lines

Color filtered images



Canny edge detection

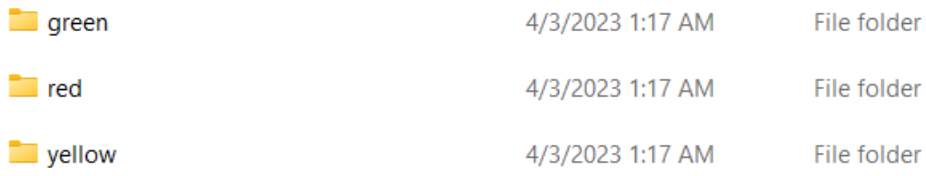


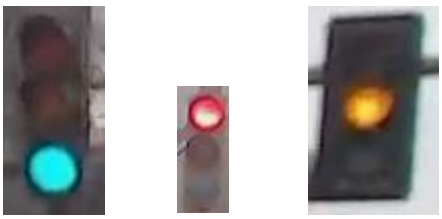
Hough transform

c) Traffic Light Classifier:

Input

```
traffic images/
├── train/
│   ├── green/
│   ├── red/
│   └── yellow/
└── test/
    ├── green/
    ├── red/
    └── yellow/
```



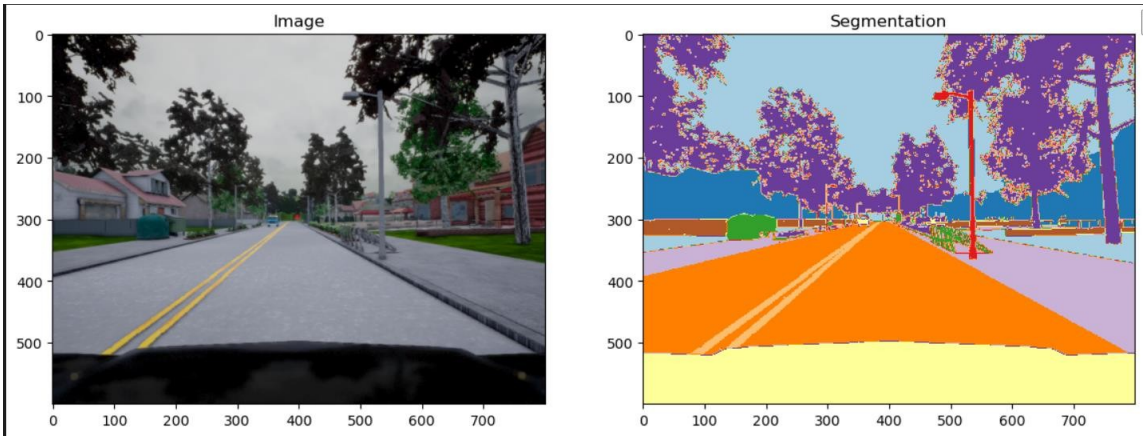


Output

If given the test folder based on the directory structure, it predicted whether the images present in the certain folder were green, blue or yellow for all the models

```
Prediction for the last trained model - in our case NAIVE BAYES:  traffic_light_images\test\green\00febbe1-a9ae-4b5f-b682-8ebfdae485a3.jpg : green
Prediction for the last trained model - in our case NAIVE BAYES:  traffic_light_images\test\green\01ae3c3d-21c8-4711-853a-ba6fda9553bf.jpg : green
Prediction for the last trained model - in our case NAIVE BAYES:  traffic_light_images\test\green\06ab17cd-888e-45de-9482-b432e23cea6a.jpg : green
Prediction for the last trained model - in our case NAIVE BAYES:  traffic_light_images\test\green\0a251f2f-babd-4257-87be-2940bd4bacc8.jpg : green
Prediction for the last trained model - in our case NAIVE BAYES:  traffic_light_images\test\green\0ab8c5a1-a750-4137-ad0a-13e5da55bd09.jpg : green
Prediction for the last trained model - in our case NAIVE BAYES:  traffic_light_images\test\green\0acf7bce-03d1-46af-aac2-bbca22e1c1cb.jpg : green
Prediction for the last trained model - in our case NAIVE BAYES:  traffic_light_images\test\green\0ae63733-c453-42b2-89cf-e4a7877feda6.jpg : green
Prediction for the last trained model - in our case NAIVE BAYES:  traffic_light_images\test\green\0aec12bd-ce95-4400-937b-ece1fd7c3011.jpg : green
Prediction for the last trained model - in our case NAIVE BAYES:  traffic_light_images\test\green\0b2b43f5-cad6-41a3-80ab-baf02e9f7ff1.jpg : green
Prediction for the last trained model - in our case NAIVE BAYES:  traffic_light_images\test\green\0b3606b7-bf9e-49d8-8de8-801bb8374b2d.jpg : green
Prediction for the last trained model - in our case NAIVE BAYES:  traffic_light_images\test\green\0b534a19-781b-4883-8005-4dce2478e772.jpg : green
Prediction for the last trained model - in our case NAIVE BAYES:  traffic_light_images\test\green\0b602dec-60ad-4601-ba50-30850e9b6895.jpg : green
Prediction for the last trained model - in our case NAIVE BAYES:  traffic_light_images\test\green\0bbaa6ca-9f78-4cad-8c22-b95648fb90be.jpg : green
Prediction for the last trained model - in our case NAIVE BAYES:  traffic_light_images\test\green\0bc8252c-d3f6-4ef4-b4aa-d9f46f28f137.jpg : green
Prediction for the last trained model - in our case NAIVE BAYES:  traffic_light_images\test\green\0be91609-32ba-4b7b-b460-cc47dd62b740.jpg : green
Prediction for the last trained model - in our case NAIVE BAYES:  traffic_light_images\test\green\0bf347bb-50fe-46ce-86f4-cdd42538c495.jpg : green
Prediction for the last trained model - in our case NAIVE BAYES:  traffic_light_images\test\green\0c3c3f67-02b9-4a7b-beb9-87600beee657.jpg : green
Prediction for the last trained model - in our case NAIVE BAYES:  traffic_light_images\test\green\0c776797-c240-4fea-a26a-05b36fbc5ca1.jpg : green
Prediction for the last trained model - in our case NAIVE BAYES:  traffic_light_images\test\green\0da38382-3b5b-4114-b54f-c706269e4a34.jpg : green
Prediction for the last trained model - in our case NAIVE BAYES:  traffic_light_images\test\green\0e41f866-a861-415a-b9de-0ed37472816d.jpg : green
Prediction for the last trained model - in our case NAIVE BAYES:  traffic_light_images\test\green\0e470b16-71f2-471c-8b31-a21f5ab4d814.jpg : green
Prediction for the last trained model - in our case NAIVE BAYES:  traffic_light_images\test\green\0ed20f0b-04f4-49f4-ac86-5e10944bb7d8.jpg : green
Prediction for the last trained model - in our case NAIVE BAYES:  traffic_light_images\test\green\0f238957-63d2-434a-ab6b-e39169acb140.jpg : green
Prediction for the last trained model - in our case NAIVE BAYES:  traffic_light_images\test\green\0f3a0257-e123-48aa-ac0b-445bf7043183.jpg : green
Prediction for the last trained model - in our case NAIVE BAYES:  traffic_light_images\test\green\0fdeb7c8-b5a7-487a-864a-bebeead644c0.jpg : green
...
Prediction for the last trained model - in our case NAIVE BAYES:  traffic_light_images\test\yellow\4ba8fe29-6d69-4c32-be18-eb3ec73cb3de.jpg : yellow
Prediction for the last trained model - in our case NAIVE BAYES:  traffic_light_images\test\yellow\5af79a62-defd-4593-b058-4cdc2f4176aa.jpg : yellow
Prediction for the last trained model - in our case NAIVE BAYES:  traffic_light_images\test\yellow\5d309b84-aef3-4098-a14d-5cb05f5821c9.jpg : yellow
Prediction for the last trained model - in our case NAIVE BAYES:  traffic_light_images\test\yellow\6cfdbbbc-b63b-4dcf-bd16-57e06fdf51a9.jpg : yellow
```

d) Road Segmentation (Basic Level) –



Conclusion:

The project's goal is to create a self-driving car system with various components such as vehicle recognition, lane line detection, traffic signal categorization, and route division. These components are critical for autonomous driving because they provide the car with knowledge about its surroundings and allow it to make educated choices. They have been implemented at a basic level in the project using various technologies such as the pre-trained YOLO4 model for vehicle detection, image processing methods for lane line detection, machine learning models for traffic light classification, and U-Net model architecture for road segmentation. However, while training the U-Net model for Road Segmentation, the team encountered obstacles such as restricted GPU and processing capacity resources. Our future goal for this project would be to work further on it to completely implement it as well as improve the accuracy of the models and algorithms used. Once the accuracy of the models is improved, we would focus on using advanced techniques of Computer Vision for Vehicle Detection, Road Segmentation, Traffic Light Classification and Road Lane Line Detection.