Comprehensive Autonomous Driving System

Mentor: Mr Nitig Singh

Team Leads: Faraaz Ahmed, Dyna Joshy, Rameshwari Vadhvani, Sai Santhosh

Team Co-lead: Srinithi Jaikumar, Sri Jishnu, Raju Kumar

Team Members: Nitesh Kumar, Kishore Kumar, Parvinder Kumar, Adars, Rakshita, Abhishek, Shivanand, Pnv Sai, Deekshitha, Anirwin.

TABLE OF CONTENTS

01 Introduction

02 Problem definition

03 About Dataset

04 Data preparation/ preprocessing

05 Model Selection

06 Training

07 Challenges

08 Conclusion

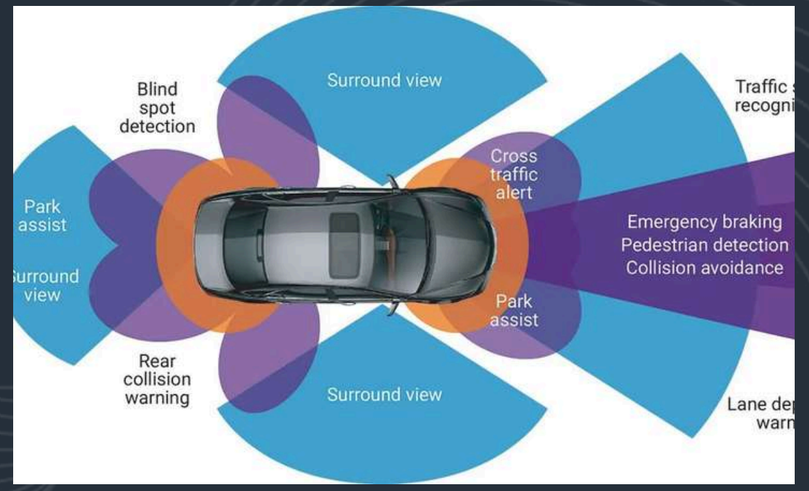
Overview of Autonomous Driving

Autonomous driving technology enables vehicles to operate and navigate without human intervention by using AI, sensors, and cameras. The system continuously analyzes its surroundings, identifying objects, pedestrians, and road conditions to make safe, real-time driving decisions.

Levels of Autonomy ranges from 1 to 5.

Vision Statement

Enhance road safety
Promote efficiency
Inclusive mobility
Support environmental sustainability within smart cities



Mission and Objectives

- Develop a web application for real-time autonomous driving video analysis and detection.
- Enable uploading and processing of driving videos to identify key elements like traffic signs, lanes, objects, and traffic lights.
- Ensure accurate detection using pre-trained models optimized for urban environments.
- Improve user experience with a simple, intuitive interface and efficient processing

Motivation

- **Driving into the Future**
- **Sustainable Mobility through Automation**:
- **Enhanced Road Safety**
- **Reduced Traffic Congestion**
- **Increased Accessibility**

ELON MUSK

A significant portion of traffic accidents are due to human error. With autonomous driving, we have the potential to save millions of lives over time

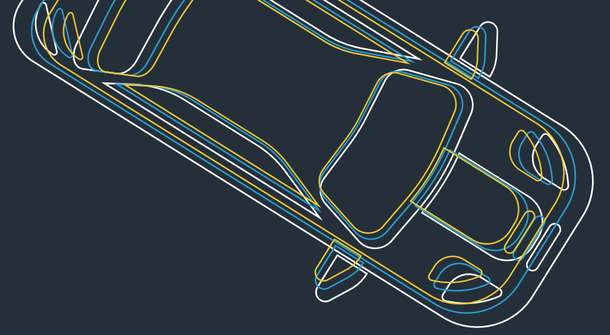
SUNDAR PICHAI

“Self-driving cars will make transportation safer, more accessible, and more efficient.”

Case Studies

- **Case Study 1: Waymo – Autonomous Ride-Hailing Service**: Waymo demonstrates the feasibility of self-driving technology, providing a safe and reliable ride-hailing service in Phoenix without human drivers.
- **Case Study 2: Tesla Autopilot – Driver Assistance to Full Self-Driving**: Tesla's Autopilot system enhances driver convenience and safety, progressing towards full autonomy with features like lane-keeping and adaptive cruise control.
- **Case Study 3: Autonomous Trucks in Mining**: Autonomous trucks in Brønnøy Kalk Mine prioritize safety and efficiency, navigating challenging terrains while reducing operational costs through automation.


Problem Statement



DESIGN A SYSTEM APPLICATION FOR AUTOMOBILES BASED AI TECHNIQUE WHICH CAN

- OBJECT DETECTION AND RESPONSE
- LANE DETECTION AND RESPONSE
- OBJECT DIFFERENTIATION AND SEGMENTATION
- TRAFFIC SIGN AND SIGNAL DETECTION
- REAL-TIME PROCESSING VIA WEB APPLICATION
- REAL-WORLD CHALLENGES

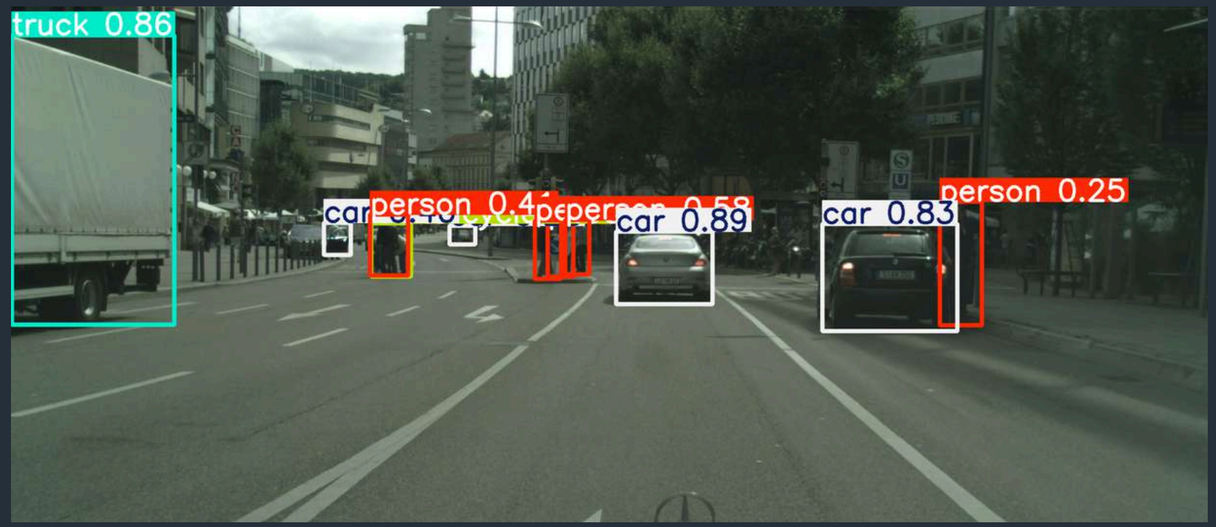
Introduction

Autonomous driving is transforming transportation by enabling vehicles to operate without human input. This project develops an AI-driven system for key tasks like object detection, lane tracking, traffic sign recognition, and semantic segmentation.

The goal is to enhance road safety, optimize traffic, and promote sustainable, intelligent mobility using advanced computer vision and machine learning techniques.

MODULES TO BE INTEGRATED IN ACCORDANCE TO GOALS

- **Object Detection**



- **Lane Detection**

- **Semantic Segmentation**

- **Traffic signal Recognition**

About the Datasets used

COCO 2017 Dataset

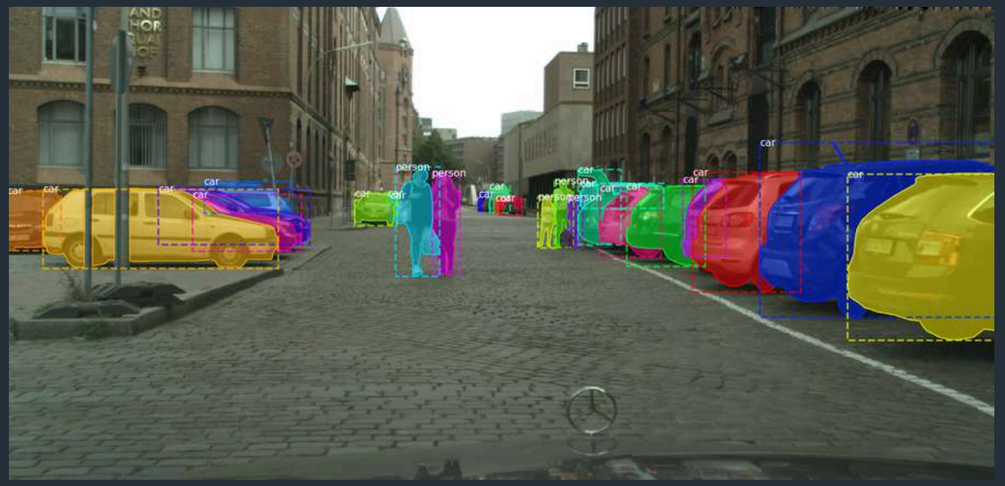
- Contains over 330,000 annotated images for object detection, segmentation, and captioning.
- Widely used benchmark for improving model robustness across diverse object categories.

Cityscapes Dataset

- Features 5,000 high-quality images with semantic segmentation annotations.
- Focused on urban environments, ideal for autonomous driving applications.

GTSRB Dataset

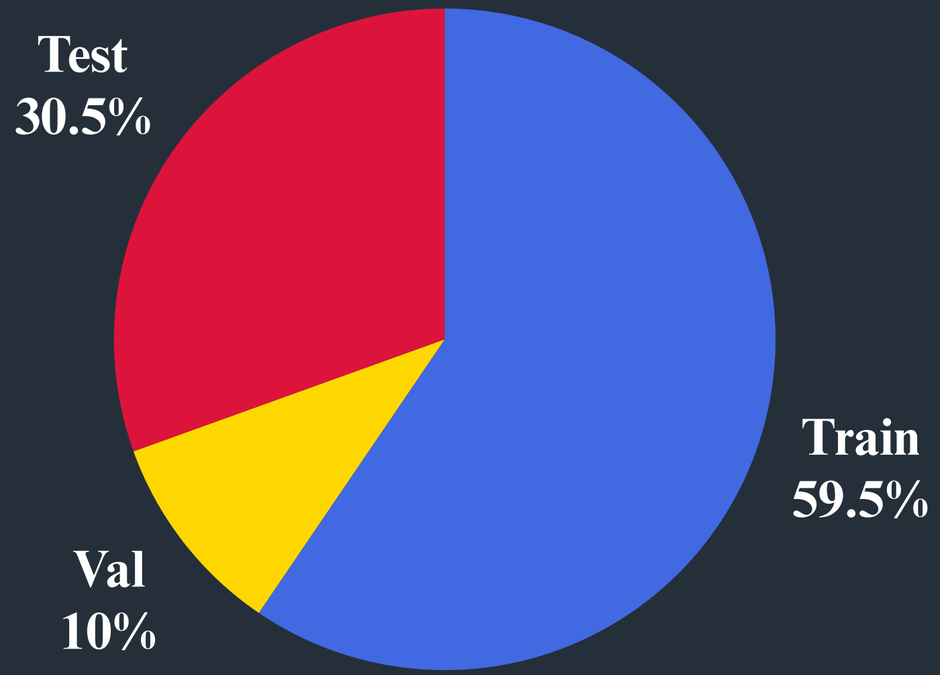
- Provides over 50,000 labeled images of 43 traffic sign classes.
- Essential for developing reliable traffic sign recognition systems.



Data Composition/Preprocessing Overview

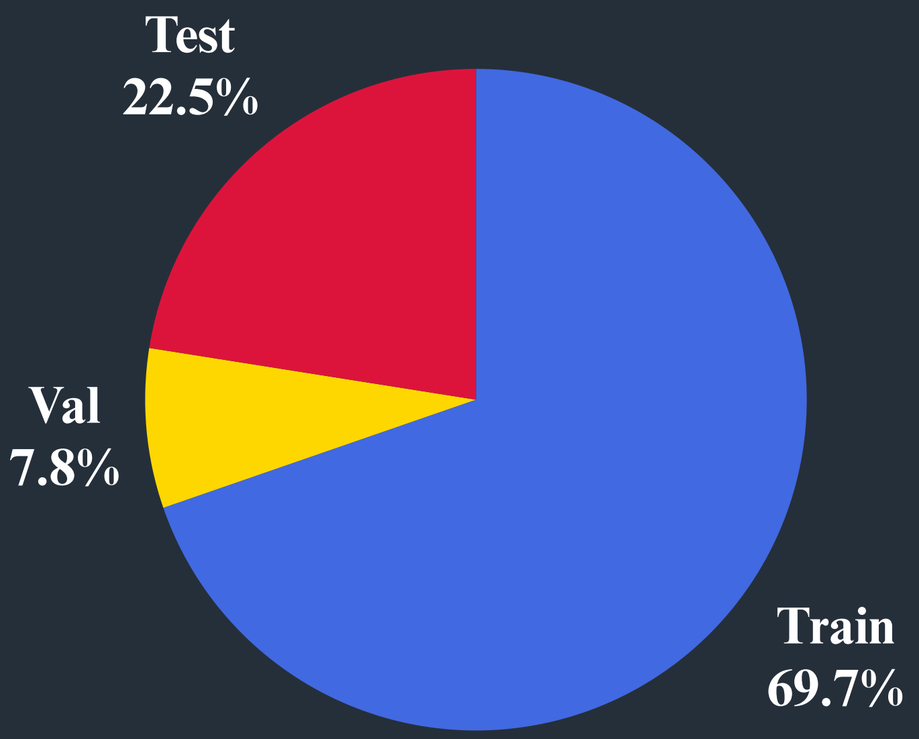
Data Composition Analysis of Cityscapes Dataset

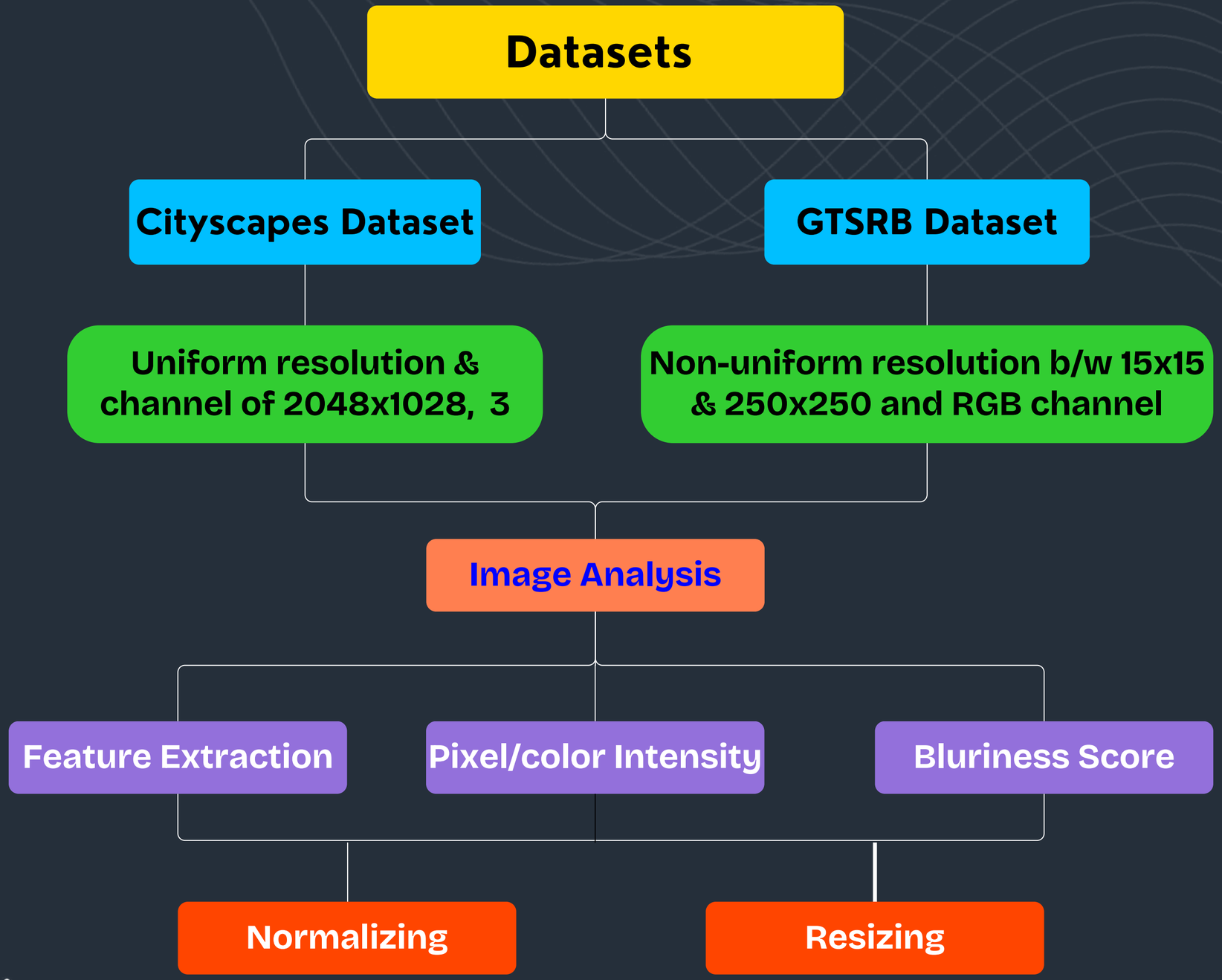
The dataset comprises **5000 images** split into training, validation, and test sets in the ration as shown below

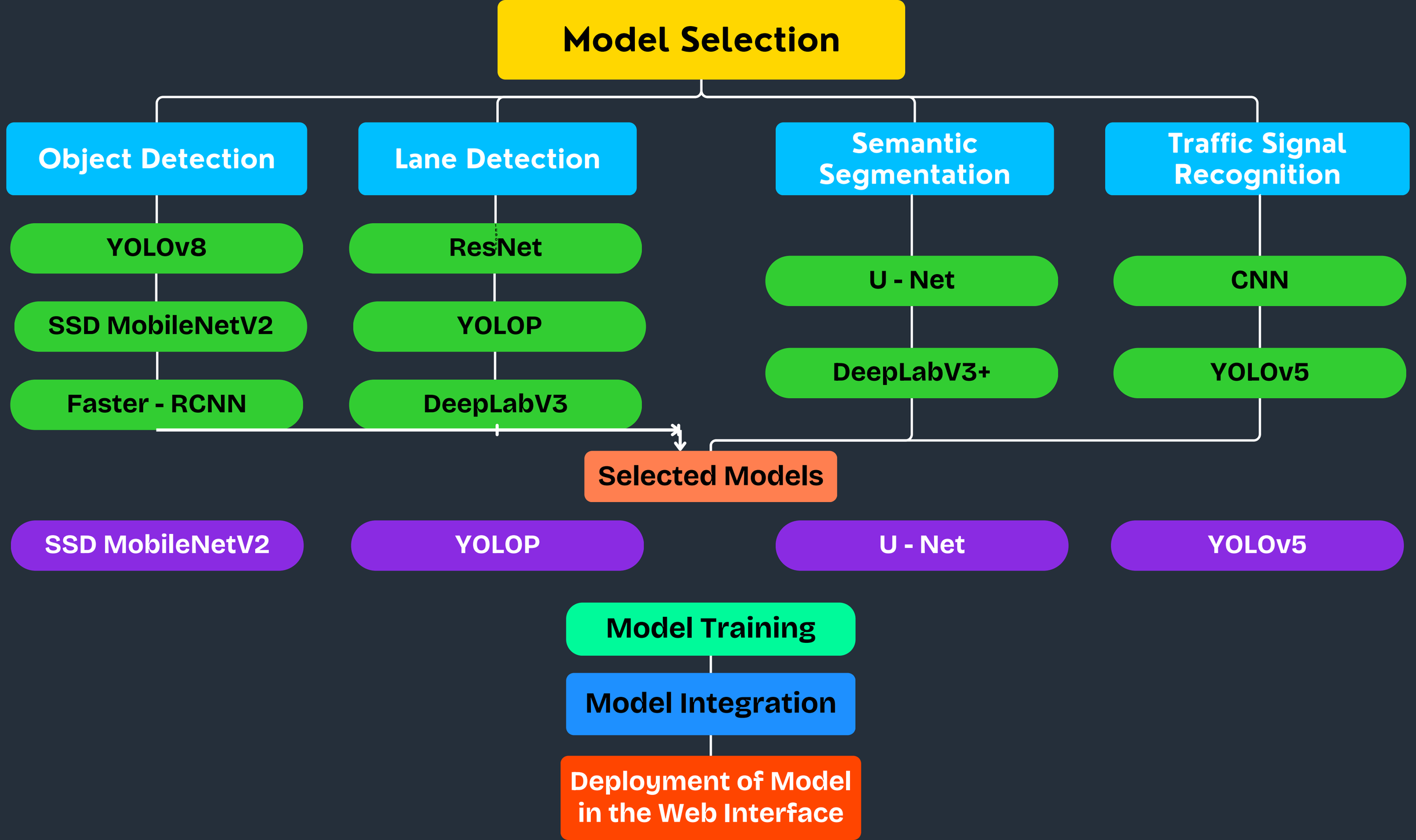


Data Composition Analysis of GTSRB Dataset

The GTSRB dataset comprises over **50,000 images** of traffic signs, divided into training, validation, and test sets in the ratio as shown below




Datasets
Cityscapes Dataset
GTSRB Dataset
Uniform resolution & channel of 2048x1028, 3
Non-uniform resolution b/w 15x15 & 250x250 and RGB channel
Image Analysis
Feature Extraction
Pixel/color Intensity
Bluriness Score
Normalizing
Resizing

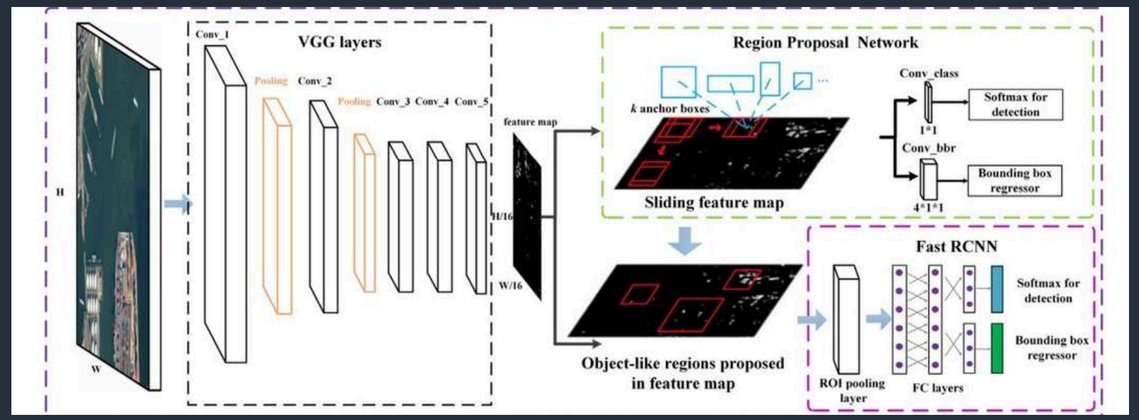

Model Selection
Object Detection
Lane Detection
Semantic Segmentation
Traffic Signal Recognition
YOLOv8
SSD MobileNetV2
Faster - RCNN
ResNet
YOLOP
DeepLabV3
U - Net
DeepLabV3+
CNN
YOLOv5
Selected Models
SSD MobileNetV2
YOLOP
U - Net
YOLOv5
Model Training
Model Integration
Deployment of Model in the Web Interface

Model Exploration and Selection For Object Detection

The Models Explored are:

YOLOv8 (You Only Look Once version 8)

Faster R-CNN (Region-based Convolutional Neural Networks)



SSD (Single Shot MultiBox Detector) MobileNetV2

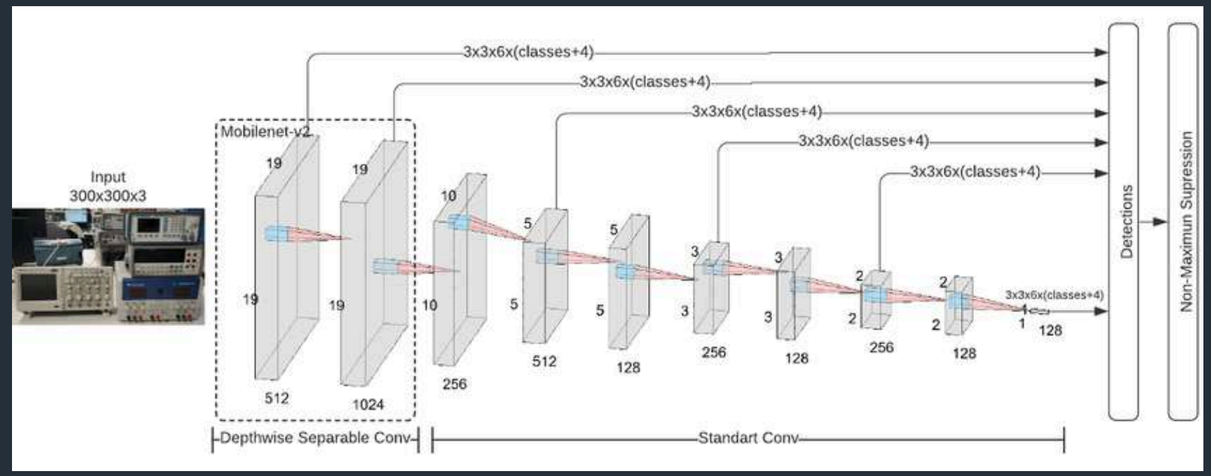


Image SourceS:
LinkedIn Jizong ZHAN
researchgate.net May 2022 John Estrada

Final Model selection For Object Detection

Criteria	YOLOv8	Faster R-CNN	SSD with MobileNetV2
Accuracy	High	Very High	Moderate
Data Required	High (Heavily Labeled)	High (Labeled Regions)	Low (Less Labeled Data)
Inference Time	Fast (Real-time)	Slow (High Inference Time)	Fast (Real-time)
Resource Intensity	High (GPU Intensive)	Moderate (High Memory Usage)	Low (Efficient)
Detection of Nearby Objects	Moderate	High	High
Suitability for Real-time Applications	High	Low	Very High

The final selected model, SSD MobileNet, was chosen for its low inference time, which is crucial for object detection, as high accuracy is more important for nearby objects, and it requires minimal training data or additional layers.

Challenges faced

1) Improper annotations: Inconsistent annotations format for bounding boxes . Had to annotate the images Manually which consumed lot of time and also YOLOv8 didn't perform good on that.

2) Training issue: The input & labels format is quite complex to cast preprocessed images to

High Computational Requirements: Large image resolutions and complex models (e.g., Faster R-CNN) require substantial resources and training time.

Conflict of Dataset Goal : The Cityscape dataset was primarily meant for segmentation tasks and object detection can't be done very easily

Overfitting: Limited dataset size increases the risk of poor generalization on unseen data.

3) Due to conflict of input and labels format: fine tuning or Transfer learning on our own Cityscapes dataset is very difficult and requires a lot of preprocessing or is next to impossible.

Solution

1) Preprocess the Currently present data of semantic segmentation, and convert it in the format of bounding box annotation suitable for training

2) We used Cloud computational resources available on the Colab

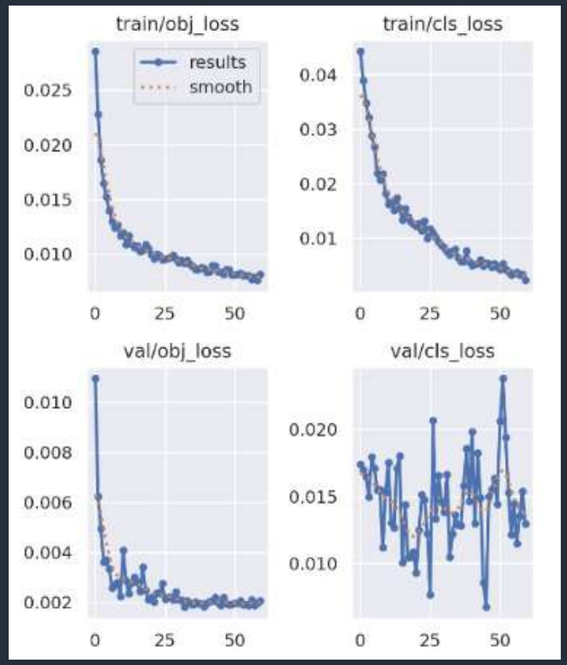
Change the format from segmented to bounding boxes.

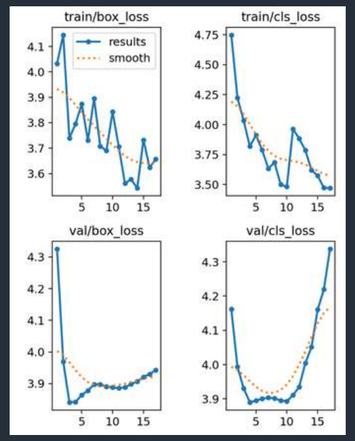
3) Used Data Augmentation to overcome Overfitting

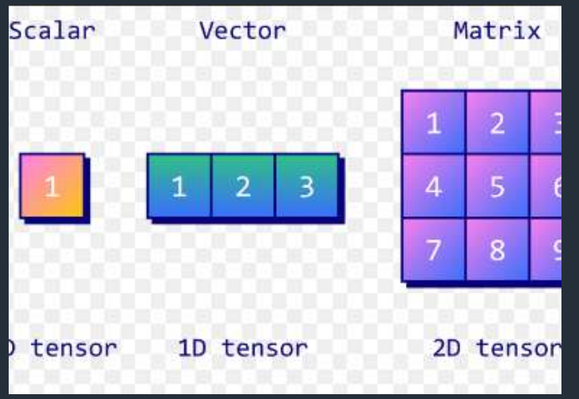
Used a Pre trained Model:

Due to lack of recourses and time constraint developing a custom high-end model was not possible, so using pre-trained models was far of efficient for our project

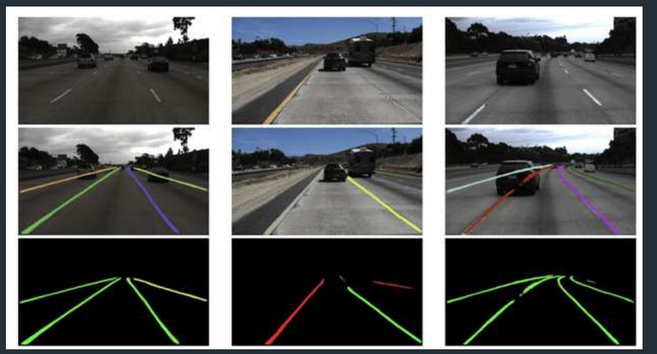
```
1 0.178835 0.460303 0.008667 0.043594
1 0.260615 0.455488 0.007217 0.042246
1 0.080667 0.462710 0.020581 0.086260
1 0.697842 0.442427 0.026113 0.138857
1 0.619224 0.440840 0.018174 0.084863
0 0.225671 0.453691 0.054106 0.097227
0 0.349683 0.458086 0.079199 0.122324
0 0.404377 0.447104 0.036460 0.056455
0 0.428687 0.444360 0.021562 0.041553
```





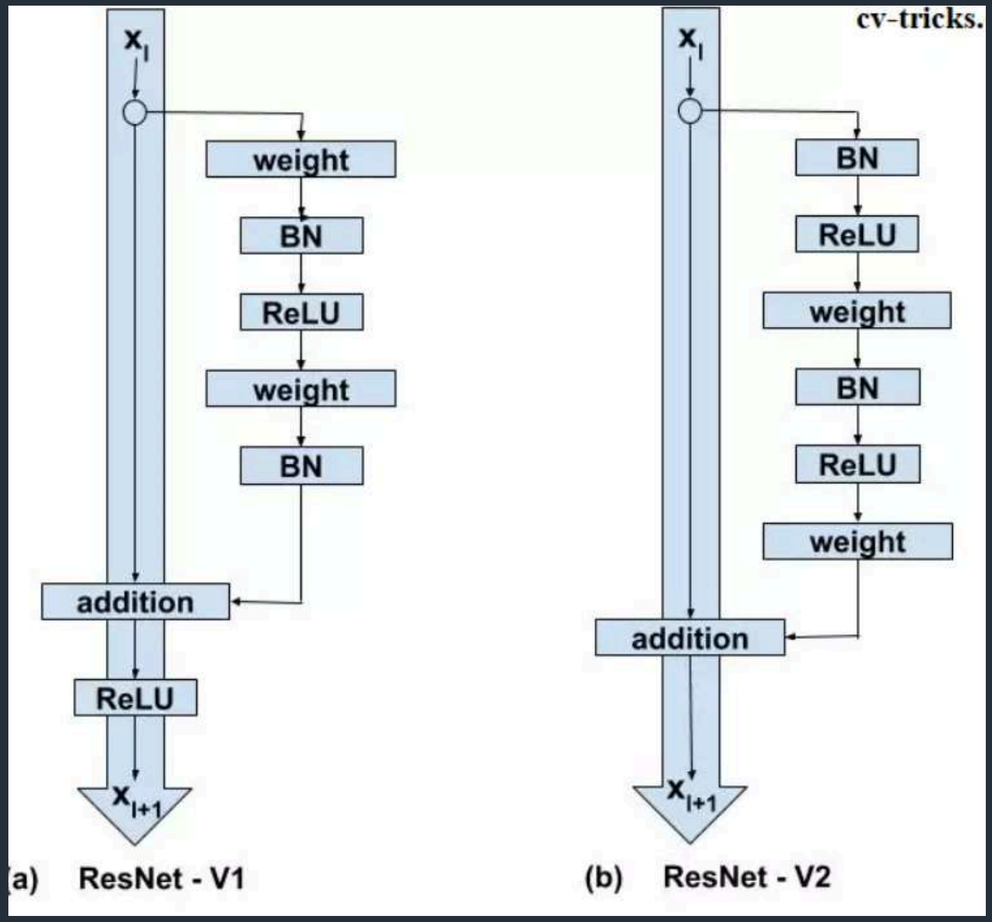


Model Explorations & Selection for Lane Detection



The Models Explored are:

Hough Transform	Classic geometric method for lane detection that maps points in the image to a parameter space for detecting straight lines.	Not based on a traditional loss function; instead, it relies on geometric optimization criteria.
Traditional Computer Vision	Uses edge detection (e.g., Canny) combined with region growing or polynomial fitting techniques (e.g., RANSAC) for lane detection.	Not based on a direct loss function but uses thresholds and heuristics for line fitting and continuity.
Convolutional Neural Networks (CNN)	Basic CNN for lane detection, typically based on semantic segmentation.	**Cross-entropy Loss**: for pixel-wise classification of road vs. non-road regions.
SegNet	Encoder-decoder CNN for pixel-wise segmentation.	**Cross-entropy Loss**: for pixel-wise classification of lane and background.
LaneNet	A deep learning model based on CNNs that performs both lane segmentation and lane instance recognition.	**Binary Cross-entropy Loss**: for segmentation; **Triplet Loss**: for instance segmentation (to distinguish lanes).



The ResNET model

Challenges we faced during Lane Detection

1. Variability in Lane Appearance: Lanes can vary significantly based on road type (highways, city roads, rural roads, etc.), road markings (solid, dashed, colored, faded, or missing), and weather conditions. ResNet, like any deep learning model, needs to generalize well across these variations. If the training data doesn't adequately cover these scenarios, performance will degrade.

2. Surface Quality: Road surface conditions can vary significantly—some lanes may be clear and well-marked, while others may be faded or poorly marked. This inconsistency can complicate lane detection.

3. Curved and Intersecting Lanes: Lanes are often not straight and may curve, split, or merge. While ResNet is capable of capturing high-level features, accurately detecting curved or intersecting lanes requires a model that can understand geometric context over large regions.

Solution 1: Data Augmentation and Diverse Training Datasets To address the variability in lane appearance across different road types, weather conditions, and road markings, it's essential to incorporate a diverse dataset during training.

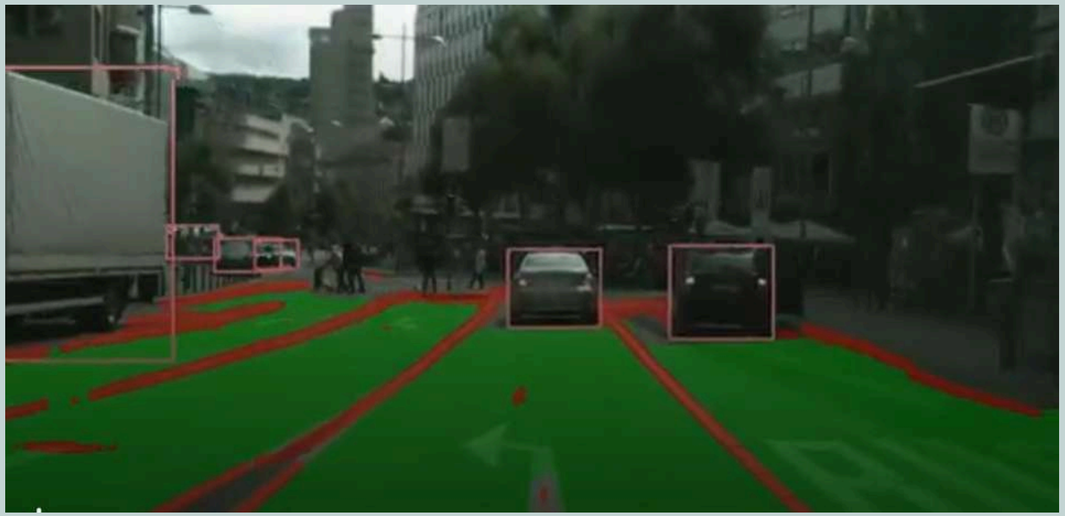
Solution 2: To address this challenge, we perform pre-processing on images to improve their visual clarity. This involves enhancing the contrast to make crucial features, such as lane boundaries, more prominent. By doing so, the lane markers and edges become more distinguishable, aiding in accurate detection and analysis.

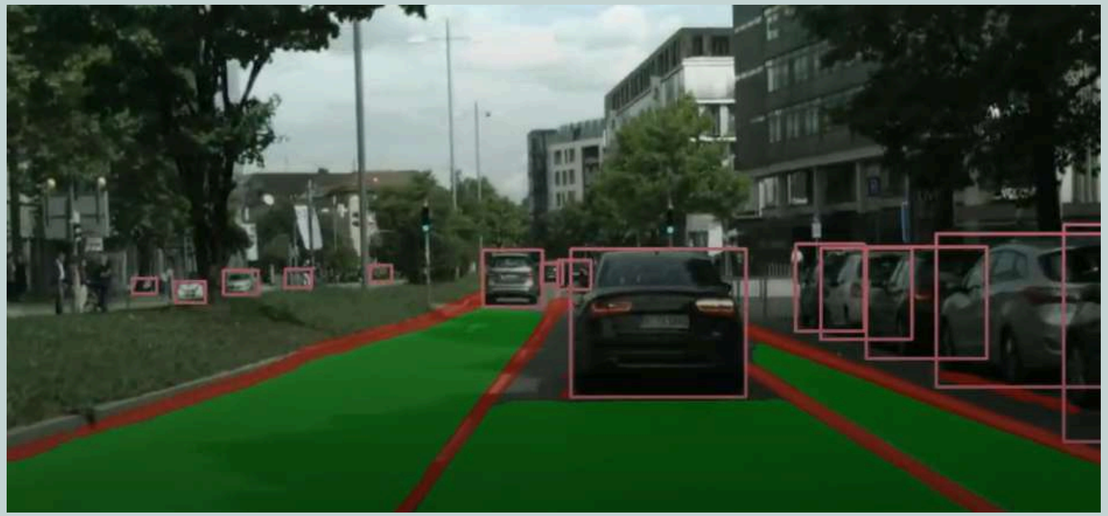
Solution 3: Geometric Understanding and Post-processing Algorithms For curved and intersecting lanes, the model needs to not only recognize lane boundaries but also understand their geometric context. Incorporating a recurrent neural network (RNN) or Long Short-Term Memory (LSTM) layers alongside ResNet can improve the model’s ability to learn temporal dependencies and capture long-range geometric patterns.

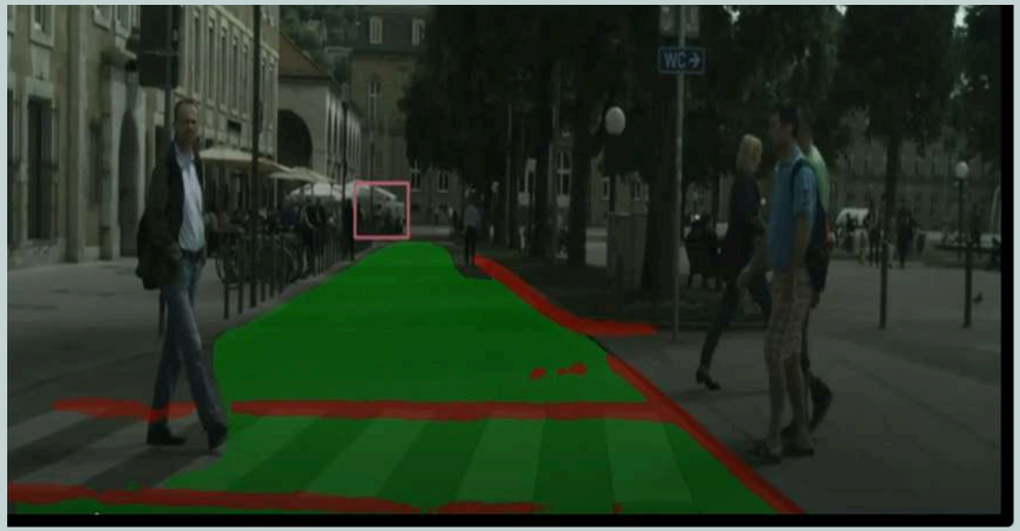
Models explored for solutions

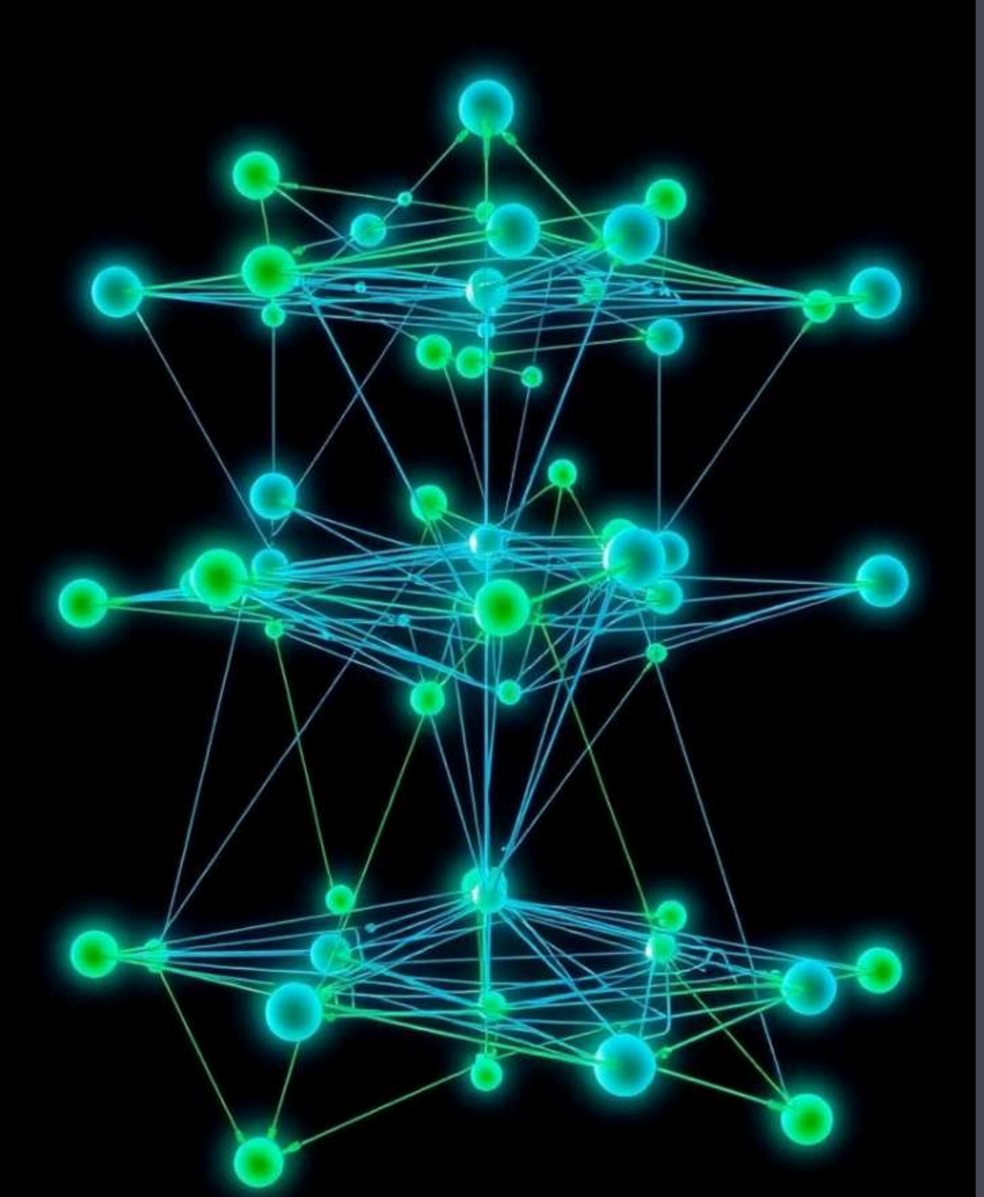
- **Integrating Yolo with LSTM to infer lane postions**
- **DeepLabV3 to incorporate spatial context.**
- **Training deep learning models with augmented datasets that simulate various lighting and weather conditions**

Results using YoloP for Lane Detection









Model Selection for Semantic Segmentation

U-Net:

U-Net's **encoder-decoder architecture will allow us to capture detailed spatial information and context, making it suitable for precise pixel-level segmentation.**

DeepLab (V3+ Variant):

This model will utilize atrous convolutions for multi-scale context and an encoder-decoder structure to enhance edge details, improving performance on complex urban scenes.

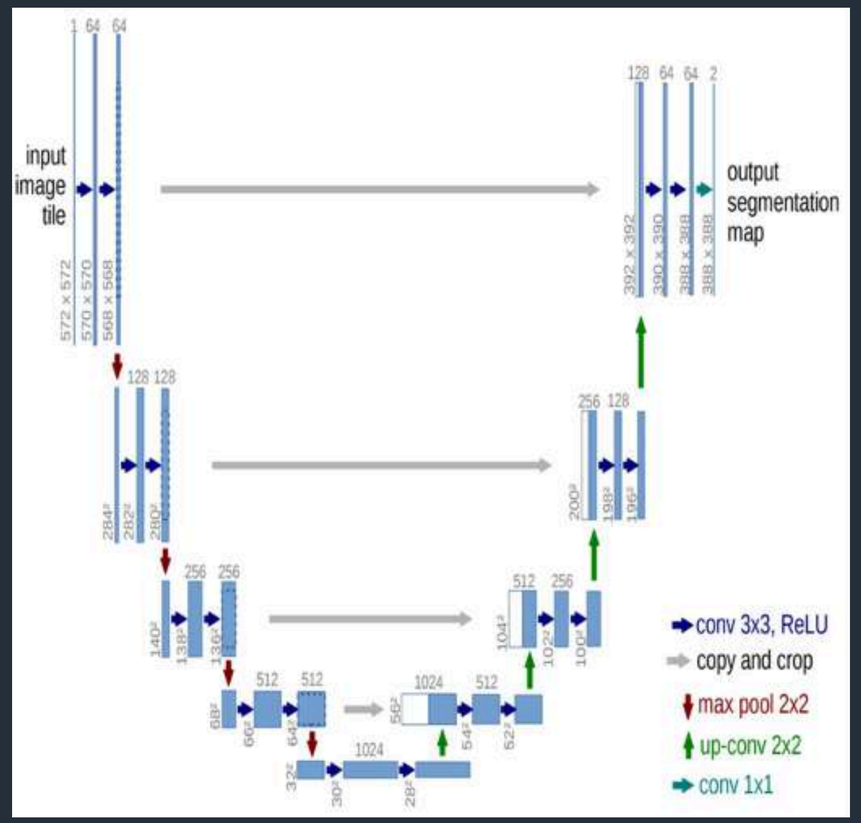
Fourier Transformation-Based Model :

Leveraging spatial frequencies, this approach will help detect textures and patterns, complementing segmentation by capturing both global context and fine details.

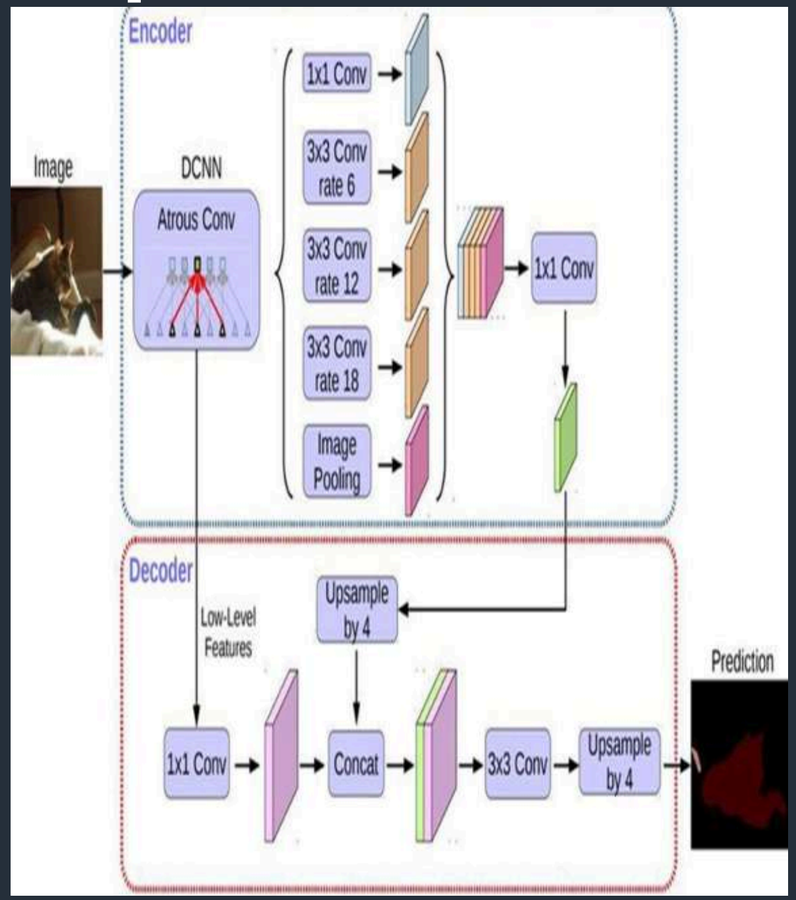
Model Explorations & Selection for Sementic Segmentation

The Models Explored are:

The U-NET model



The Deep Lab V3+ model



Challenges faced for Semantic Segmentation

Challenge 1: Difficulty in Model Selection

Selecting the right model for semantic segmentation can be challenging due to the vast number of available options.

Each model has its strengths and weaknesses, making it difficult to determine which will perform best for a specific task.

Solution

1. Facilitated Communication Daily meetings were implemented to enhance open communication among team members.

2. Adoptionand Comparison of Models These meetings allowed team members to effectively adopt and compare different models.

Challenge 2: RGB Channel Misinterpretation

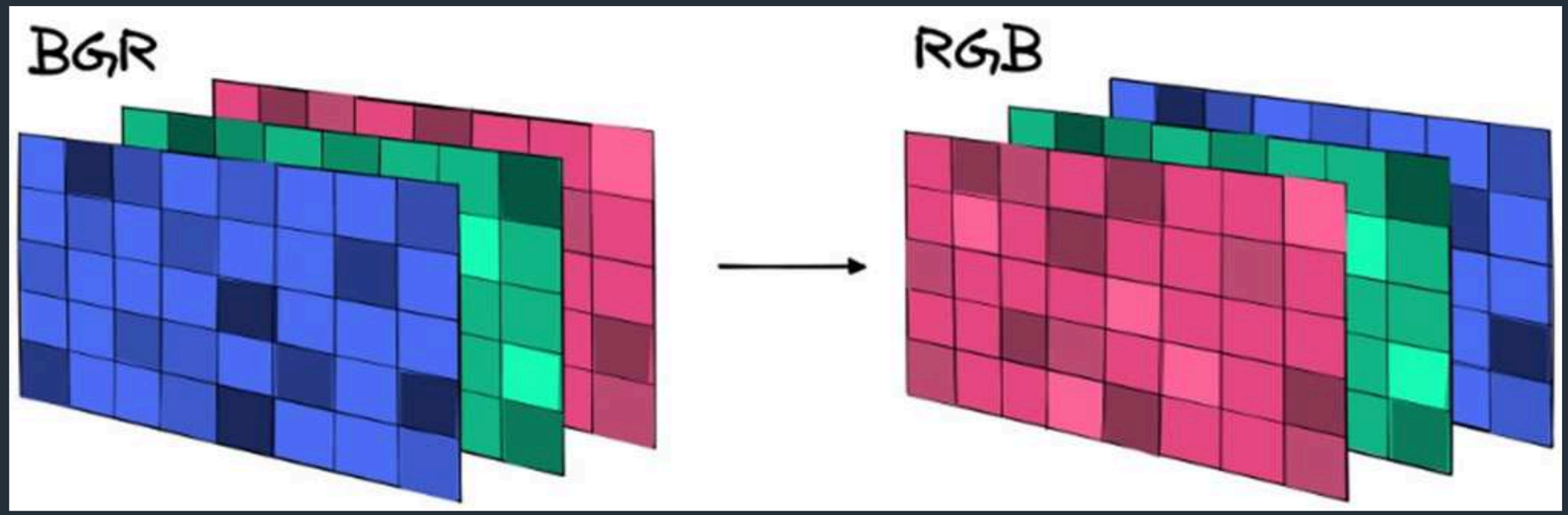
Semantic segmentation models rely on accurate color interpretation.

If the RGB channels are misinterpreted as BGR, the model's performance will suffer.

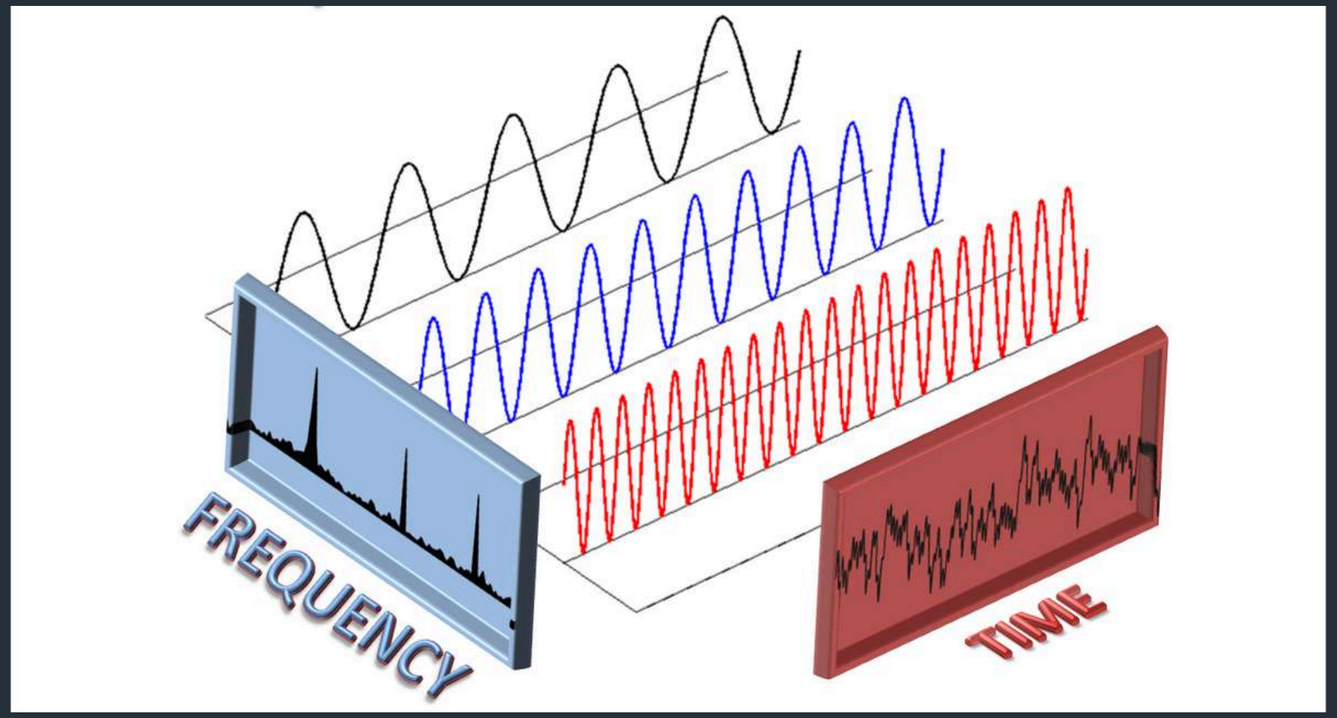
SOLUTION:

We addressed this challenge by validating input data formats before feeding them into the models.

This ensured that the correct channel order was used, leading to improved accuracy.



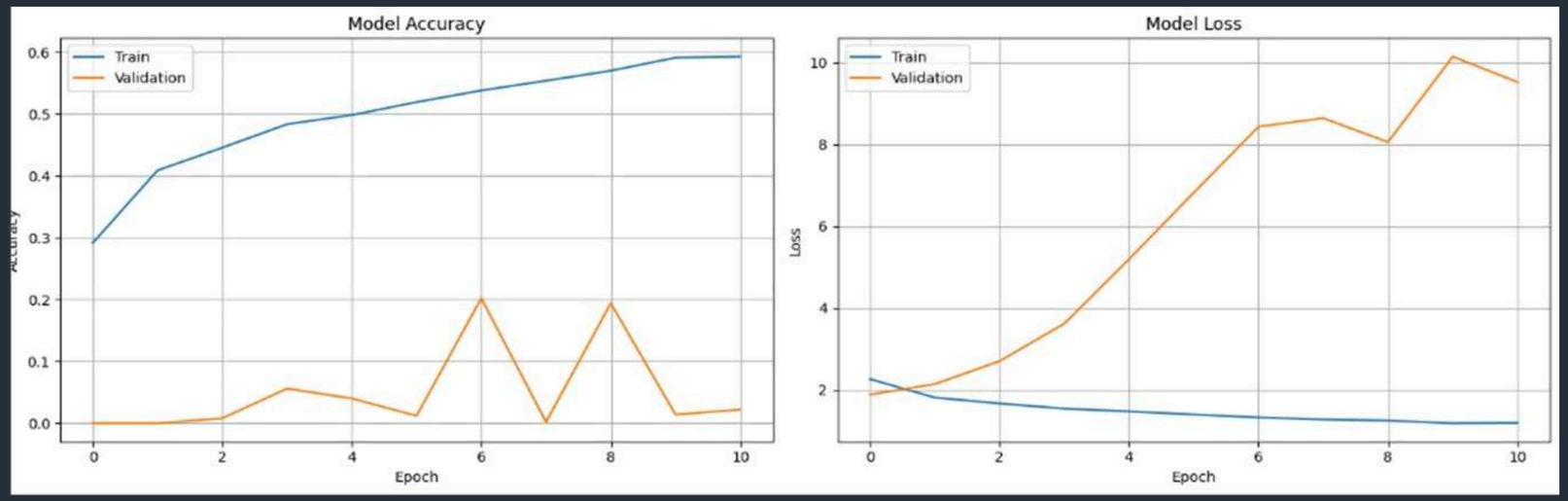
Challenge 3: Inapplicability of Fast Fourier Transformation (FFT)



1. The Fast Fourier Transform (FFT) is a technique commonly used for image processing. However, it is not suitable for all models, particularly those like DeepLabV3+ and U-Net.
2. While FFT can be effective for high-noise images, like those in medical segmentation, it can actually degrade the performance of models dealing with clearer images.

Solution:

To address this, we replaced FFT with standard preprocessing techniques like resizing and normalization.

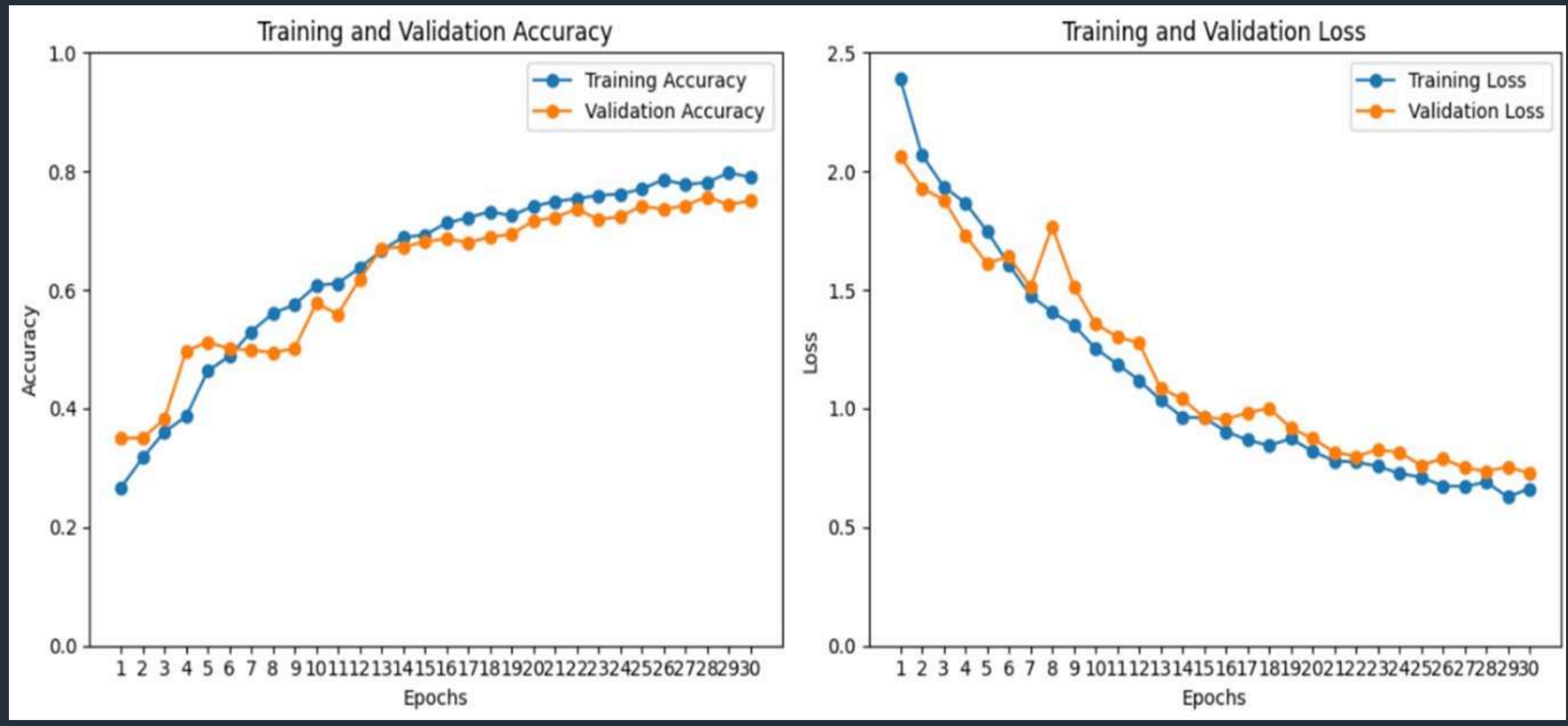


Challenge 4: Misinterpretation of Metrics

Semantic segmentation models rely heavily on accurate evaluation metrics to assess their performance.

SOLUTION:

We addressed this by re-validating our metrics, ensuring they accurately reflect the model's performance.



Challenge 5: U-net Model: Less Accuracy

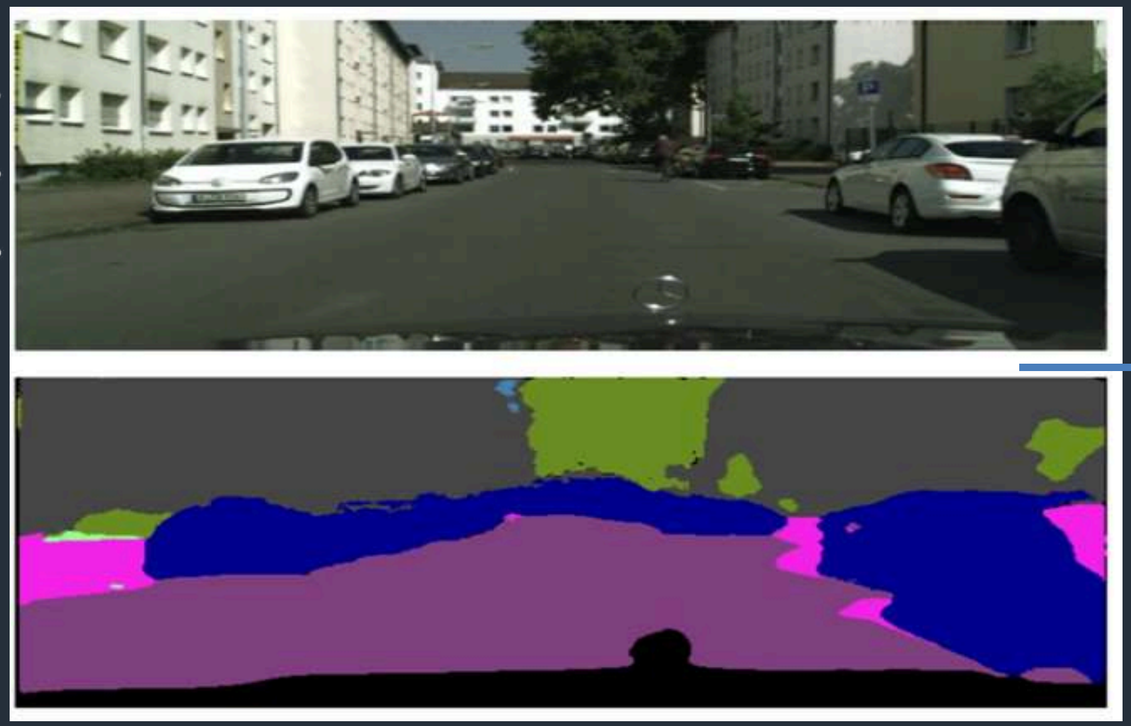
Initially, we got poor results for semantic segmentation, as shown in the picture.

We overcame this challenge by changing the model architecture.

Key Changes Made:

1. Modified the model architecture to enhance feature extraction.
2. Adjusted hyperparameters for better convergence.
3. Implemented data augmentation techniques.

Accuracy: 80%.



Challenge 6: Discrepancy Between Validation and Testing Results with DeepLabV3+ model

```
# Print the evaluation metrics
print(f"Confusion Matrix:\n{conf_matrix}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"Accuracy: {accuracy}")
```

```
Confusion Matrix:
[[131072000         0         0]
 [         0         0         0]
 [         0         0         0]]
Precision: 1.0
Recall: 1.0
Accuracy: 1.0
```

When we trained and validated the trained deeplabV3+ model, we got perfect accuracy and precision. But during testing we got poor segmentation results.

Strategies to overcome this challenge:

1. Regularization Techniques: Apply techniques like dropout or L2 regularization to prevent overfitting during training.
2. Hyperparameter tuning: Experiment with different hyperparameters to optimize the model's performance on unseen data.

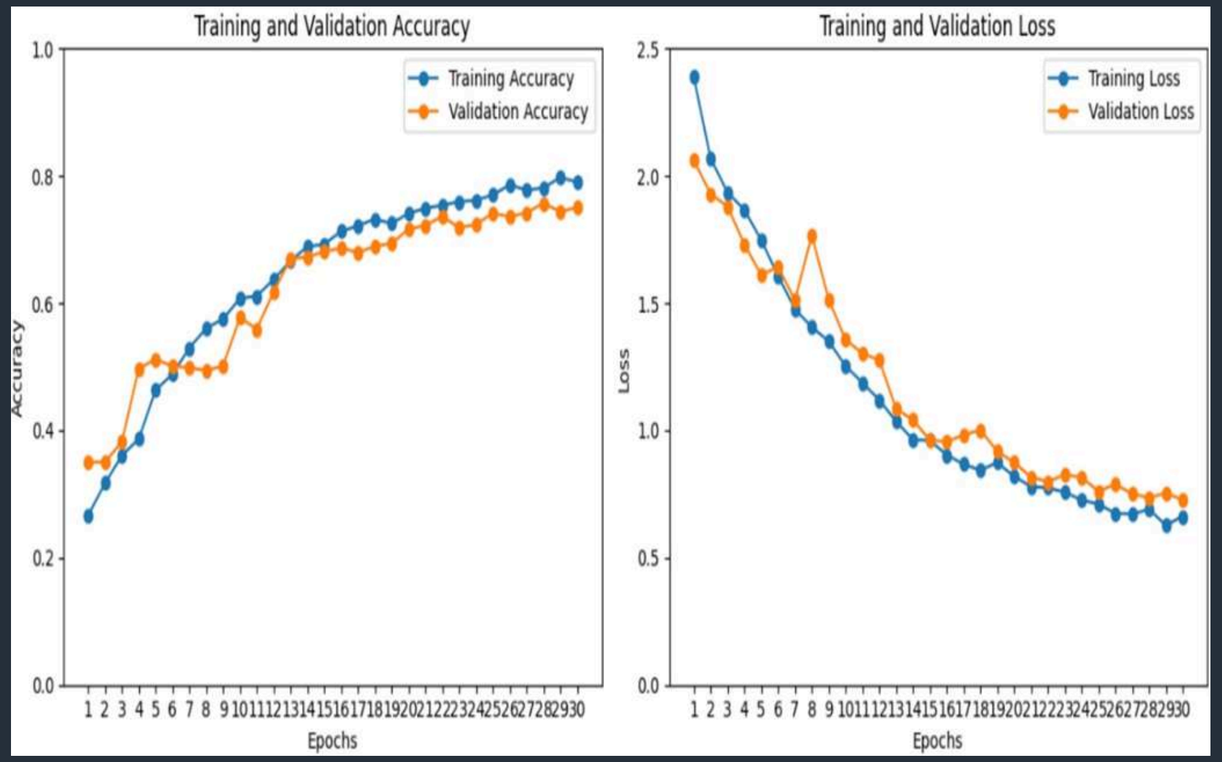
Final Results for Semantic Segmentation

```
100%|
Epoch [5/5], Loss: 0.0013
Training completed.

# After the training loop
torch.save(model.state_dict(), "deeplabv3plus_cityscapes.pth")
print("Model saved successfully.")

Model saved successfully.
```

1. DeepLabV3+ Model: Successfully trained on the whole dataset but still showed poor results on the testing dataset, so not chosen for semantic segmentation.
2. U-Net Model: Successfully trained, validated, and tested for semantic segmentation, achieving 80% accuracy. Hence chosen for sematic segmentation.



MODEL SELECTION FOR TRAFFIC LIGHT DETECTION AND SIGN DETECTION

The Models Explored are:

- **CNN (CONVOLUTION NEURAL NETWORK)**

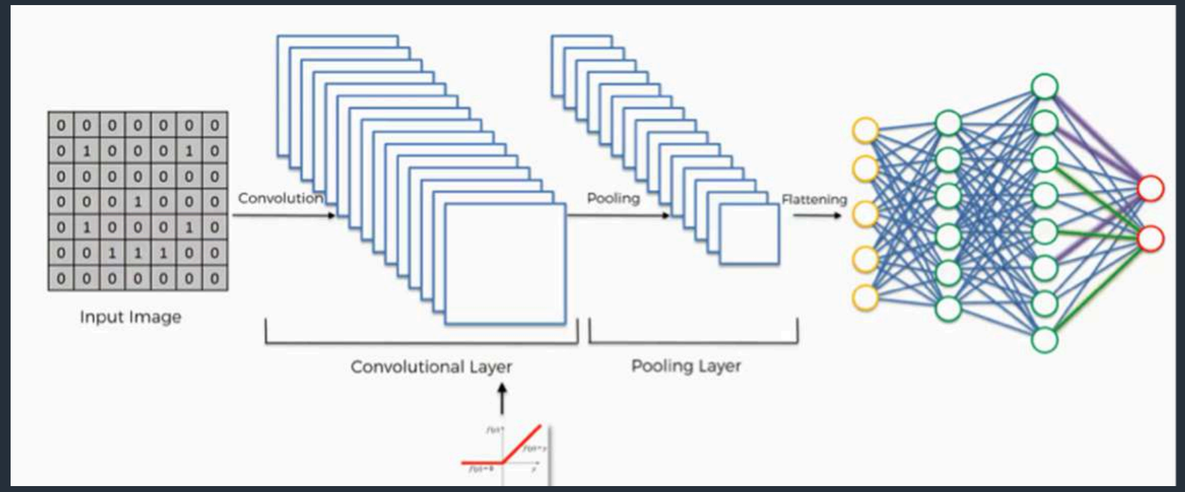
The CNN model was selected for this traffic light detection due to its strong capability in processing visual data through convolutional layers that effectively detect features such as edges, shapes, and patterns

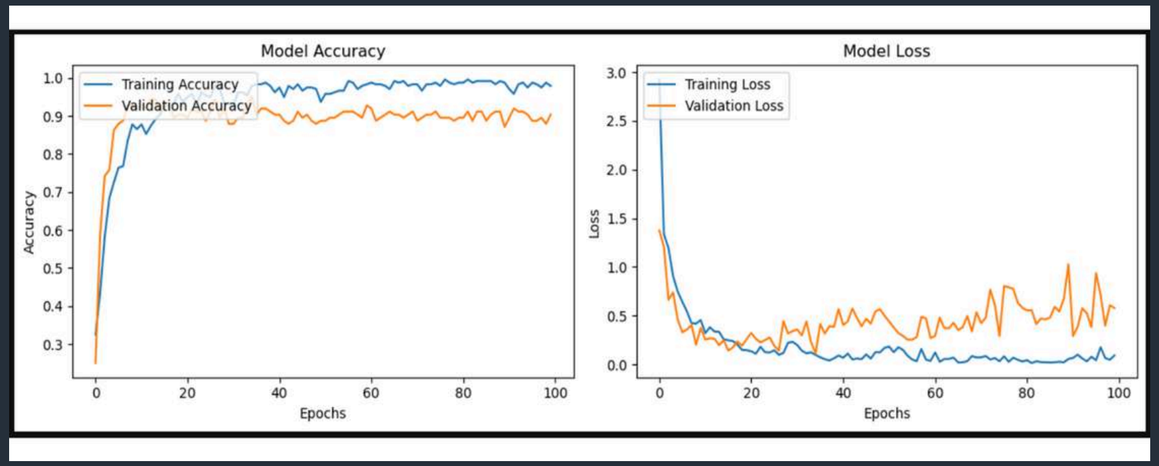
- **YOLOV5 (YOU ONLY LOOK ONCE)**

The YOLO (You Only Look Once) model was selected for sign detection due to its real-time object detection capabilities and ability to process visual data in a single pass through the network.

CNN MODEL FOR TRAFFIC LIGHT DETECTION

- Conv Layers: Extract spatial features, 32 filters.
- Pooling: Reduces dimensions, retains key features.
- Flattening: Converts 2D maps to 1D vector.
- Dense Layers: Learns patterns, classifies outputs.
- Compilation: Adam optimizer, categorical loss function.





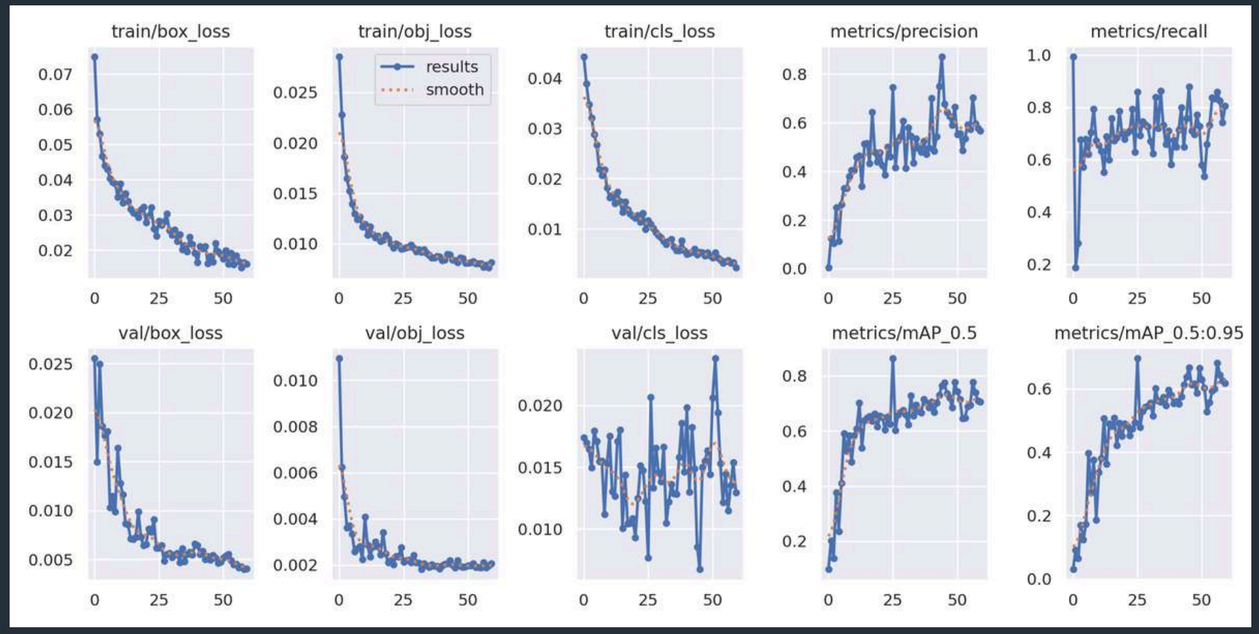
- Training: The model was trained for 100 epochs using a validation set.
- Results: The achieved accuracy was approximately 90%.

YOLO MODEL FOR SIGN DETECTION

- **The YOLO (You Only Look Once) model was selected for sign detection due to its real-time object detection capabilities and ability to process visual data in a single pass through the network.**

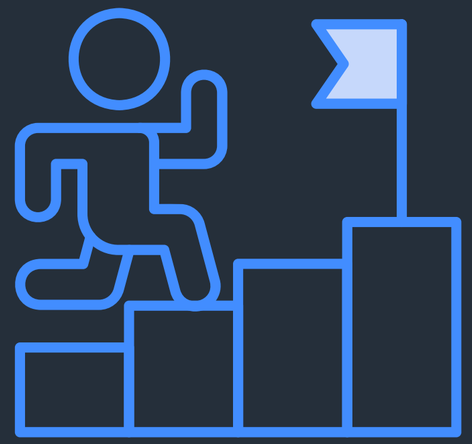
- **Trained the model to recognize six different traffic-related classes:**

  - **30 Speed**
  - **60 Speed**
  - **Pedestrian Crossing'**
  - **Turn Left Ahead**
  - **Turn Right Ahead**
  - **Keep Right**



Challenges faced for traffic signal detection

- We faced challenges in cropping traffic lights from the images in the Kaggle dataset
- Managing a large dataset that complicates the training process. Recognizing objects within images.
- Misleading detections caused by reflections from car lights and other sources.
- The low quality of image pixels affects detection accuracy.

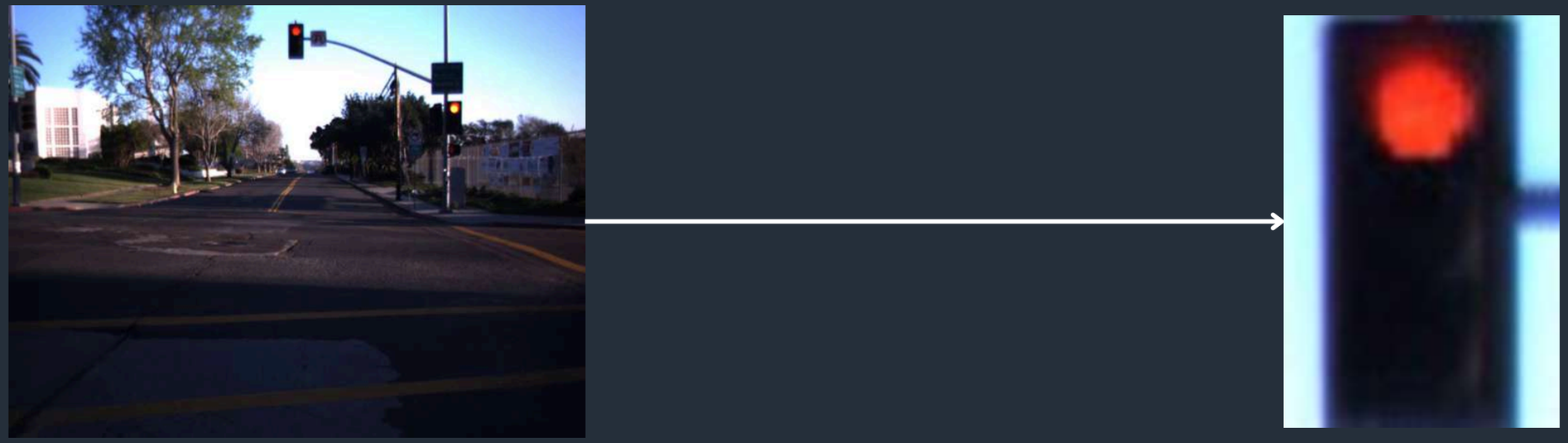


Challenges faced for Traffic Sign Detection

Code for extracting traffic light images based on their annotations:

- The code utilizes built-in functions from the Libraries csv, os, and cv2 to extract traffic light images based on their annotations.

- It takes the given coordinates from the annotations, creates a new folder, and saves all images separately.



Managing a large dataset that complicates the training process:

•**Data Preparation:** Collected 50 images from each clip and organized them into a designated folder for processing.

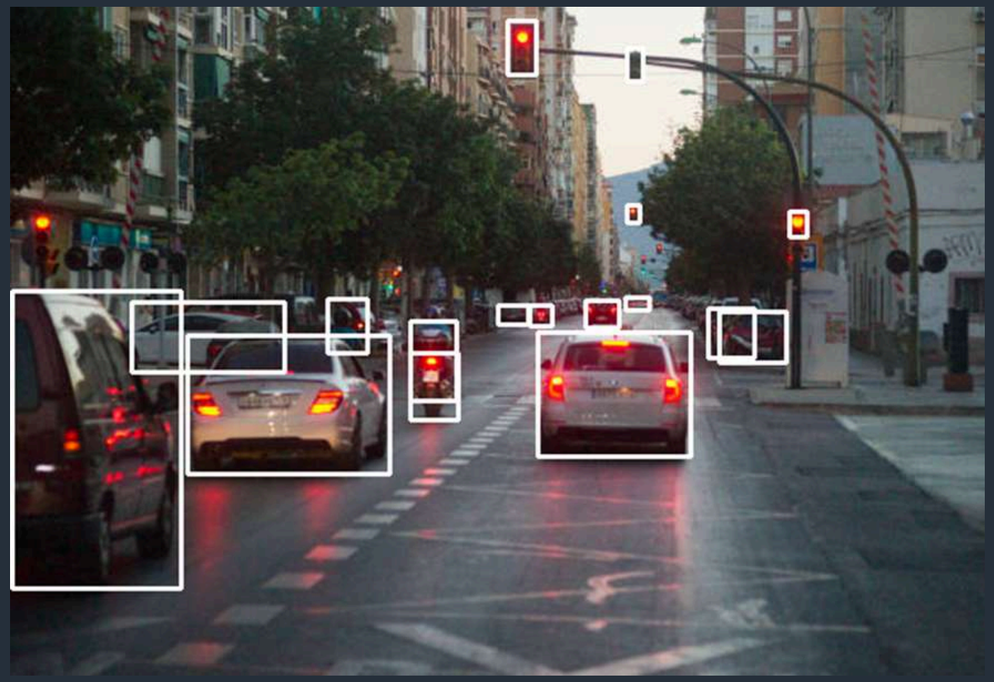
•**Model Configuration:** Loaded a pretrained Faster R-CNN model and set the model to evaluationmode**.**

•**Inference Process:** Used GPU resources for classification,processing each saved image to predict its traffic light state.

•**RESULTS :** Achieved accurate classifications of traffic light images, demonstrating the effectiveness of transfer learning in object detection tasks.

Recognizing objects within images:

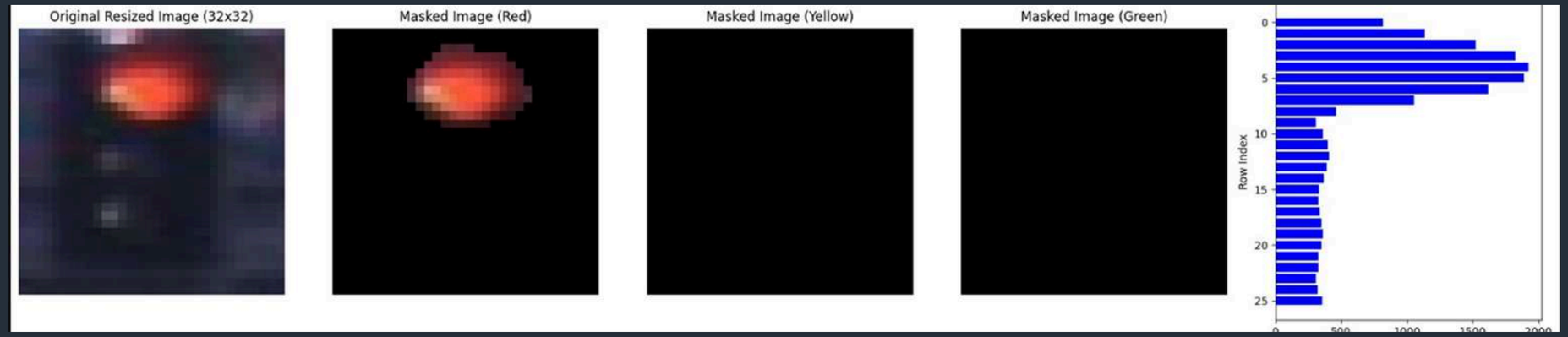
YOLOv5 detects multiple objects, requiring precise filtering to identify and isolate traffic lights from other objects in complex scenes



Misleading detections caused by reflections from car lights and other sources.

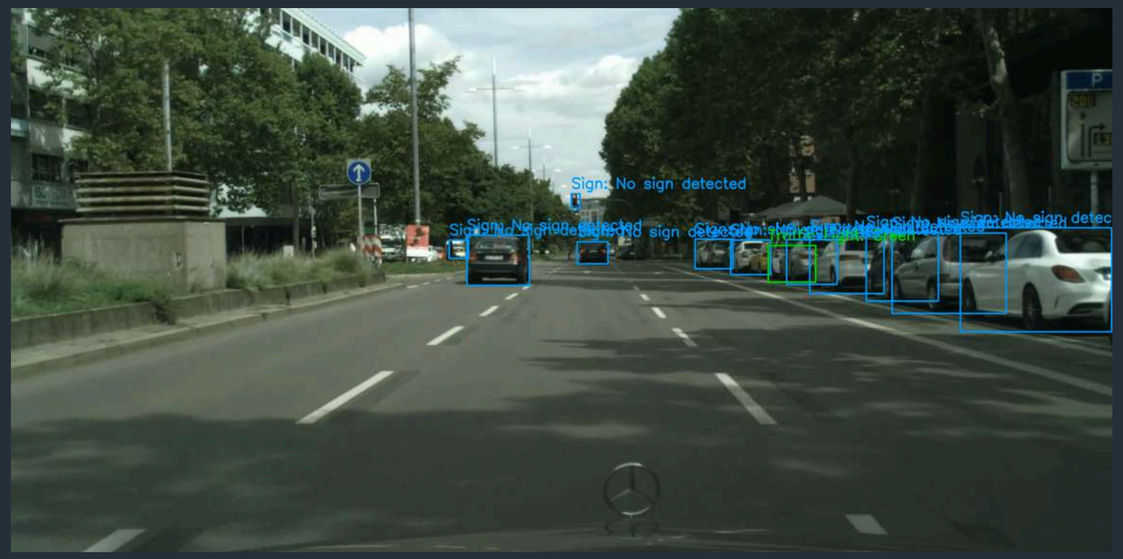
- The program calculates a brightness vector from cropped HSV values, isolating actual traffic lights by emphasizing consistent brightness patterns, helping to reduce false detections from reflections.

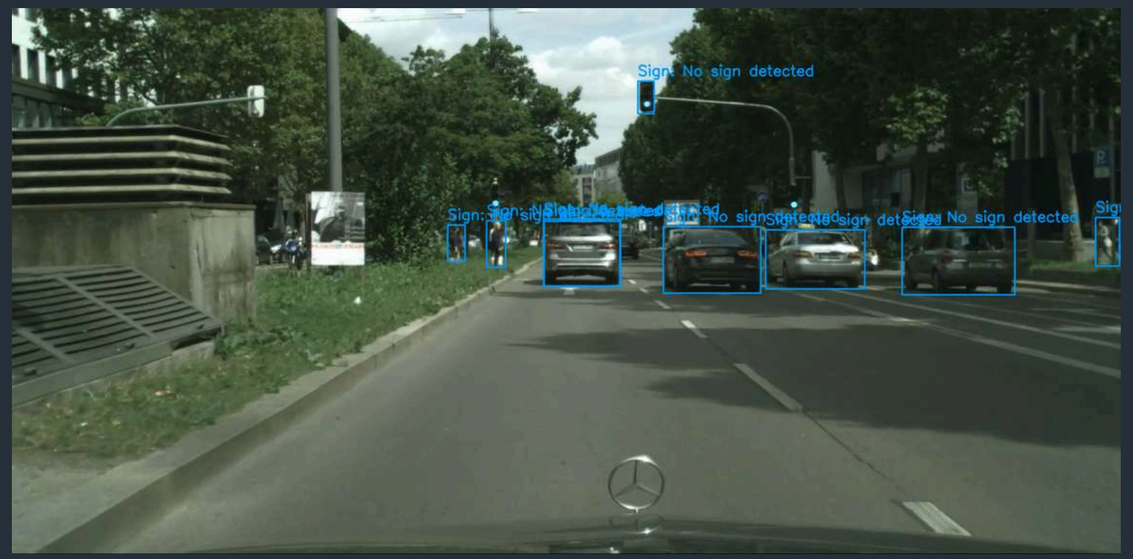
- The program calculates a brightness vector from cropped HSV values, isolating actual traffic lights by emphasizing consistent brightness patterns, helping to reduce false detections from reflections.



Challenges faced in integration of object detection and traffic signal detection :

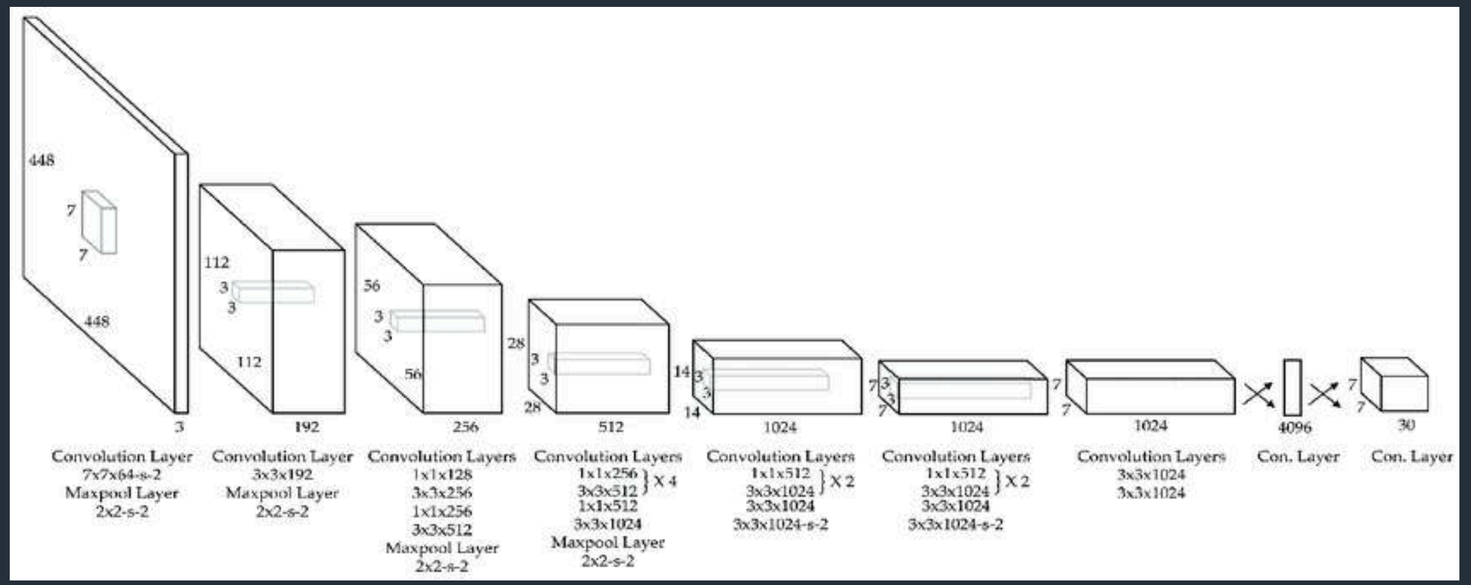
- **The accuracy of the traffic signal detection model is suboptimal.**





Enhancing Traffic Light Detection with YOLOv5 Pretrained Model

- Selected YOLOv5 pretrained model (e.g., 'yolov5s') trained on the COCO dataset, which includes the 'traffic light' class.
- Improved accuracy by leveraging a robust, pre-trained model instead of custom training on limited resources.
- Fast, real-time detection capabilities, making it suitable for autonomous driving applications.



DECISION MAKING

- **Overview**: Implemented a basic algorithm for decision-making in autonomous driving scenarios.
- **Functionality**: Analyzes detected objects, traffic signs, and lanes to make real-time decisions like turning, braking, or accelerating.
- **Focus**: Prioritizes simplicity and efficiency for reliable decision-making in diverse road environments.
- **Outcome**: Ensures safe and adaptive responses to dynamic traffic conditions.

RESULTS

Conclusion:

- Comprehensive Autonomous Driving Solution: Developed a modular autonomous driving system with real-time capabilities taking input feed from dash cams and GPS receiver.
- Real-Time Functionality
- Driver Support
- Versatility and Scalability
- Foundation for Future Innovations
- Integrated various Ai enabled modules such as:

1. Object detection for cars, automobiles, pedestrians, etc for road safety with SSD MobileNetV2 model (88.6 %)
2. Lane detection to achieve consciousness of the driver and public safety with yolop model (79.4%)
3. Semantic segmentation: Achieved a detailed scene understanding for the decision-making algorithms via semantic segmentation with U-net model (80%).
4. Traffic sign / signal recognition and its compliance with YOLOV5 model (90%)

FUTURE GOALS:

- **Reinforcement Learning for Adaptive Decision-Making:**
  Implement reinforcement learning algorithms to enable adaptive and dynamic decision-making.
- **Hybrid Architectures:**
  Develop and integrate hybrid architectures to enhance system performance in challenging real-world scenarios.
- **Driver Drowsiness Detection:**
  Incorporate real-time drowsiness detection to monitor driver alertness and ensure safety.
- **Sophisticated Decision-Making Algorithms:**
  Introduce advanced algorithms to handle complex and unpredictable situations effectively.
- **Enhanced Lane Detection:**
  Improve lane detection accuracy under varying weather, lighting, and road conditions.
- **Advanced Object Recognition:**
  Enhance object recognition systems to function reliably in diverse and complex environments.
- **System Robustness:**
  Continuously improve the system’s robustness to adapt to real-world complexities and uncertainties.


thank
you