A PROJECT REPORT
ON

AI- BASED AUTOMONOUMS DRIVING MODEL

Submitted By:

PARVINDER KUMAR

Submitted To:

MR. NITIG SINGH

INFOSYS SPRINGBOARD INTERNSHIP 5.0 BATCH- 1

ABSTRACT

Autonomous driving requires precise perception of the surrounding environment to ensure safe and reliable vehicle operation. This paper presents a comprehensive deep learning-based model that integrates key perception tasks such as lane tracking, object detection, and semantic segmentation, essential for urban driving scenarios.

The model utilizes **lane tracking** to accurately detect and follow road lanes under various conditions, including occlusions and complex lane markings. **Object detection** identifies and classifies dynamic objects, such as vehicles, pedestrians, and cyclists, ensuring collision avoidance and path planning. **Semantic segmentation** provides pixel-level understanding of the scene, identifying different classes such as road surfaces, sidewalks, traffic signs, and obstacles, which enhances overall scene comprehension for decision-making processes.

We employ the **Cityscapes** dataset, which offers a large-scale, diverse set of annotated urban street scenes, to train and evaluate the model. The dataset includes high-resolution images with fine-grained annotations, allowing the model to generalize well across different urban environments. Our approach outperforms several state-of-the-art methods in terms of accuracy and robustness, particularly in challenging real-world driving conditions.

The proposed system is designed to operate in real-time, making it suitable for deployment in autonomous vehicles, where fast and accurate scene understanding is critical. Through extensive experimentation, we demonstrate that integrating these tasks into a unified framework significantly improves the vehicle's perception capabilities, ultimately contributing to safer and more efficient autonomous driving.

***Keywords*:** Semantic Segmentation, Object Detection, Lane Detection, Traffic Sign Recognition

Introduction

Autonomous driving systems rely on the vehicle's ability to accurately perceive and understand the surrounding environment. This requires the integration of several critical components: **semantic segmentation**, **object detection**, **lane detection**, and **traffic sign recognition**. Each of these tasks plays a vital role in enabling the vehicle to navigate safely through complex urban environments.

- **Semantic segmentation** classifies each pixel in the image into specific categories, such as roads, cars, pedestrians, and obstacles, allowing the vehicle to understand the full scene in a fine-grained manner.
- **Object detection** identifies and classifies important objects like vehicles, pedestrians, and obstacles, which is crucial for collision avoidance and path planning. Advanced techniques, such as color grading, can enhance the accuracy of detection.
- **Lane detection** ensures that the vehicle remains within the correct lanes by identifying lane markings, even in challenging conditions.
- **Traffic sign recognition** detects and classifies traffic signs, enabling the vehicle to follow road rules and respond to various signals.

To develop a robust autonomous driving model, selecting appropriate deep learning frameworks and pre-trained models is essential. **PyTorch** and **TensorFlow/Keras** are popular choices for building and training models, while **YOLO** is widely used for real-time object detection. For **semantic segmentation**, models like **DeepLab** or **U-Net** are effective, while **OpenCV** is often employed for image processing tasks such as lane detection. Supporting algorithms for tasks like traffic sign classification can be implemented using **Scikit-learn**.

High-quality datasets are critical for training these models. For **semantic segmentation**, datasets such as **Cityscapes** and **CamVid** provide urban street scenes with pixel-level annotations. **COCO** and **PASCAL VOC** are widely used for object detection, while the **GTSRB** dataset is tailored for traffic sign recognition. For **lane detection**, the **CULane** and **TuSimple** datasets offer annotated lane images in various driving conditions.

By combining these frameworks, models, and datasets, a comprehensive autonomous driving model can be developed, capable of performing key perception tasks in real-time to ensure safe navigation through complex environments.

1.1 Problem Definition:

The development of autonomous driving systems hinges on the vehicle's ability to accurately perceive and interpret its surroundings in real-time. This involves solving several interrelated perception tasks such as semantic segmentation, object detection, lane detection, and traffic sign recognition. Each of these tasks presents unique challenges in the context of complex urban environments, where diverse road conditions, dynamic objects, and varying lighting scenarios must be accounted for.

1. **Semantic Segmentation**: The vehicle must classify every pixel in its camera feed into categories like road, sidewalk, vehicles, pedestrians, and other relevant objects. The challenge lies in achieving fine-grained scene understanding, even in the presence of occlusions, lighting variations, and diverse urban settings.
2. **Object Detection**: The system needs to accurately detect and classify objects like cars, pedestrians, and obstacles in real-time to avoid collisions and make safe driving decisions. Object detection in dynamic, cluttered environments with varying sizes and shapes of objects is critical for successful navigation.
3. **Lane Detection**: Accurately identifying lane markings ensures the vehicle stays within designated boundaries. This becomes challenging when lane markings are faded, obscured, or change due to varying road conditions, turns, and intersections.
4. **Traffic Sign Recognition**: Detecting and recognizing traffic signs is essential for the vehicle to obey traffic laws and respond to road signals. The complexity increases with different types of signs, varying visibility, and environmental factors like bad weather or partial occlusions.

The primary problem is to develop a unified deep learning-based model that integrates these core perception tasks, ensuring reliable and efficient real-time performance in diverse and complex driving scenarios. The model should be capable of handling the variability and complexity of real-world driving conditions, while leveraging high-quality datasets and pre-trained models to improve generalization and accuracy. The ultimate goal is to enable autonomous vehicles to navigate safely, understanding and responding to their environment in real-time.

1.2 Solution:

To address the challenges in autonomous driving, we propose a unified deep learning-based model that integrates **semantic segmentation**, **object detection**, **lane detection**, and **traffic sign recognition**. By combining these key perception tasks, the system can achieve a comprehensive understanding of urban driving environments and enable safe, real-time decision-making for autonomous vehicles.

1. **Semantic Segmentation**:
   We implement a pixel-wise classification model using state-of-the-art architectures like **DeepLab** or **U-Net**. These models are designed to classify each pixel in the image into categories such as road, vehicle, pedestrian, and building. **Cityscapes** and **CamVid** datasets will be used to train and validate the model, ensuring high accuracy in various urban scenarios. Semantic segmentation enables the system to understand the entire scene at a fine-grained level, making it easier to identify road boundaries, obstacles, and drivable areas.
2. **Object Detection**:
   For detecting and classifying dynamic objects, we use the **YOLO** (You Only Look Once) model due to its real-time object detection capabilities. YOLO is efficient at identifying multiple objects, such as vehicles, pedestrians, and cyclists, in a single pass, making it highly suitable for real-time performance in autonomous driving. We leverage datasets like **COCO** and **PASCAL VOC** for training the model to detect various objects commonly found in urban driving environments.
3. **Lane Detection**:
   Lane detection is handled using **OpenCV** combined with deep learning techniques. The model detects lane boundaries even in challenging conditions, such as faded lines, curves, or occluded markings. Pre-trained models can be fine-tuned using **CULane** or **TuSimple** datasets, which contain real-world lane annotations for training on complex road geometries.
4. **Traffic Sign Recognition**:
   For traffic sign recognition, a classification model based on convolutional neural networks (CNNs) is used. **Scikit-learn** provides useful tools for implementing algorithms that classify detected traffic signs from images. The **GTSRB** (German Traffic Sign Recognition Benchmark) dataset is used to train the system to recognize and interpret various road signs. Accurate recognition of traffic signs ensures the vehicle can respond correctly to signals and road rules, such as speed limits and stop signs.

Integration of Components

To integrate these components into a cohesive autonomous driving model:

- **PyTorch** or **TensorFlow/Keras** frameworks will be used to build and train the models, allowing for flexible development and optimization.
- A real-time inference pipeline will be created to combine the outputs of the segmentation, object detection, lane detection, and traffic sign recognition models.

This enables the system to make decisions based on a comprehensive understanding of the driving environment.
- The model will be optimized for real-time performance using techniques like model pruning, quantization, and hardware acceleration (e.g., GPUs or TPUs).

Datasets

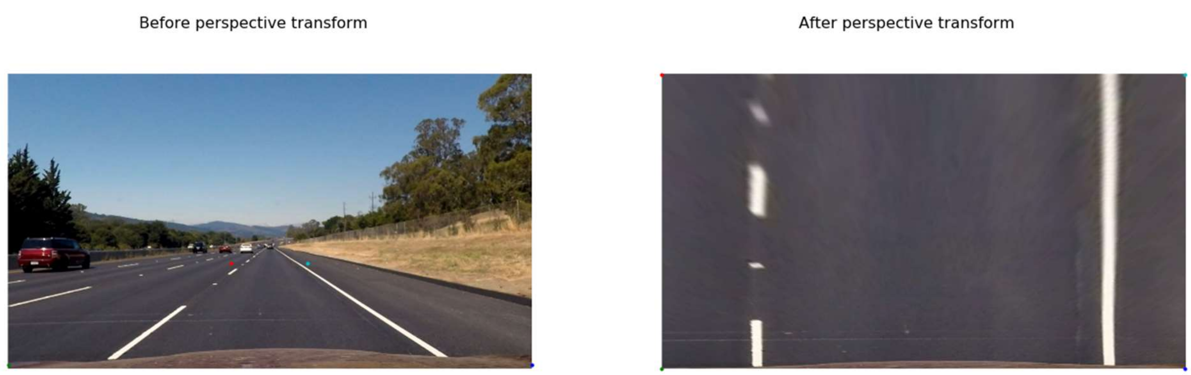
We will use high-quality datasets such as:

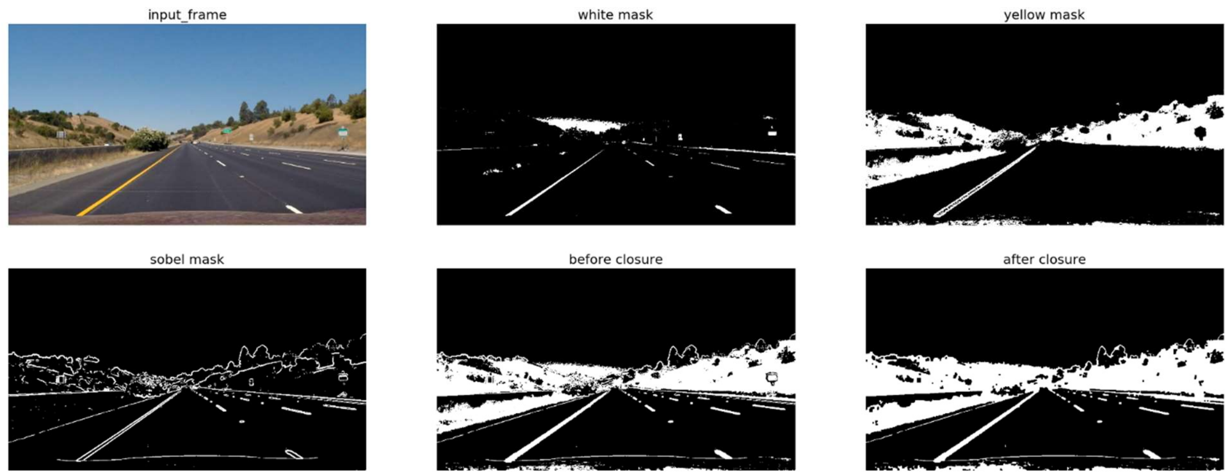
- **Cityscapes** and **CamVid** for semantic segmentation,
- **COCO** and **PASCAL VOC** for object detection,
- **CULane** and **TuSimple** for lane detection,
- **GTSRB** for traffic sign recognition.

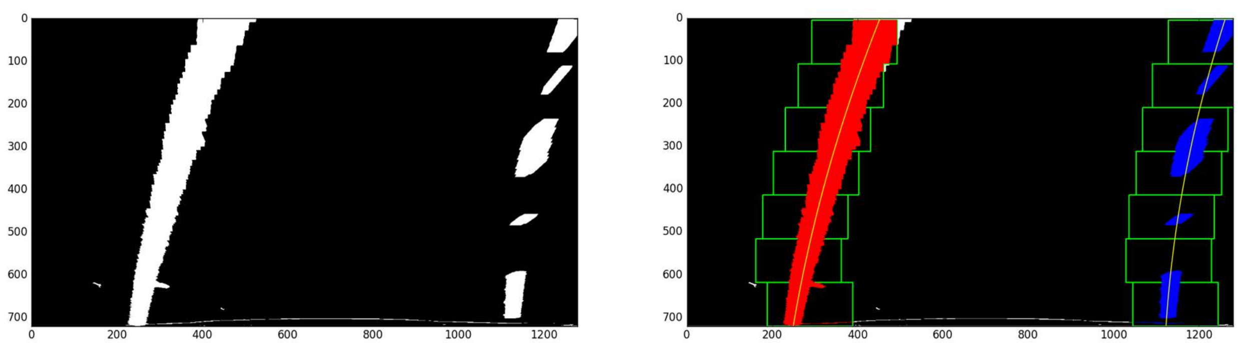
By training on these datasets, the model can generalize well to a wide variety of urban driving conditions and scenarios.

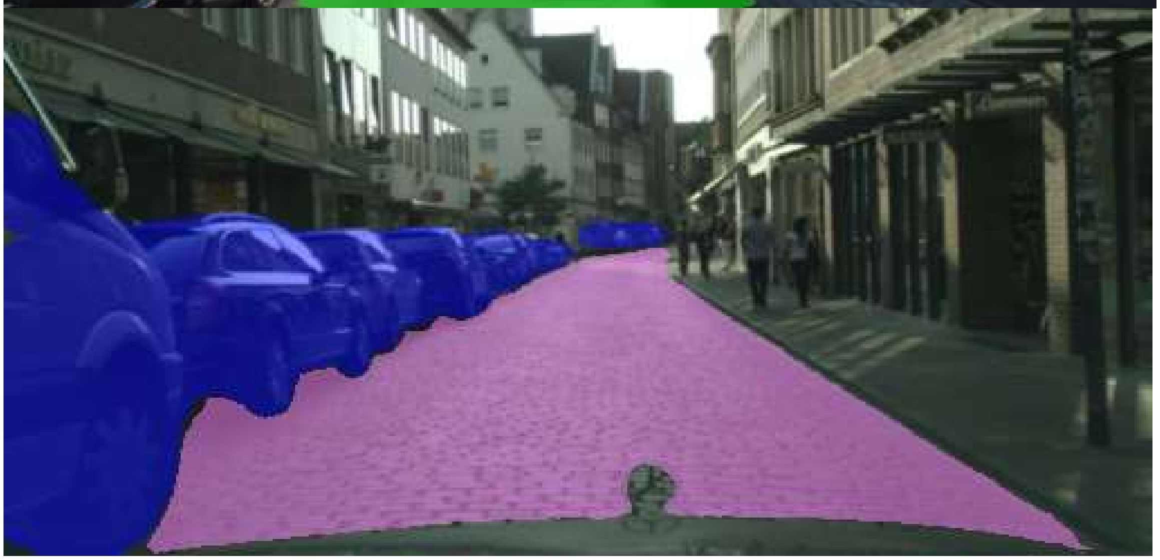
Conclusion

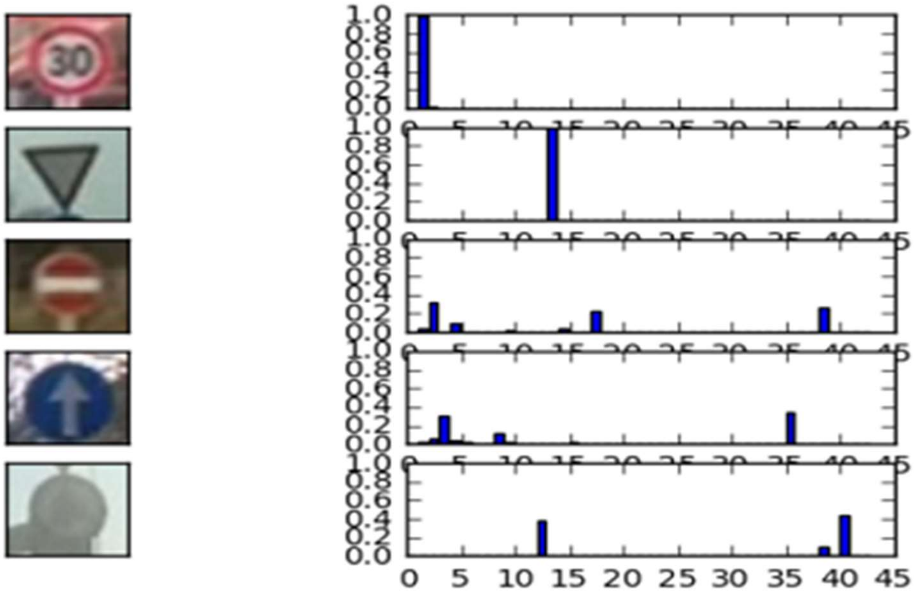
The proposed solution integrates multiple key tasks—semantic segmentation, object detection, lane detection, and traffic sign recognition—into a single deep learning model. This unified approach will enhance the autonomous vehicle's ability to navigate complex urban environments, making real-time decisions based on a detailed understanding of the scene.




input_frame
white mask
yellow mask
sobel mask
before closure
after closure


0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200




30
1.0
0.8
0.6
0.4
0.2
0.0
0 5 10 15 20 25 30 35 40 45

References

[1] A. A. Assidiq, O. O. Khalifa, M. R. Islam and S. Khan, "Real time lane detection for autonomous vehicles," 2008 International Conference on Computer and Communication Engineering, Kuala Lumpur, Malaysia, 2008, pp. 82-88, doi: 10.1109/ICCCE.2008.4580573. keywords: {Vehicle detection;Remotely operated vehicles;Mobile robots;Road vehicles;Road safety;Robustness;Road accidents;Machine vision;Cameras;Testing;Driver Assistance System;Lane detection;computer vision;intelligent vehicles},

[2] M. Masmoudi, H. Ghazzai, M. Frikha and Y. Massoud, "Object Detection Learning Techniques for Autonomous Vehicle Applications," 2019 IEEE International Conference on Vehicular Electronics and Safety (ICVES), Cairo, Egypt, 2019, pp. 1-5, doi: 10.1109/ICVES.2019.8906437. keywords: {Autonomous vehicle;deep learning;image processing;machine learning;vehicle detection},

[3] H. -H. Jebamikyous and R. Kashef, "Deep Learning-Based Semantic Segmentation in Autonomous Driving," 2021 IEEE 23rd Int Conf on High Performance Computing & Communications; 7th Int Conf on Data Science & Systems; 19th Int Conf on Smart City; 7th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys), Haikou, Hainan, China, 2021, pp. 1367-1373, doi: 10.1109/HPCC-DSS-SmartCity-DependSys53884.2021.00206. keywords: {Image segmentation;Visualization;Analytical models;Sensitivity;Smart cities;Semantics;Decision making;Autonomous Driving;Semantic Segmentation;U-Net;FCN;SegNet;Encoder-Decoder},

Code Attachments

The following is the partial / subset of the code. Code of some module(s) have been wilfully supressed.

A.1 Sample Code

```
class DLProgress(tqdm):
    last_block = 0

    def hook(self, block_num=1, block_size=1, total_size=None):
        self.total = total_size
        self.update((block_num - self.last_block) * block_size)
        self.last_block = block_num


def maybe_download_pretrained_vgg(data_dir):
    """
    Download and extract pretrained vgg model if it doesn't exist
    :param data_dir: Directory to download the model to
    """
    vgg_filename = 'vgg.zip'
    vgg_path = os.path.join(data_dir, 'vgg')
    vgg_files = [
        os.path.join(vgg_path, 'variables/variables.data-00000-of-00001'),
        os.path.join(vgg_path, 'variables/variables.index'),
        os.path.join(vgg_path, 'saved_model.pb')]
    missing_vgg_files = [vgg_file for vgg_file in vgg_files if not os.path.exists(vgg_file)]
    if missing_vgg_files:
        # Clean vgg dir
```

```
        if os.path.exists(vgg_path):
            shutil.rmtree(vgg_path)
        os.makedirs(vgg_path)
        # Download vgg
        print('Downloading pre-trained vgg model...')
        with DLProgress(unit='B', unit_scale=True, miniters=1) as pbar:
            urlretrieve(
                'https://s3-us-west-1.amazonaws.com/udacity-selfdrivingcar/vgg.zip',
                os.path.join(vgg_path, vgg_filename),
                pbar.hook)
        # Extract vgg
        print('Extracting model...')
        zip_ref = zipfile.ZipFile(os.path.join(vgg_path, vgg_filename), 'r')
        zip_ref.extractall(data_dir)
        zip_ref.close()

        # Remove zip file to save space
        os.remove(os.path.join(vgg_path, vgg_filename))
def img_size(img):
    return (img.shape[0], img.shape[1])

def random_crop(img, gt):
    h,w = img_size(img)
    nw = random.randint(1150, w-5) # Random crop size
    nh = int(nw / 3.3) # Keep original aspect ration
    x1 = random.randint(0, w - nw) # Random position of crop
    y1 = random.randint(0, h - nh)
    return img[y1:(y1+nh), x1:(x1+nw), :], gt[y1:(y1+nh), x1:(x1+nw), :]
```