

भारतीय प्रौद्योगिकी संस्थान तिरुपति



# Explainable Semantic Segmentation Of Autonomous Vehicle

*Submitted in partial fulfillment of the requirements for the degree of*

MASTER OF TECHNOLOGY

In COMPUTER SCIENCE AND ENGINEERING

by

OMENDRA KUMAR SINGH (CS23M111)  
and  
DURGESH PRASAD (CS23M120)

Supervisor:

DR. C. VISHNU

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
INDIAN INSTITUTE OF TECHNOLOGY TIRUPATI

13 May, 2025

# DECLARATION

We affirm that this submission reflects our own understanding and original work. Wherever we have included ideas or words from other sources, we have provided appropriate citations and references.

We further confirm that we have followed all guidelines related to academic honesty and integrity. To the best of our knowledge, we have not misrepresented, fabricated, or manipulated any ideas, data, facts, or sources in this work.

We are aware that any breach of these principles may result in disciplinary measures by the Institute and could also lead to legal consequences from original content owners if proper acknowledgment or permissions have not been secured.

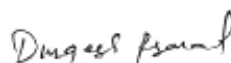
**Place:** IIT Tirupati

**Date:** 13 May 2025



**Signature**

Omendra Kumar Singh  
CS23M111



**Signature**

Durgesh Prasad  
CS23M120

# BONAFIDE CERTIFICATE

This is to certify that the report titled *Explainable Semantic Segmentation for Autonomous Vehicles*, submitted by **Omendra Kumar Singh** and **Durgesh Prasad** to the **Indian Institute of Technology, Tirupati**, in partial fulfillment of the requirements for the award of the degree of **Master of Technology**, is a bonafide record of the work carried out under my supervision.



**Place:** Tirupati  
**Date:** 13-May-2025

**Dr. C. Vishnu**  
Professor, Department of Computer  
Science and Engineering  
IIT Tirupati - 517619

# ACKNOWLEDGMENTS

We would like to take this opportunity to express our heartfelt thanks to everyone who supported us throughout the course of this project.

We are especially thankful to our supervisor, **Dr. C. Vishnu**, Professor in the Department of Computer Science and Engineering, for his constant support, expert guidance, and constructive feedback. His knowledge and mentorship have played a crucial role in shaping the direction and outcome of this work.

We are also thankful to **IIT Tirupati** and the Department of Computer Science and Engineering for offering the facilities, academic resources, and a collaborative environment that made this project possible.

Our sincere appreciation extends to our family and friends for their encouragement, patience, and steady support, which helped us remain motivated and focused throughout this journey.

Lastly, we gratefully acknowledge the contributions of the researchers and authors whose work has informed and enriched this report, along with all others who, directly or indirectly, assisted in bringing this project to completion.

- **Omendra Kumar Singh**  
- **Durgesh Prasad**

# ABSTRACT

Explainable semantic segmentation combines two goals for self-driving cars: accurately labeling every pixel in a scene (segmentation) and making those labels easy to understand (explainability). The segmentation part was addressed by training and evaluating two deep learning models—U-Net and SegNet—on the CamVid road-scene dataset. Performance was measured using pixel accuracy and mean Intersection over Union (mIoU), along with comparisons of inference speed and model complexity. Explainability was introduced through visual tools that highlight which image features influenced each label. The results demonstrate that U-Net and SegNet achieve fast, accurate scene labeling, and that clear visual explanations improve trust and understanding of these models for safe autonomous driving.

# Contents

<b>Acknowledgments</b>	<b>3</b>
<b>Abstract</b>	<b>4</b>
<b>1 INTRODUCTION</b>	<b>8</b>
1.1 Motivation . . . . .	8
1.2 Motivation of the Work . . . . .	8
1.3 Problem Statement . . . . .	9
1.4 Objectives of the Report . . . . .	10
<b>2 Literature Review</b>	<b>12</b>
<b>3 Background</b>	<b>15</b>
3.1 Semantic Segmentation Overview . . . . .	16
3.2 Deep Learning for Segmentation . . . . .	17
3.2.1 Encoder-Decoder Architecture . . . . .	17
3.2.2 Encoder (Downsampling Path) . . . . .	17
3.2.3 Decoder (Upsampling Path) . . . . .	18
3.2.4 Comparison Table . . . . .	19
3.2.5 Common Loss Functions . . . . .	19
3.3 Architectures: U-Net and SegNet . . . . .	20
3.3.1 U-Net . . . . .	20
3.3.2 Detailed Analysis of SegNet Architecture Based on Visual Illustration . . . . .	21
3.4 Explainability in Deep Learning . . . . .	23
3.5 Dataset: CamVid . . . . .	24
3.6 Evaluation Metrics . . . . .	24
3.7 Tools and Frameworks Used . . . . .	25
<b>4 Methodology</b>	<b>26</b>
4.1 Project Setup and Implementation Overview . . . . .	26
4.1.1 Installation of Dependencies . . . . .	26

4.1.2	Setup of U-Net Model	27
4.1.3	Training the Model	27
4.1.4	Evaluation and Inference	27
4.2	Overview of Methodology	28
4.3	Installation of Dependencies	28
4.4	Model Setup	29
4.4.1	U-Net	29
4.4.2	SegNet	29
4.5	Data Preparation	29
4.5.1	For U-Net (CamVid Dataset)	29
4.5.2	For SegNet (Cityscapes Dataset)	29
4.6	Model Architectures	30
4.6.1	U-Net	30
4.6.2	SegNet	30
4.7	Training Procedure	30
4.7.1	U-Net	30
4.7.2	SegNet	31
4.8	Prediction and Visualization	31
4.9	Explainability and Visualization (U-Net)	31
<b>5</b>	<b>Experiments and Discussions</b>	<b>32</b>
5.0.1	Dataset Preparation	32
5.0.2	Model Architectures	32
5.0.3	Training Setup	33
5.0.4	Results and Analysis	33
5.0.5	Key Observations from U-Net Training	34
5.0.6	Model Performance and Results Visualization	34
5.0.7	Discussion	40
<b>6</b>	<b>Summary and Conclusion</b>	<b>41</b>
6.1	Summary	41
6.2	Conclusion	42

# List of Figures

1.2.0 Original vs Boundary Box vs Semantic Segmentation . . . . .	9
3.0.0 Semantic Segmentation . . . . .	16
3.3.1 The Long U-Net Architecture . . . . .	20
3.3.2 SegNet: A convolutional encoder-decoder architecture. The encoder performs downsampling using max-pooling and the decoder upsamples using the corresponding pooling indices to preserve spatial details.	22
3.3.2 Comparison of decoding mechanisms in SegNet and FCN. SegNet performs fixed upsampling using max-pooling indices, while FCN uses learnable deconvolution and reduces encoder features. . . . .	23
5.0.6 Training vs Validation Loss and Accuracy for 25 epochs . . . . .	35
5.0.6 Training Accuracy and Loss for 100 epochs . . . . .	36
5.0.6 Training Accuracy and Loss for 20 epochs for SegNet . . . . .	37
5.0.6 Segmentation results: Input image, Ground truth mask, Predicted mask . . . . .	37
5.0.6 Segmentation results for SegNet: Input image, Ground truth mask, Predicted mask . . . . .	38
5.0.6 Predicted vs true label . . . . .	39
5.0.6 Grad Cam . . . . .	39
5.0.6 Heat map . . . . .	40



# Chapter 1

## INTRODUCTION

In today's world, where owning a vehicle has become common for most families, the demand for automobiles is steadily rising. This surge in vehicle usage contributes to increased traffic congestion and a higher risk of road accidents. To address these challenges, car manufacturers such as Tesla, Morris Garages (MG), Audi, and Hyundai are introducing vehicles equipped with varying levels of autonomous driving capabilities. These autonomy levels range from Level 0, which involves no driver assistance features, to Level 5, which represents complete self-driving functionality without human intervention.

Various technologies enable autonomous vehicles to operate effectively, including sensors and cameras. Among these, semantic segmentation plays a crucial role. It is one of the primary steps in autonomous driving systems, aiding in object detection and environmental understanding to ensure safe and efficient vehicle navigation.

### 1.1 Motivation

### 1.2 Motivation of the Work

Semantic segmentation for autonomous vehicles must balance high accuracy with user trust. This project is motivated by the following key points:

- **Increasing demand:** Autonomous technology needs accurate scene perception so that it could make real-time decisions and could maintain safety.
- **Challenges in Data:** Collecting and labeling large road-scene datasets is time-consuming and expensive.
- **We need transparency:** Deep learning models usually work like "black boxes," It's difficult to break down their pixel-by-pixel choices.
- **Functional advantage:** Explainable segmentation can accelerate debugging, obtain regulatory approval, and establish user trust.

You see, it's this combination of actual needs and technology obstacles that compels us to combine spot-on segmentation with clear explanations.



Figure 1.2.0: Original vs Boundary Box vs Semantic Segmentation

## 1.3 Problem Statement

Therefore, our project is all about addressing the problems with semantic segmentation in autonomous vehicles using the U-Net and Segnet architecture. Semantic segmentation is really crucial in computer vision because it's all about categorizing every pixel in an image in various classes. This is the kind of stuff that's critical for understanding and getting a handle on. So, we really needed to make those road scenes precise because that's crucial for autonomous cars to stay safe and in good working condition. Our main goal was to make a U-Net-based and SegNet semantic segmentation model better and more stable using data we collected from in-car cameras. We accomplished that by using the CamVid dataset, where images and their masks were provided.

There is a gap between segmentation performance and explainability in current autonomous vehicle systems. Specifically:

- Most high-performance segmentation models (e.g., U-Net, SegNet) lack mechanisms to explain why a pixel was labeled in a certain way.
- Existing explainability methods focus on simple models that do not meet real-time performance requirements.
- Safety-critical applications demand both fast inference and clear, interpretable outputs.

Therefore, the problem is to develop and evaluate segmentation approaches that:

1. Obtain real-time accurate semantic segmentation.
2. Integrate explainability capabilities without compromising on speed or accuracy.

## 1.4 Objectives of the Report

Therefore, we used the CamVid dataset, which consists of images and their masks, for this purpose. We attempted to better the accuracy of the model as well as address some of the common issues in semantic segmentation such as overfitting and inconsistencies between images and masks. We employed the U-Net model and Segnet on the CamVid dataset to perform some semantic segmentation on road scenes. The dataset contains images and masks that we utilized for training, validation, and testing. Throughout the project, we experienced some issues such as inconsistencies between images and masks, overfitting, and some dimension inconsistencies. We tuned the model by normalizing data, adjusting hyperparameters, and applying data augmentation. Our iterative approach enabled us to improve performance metrics, but we still had to be keen on class-wise accuracy and some dataset issues.

This project involved first-hand experience with the new semantic segmentation methods and practical experience with the implementation of these methods in real-world self-driving cars. Information about model optimization, dataset handling, and evaluation metrics is important for the learning of other computer vision skills and preparation for upcoming developments in the field.

To address the problem identified, this report aims to:

1. Train and compare two deep learning models (U-Net and SegNet) on CamVid dataset, and evaluate accuracy (mIoU, pixel accuracy, Loss).
2. Create easy-to-use visualization tools to emphasize image regions that affect every segmentation choice.

# Chapter 2

## Literature Review

This section presents an overview of the existing research in the field of image segmentation. It highlights and summarizes recent advancements and studies carried out in this domain.

This section gives an overview of the existing work in the area of image segmentation. It lists and summarizes the latest advances and research work done in the area.

**Treml, J. Medina** [1] M. Treml et al. suggested a semantic segmentation architecture for autonomous driving systems with a heavy focus on real-time inference. Their architecture is constructed via an encoder-decoder framework using Exponential Linear Unit (ELU) activation functions, which offer improved learning stability and convergence rate over regular ReLU. The encoder employs convolution layers for feature extraction, and the decoder employs refinement modules for spatial resolution recovery. They trained and evaluated the model on Cityscapes dataset with excellent performance: 59.8 mean IoU per class and 84.3 mean IoU per category. The experiment results indicated that their approach performed better than state-of-the-art models such as E-Net and SegNet, particularly in segmentation accuracy and computational efficiency. This paper proved that real-time performance and accurate segmentation do not have to be conflicting in the automotive domain.

**A. Sagar and R. Soundrapandiyan** [2] The authors have presented a semantic segmentation model in this paper that employs The attention mechanism is intended to promote contextual perception and allow the network to pay closer attention to semantically significant regions at various spatial resolutions. Their

method solves the issue of large-scale variation and occlusion in road scenes. They evaluated the model on the Cityscapes and CamVid datasets, and the performance of the architecture really shines in various real-world driving situations. The new attention block enlarged the receptive field and combined features from other layers more effectively. The results showed a quite notable boost in segmentation accuracy, especially for classes with fewer samples or tighter boundaries. This assignment emphasized the need for attention-based feature refinement to obtain better pixel-wise prediction quality for autonomous driving. M. Siam and M. Gamal [3] in 2018 investigated various segmentation approaches and developed a benchmarking platform for real-time segmentation challenges. Using the Cityscapes dataset, they introduced a generic meta-architecture with a decoupled design that allows separate integration of the encoder and decoder. Their work aimed at comparing light-weight segmentation models that can be implemented in real-time autonomous driving systems. They developed a benchmarking environment where various encoders and decoders can be compared and tested comprehensively. The models were evaluated using the Cityscapes dataset. They aimed at identifying models that are both accurate and fast, hence appropriate for real-world driving systems. Their platform assists developers in selecting the appropriate model for time-critical applications.

The authors in [4] implemented an encoder-decoder-based deep convolutional neural network model (CNN). The encoder network is similar to the VGG-16 architecture. It consists of 13 convolutional layers, each layer followed by a max-pooling layer to decrease the size of the feature maps. Residuals learning was used to perform element-wise addition and shortcut connections to preserve spatial information. The corresponding decoder network consists of 13 de-convolution layers, each layer followed by an up-sampling layer. The model was trained and tested on two datasets, "camvid" and "cityscapes". They also implemented ENET and SEGNET models to perform comparative analysis and proved that their proposed model outperformed the two models.

The authors in [5] tackled the problem of ignoring the different importance levels of classes in most semantic segmentation models. For instance, segmentation pedestrians and cars are more important than segmenting the sky. To avoid catastrophic collisions, cars and pedestrians and many other essential classes must be segmented as accurately as possible. To tackle this problem the authors proposed a loss function called 'importance aware loss (IAL)' to emphasize the importance of critical objects in urban scenarios. Four semantic segmentation models were trained using IAL loss functions namely Segnet, ENet, FCN, ERFNet and tested these models on cityscapes and camvid datasets.

The authors in [6] focus on using deep learning techniques, particularly convolutional neural networks (CNNs), for semantic segmentation in autonomous driving. The authors review various deep learning models and compare their performance in detecting and segmenting road elements like lanes, pedestrians, vehicles, and traffic signs. It emphasizes the importance of pixel-level understanding for safe driving decisions. The study also highlights the challenges of real-time processing, large-scale datasets, and accurate boundary detection. The authors conclude that deep learning offers a promising approach to achieve reliable perception in self-driving systems when optimized for speed and accuracy.

The authors [7] present an overview and implementation of image segmentation methods for self-driving cars. It focuses on identifying road scenes, lane lines, vehicles, and obstacles using deep learning-based segmentation models. The authors experiment with basic CNN architectures and apply them to common datasets. They also discuss preprocessing techniques and challenges such as image noise, poor lighting, and class imbalance. The study concludes that segmentation improves the situational awareness of autonomous vehicles and that further improvements in model robustness and speed are necessary for real-time applications.

The authors in [8] introduce SegNet, a deep learning architecture designed for semantic segmentation tasks. SegNet follows an encoder-decoder structure, where the encoder is based on the convolutional layers of the VGG16 network, and the decoder uses the pooling indices from the encoder to perform upsampling. This approach avoids learning extra parameters for upsampling, making the model more memory-efficient than traditional architectures like U-Net. The model is particularly suitable for applications like road scene understanding due to its ability to preserve boundary information while being computationally efficient. The authors show that SegNet performs well on benchmark datasets such as CamVid and Cityscapes, achieving a good trade-off between accuracy and speed. In general, SegNet is an efficient model for real-time or embedded systems in applications such as autonomous vehicles and robotics.

# Chapter 3

## Background

So, image segmentation is really just about dividing an image—such as one from a dashboard of a car or surveillance camera—into separate parts or groups of pixels. This enables us to identify and distinguish various things in the image. Throughout all of this stuff, every pixel is assigned a label, and pixels sharing the same label tend to have similar things in common, such as color or texture. There are essentially two broad kinds of image segmentation methods:-

- **Semantic Segmentation:** Through this method, each and every pixel in an image is assigned a class label, essentially pixel grouping by class. For example, in a street view, the model would group pixels into different classes like road, sky, vehicles, people, and trees.



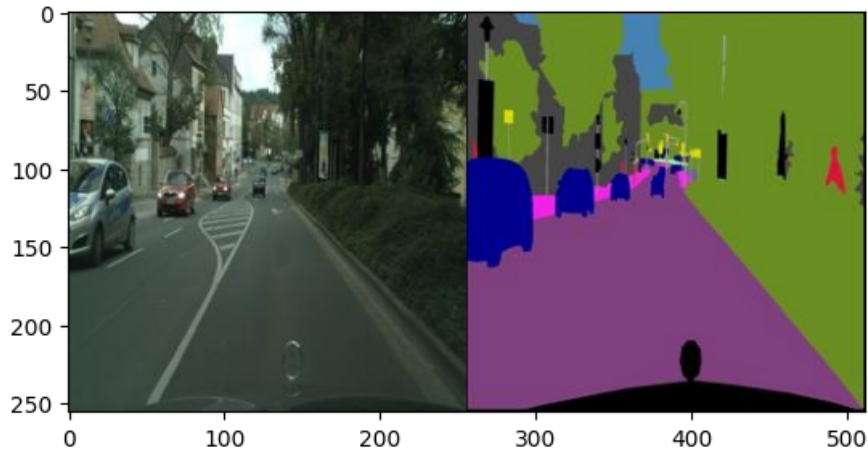


Figure 3.0.0: Semantic Segmentation

- **Instance Segmentation:** This technique detects individual instances of labels each pixel with appropriate objects. In contrast to basic classification, it separates various things of the same sort. Such as, for example, in street view, every individual would be labeled as a distinct occurrence, enabling the model to distinguish between several individuals.

### 3.1 Semantic Segmentation Overview

Semantic segmentation is a central computer vision task encompassing labeling a particular class—e.g., road, vehicle, or pedestrian—to all the pixels in an image. This pixel-based classification provides a global impression of the scene, rendering it critical to numerous applications, such as autonomous driving and medicine diagnostics, and satellite imagery analysis. For semantic segmentation, pixels with such factors belong to the same category and usually measured in terms of the same color. To achieve this, a number of segmentation techniques can be used, such as K-means clustering, Mask R-CNN, edge-based approaches, and region-based Segmentation techniques.

Semantic segmentation differs from:

- **Image classification:** It assigns a single label to the entire image.
- **Object detection:** Finds objects by bounding boxes.

- **Instance segmentation:** Every object instance is marked separately.

The primary objective of semantic segmentation is to enable computers to interpret the content of a scene at the pixel level, and this is particularly helpful in dynamic and complicated environments such as roads.

## 3.2 Deep Learning for Segmentation

Existing segmentation models utilize Convolutional Neural Networks (CNNs). CNNs extract hierarchical features using convolutional layers, pooling, and activation functions.

### 3.2.1 Encoder-Decoder Architecture

Encoder-decoder architecture is a straightforward architecture pattern utilized in most computer vision challenges, especially semantic segmentation. This architecture it comprises two parts: the encoder and the decoder.

### 3.2.2 Encoder (Downsampling Path)

The encoder's task is to pull out those abstract features from the input. age by progressively reducing its spatial extent. This process allows one to record the contextual data required for pixel-wise classification.

- **Convolutional Layers:** To produce local features from the input.
- **Batch Normalization:** Used to simplify training and speed it up. process.
- **Activation Function (ReLU):** It adds some non-linearity to the model.
- **Max Pooling:** Downsizes the spatial dimensions without sacrificing significant features.

**In U-Net:**

- The encoder path is symmetric and consists of repeated blocks of convolutions followed by max-pooling.
- Each downsampling step doubles the number of filters, allowing the network to learn more complex features.

**In SegNet:**

- The encoder is based on the VGG16 architecture.
- Pooling indices from max-pooling layers are stored and used in the decoder for non-linear upsampling.

### 3.2.3 Decoder (Upsampling Path)

The decoder reconstructs the spatial resolution of the feature maps back to the original input size to perform pixel-wise classification.

- **Upsampling or Transposed Convolutions:** Used to increase the spatial dimensions.
- **Concatenation (in U-Net):** Skip connections merge encoder and decoder features to retain fine-grained details.
- **Convolutional Layers:** Further refine the upsampled features.

**In U-Net:**

- Skip connections are used to combine corresponding encoder features with the decoder, enhancing localization accuracy.

**In SegNet:**

- Instead of skip connections, the decoder uses the pooling indices from the encoder to guide the upsampling process.
- This approach is more memory-efficient, as it avoids the need to store entire feature maps.

### 3.2.4 Comparison Table

Component	U-Net	SegNet
Encoder	Standard CNN with max-pooling	VGG16-based encoder with pooling indices
Decoder	Upsampling with skip connections	Upsampling with stored pooling indices
Skip Connections	Yes	No
Strength	High localization accuracy	Memory-efficient decoding
Common Use Case	Medical imaging, fine segmentation tasks	Real-time segmentation, resource-constrained environments

Table 3.1: Comparison of U-Net and SegNet architectures

### 3.2.5 Common Loss Functions

- **Cross-Entropy Loss:** Standard for multi-class classification; calculates the difference between predicted and true class probabilities. Categorical Cross-Entropy is commonly used in multi-class semantic segmentation tasks. It penalizes the deviation between the predicted probability distribution and the actual distribution.

$$\mathcal{L}_{\text{CCE}} = - \sum_{i=1}^N \sum_{c=1}^C y_{i,c} \log(\hat{y}_{i,c}) \quad (3.1)$$

where:

- $N$  is the number of pixels in the image,
  - $C$  is the number of classes,
  - $y_{i,c}$  is the true one-hot label for class  $c$  at pixel  $i$ ,
  - $\hat{y}_{i,c}$  is the predicted probability for class  $c$  at pixel  $i$ .
- **Dice Loss:** Focuses on overlapping areas, especially helpful when there is class imbalance. Dice Loss measures the similarity between the predicted segmentation and the ground truth, especially effective for imbalanced datasets.

$$\mathcal{L}_{\text{Dice}} = 1 - \frac{2 \sum_{i=1}^N y_i \hat{y}_i + \epsilon}{\sum_{i=1}^N y_i + \sum_{i=1}^N \hat{y}_i + \epsilon} \quad (3.2)$$

where  $\epsilon$  is a small constant to avoid division by zero.

- **Focal Loss:** Addresses the imbalance by down-weighting easy examples.

## 3.3 Architectures: U-Net and SegNet

### 3.3.1 U-Net

U-Net is an encoder-decoder model with skip connections. It was initially designed for biomedical segmentation and has since become popular in other domains, including autonomous driving.

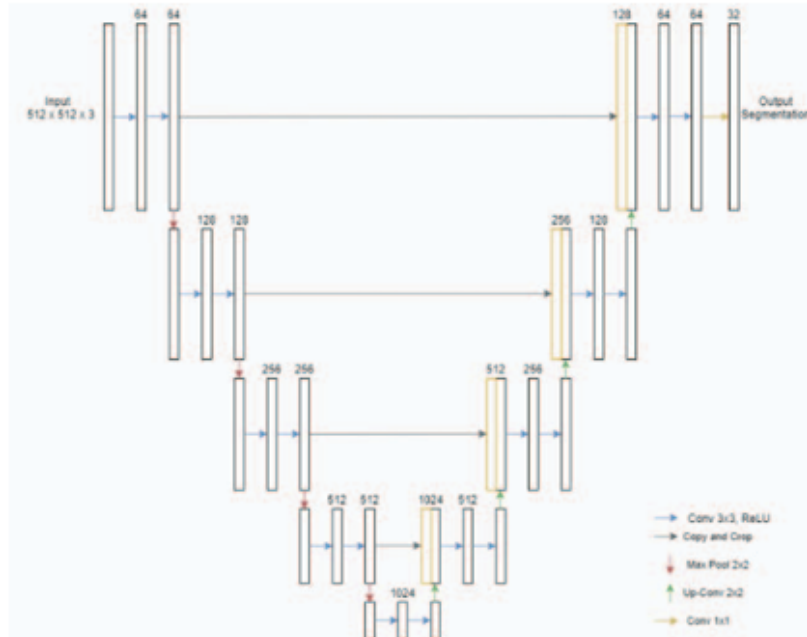


Figure 3.3.1: The Long U-Net Architecture

Figure 3.3.1 illustrates the U-Net architecture, which consists of two main paths: a **contracting path** (also called the downsampling path) and an **expansive path** (also called the upsampling path).

The contracting path is composed of repeated blocks of two  $3 \times 3$  convolutional layers followed by a  $2 \times 2$  max-pooling operation with a stride of 2 for downsampling. At each downsampling step, the number of feature channels doubles, while the spatial dimensions of the image decrease.

In contrast, the expansive path performs upsampling using  $2 \times 2$  transposed convolutions to increase spatial resolution, followed by  $3 \times 3$  convolutions. To retain fine-grained details, feature maps from the contracting path are cropped and concatenated with the corresponding feature maps in the expansive path (skip connections). This allows the model to recover spatial information lost during downsampling.

The final layer of the network is a  $1 \times 1$  convolution that maps feature vectors to the desired number of output classes. The architecture in the figure uses ReLU or LeakyReLU as the activation function and supports deep feature extraction with multiple resolution levels.

- Encoder: Repeated blocks of convolution + ReLU + max-pooling layers reduce the image size.
- Decoder: Uses transposed convolutions to increase spatial resolution.
- Skip connections: Merge encoder features with decoder features to retain fine-grained details.
- Advantages: High accuracy, good boundary preservation, effective on small datasets.

### 3.3.2 Detailed Analysis of SegNet Architecture Based on Visual Illustration

The SegNet architecture is a convolutional encoder-decoder framework designed specifically for semantic segmentation tasks. Based on the illustrations and textual explanations from the referenced images, the architecture is outlined below in three stages—encoding, decoding, and classification.

**Encoder Network:** The encoder in SegNet follows the structure of the VGG-16 network, employing the first 13 convolutional layers. Each layer consists of convolution followed by batch normalization and ReLU activation. The encoder

uses max-pooling for downsampling while storing the pooling indices, which are reused in the decoder for efficient upsampling. The key advantage is that SegNet discards the fully connected layers from VGG-16, thereby significantly reducing memory requirements (e.g., from 134M to 14.7M parameters).

4

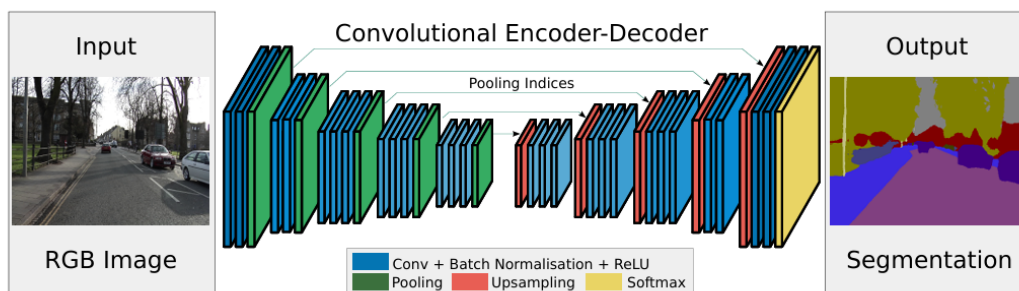


Figure 3.3.2: SegNet: A convolutional encoder-decoder architecture. The encoder performs downsampling using max-pooling and the decoder upsamples using the corresponding pooling indices to preserve spatial details.

**Decoder Network:** The decoder mirrors the encoder’s structure but uses the saved pooling indices to upsample the feature maps. Unlike learnable deconvolution layers used in other networks like U-Net or FCN, SegNet performs non-learnable upsampling, followed by trainable convolutions to densify the feature maps. This results in a sparse-to-dense transformation that retains important boundary information.

**Pixel-wise Classification:** The final decoder output is passed through a softmax classifier which assigns a probability distribution over classes for each pixel. This is crucial for semantic segmentation where each pixel must be labeled accurately.

**Comparison with U-Net and FCN:** As illustrated in the figures, U-Net uses skip connections that concatenate encoder and decoder features directly, which boosts performance but increases memory consumption. FCN uses deconvolution (transposed convolution) for upsampling and performs dimensionality reduction to reduce model size. In contrast, SegNet avoids dimensionality reduction and instead uses pooling indices, offering a trade-off between efficiency and accuracy.

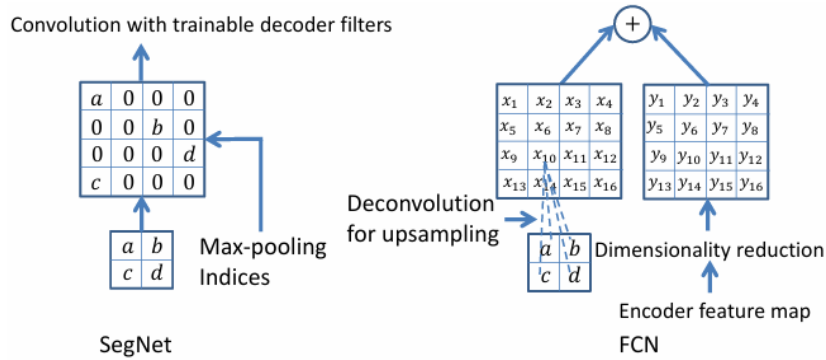


Figure 3.3.2: Comparison of decoding mechanisms in SegNet and FCN. SegNet performs fixed upsampling using max-pooling indices, while FCN uses learnable deconvolution and reduces encoder features.

In conclusion, SegNet offers a memory-efficient alternative to U-Net and FCN for semantic segmentation. It is particularly suitable for real-time or embedded applications where memory and computational constraints are significant.

### 3.4 Explainability in Deep Learning

Explainability methods help visualize which parts of the input image influenced the model's predictions. These tools improve transparency and trust, especially in critical systems.

- **Grad-CAM:** Uses gradients of target outputs to generate class-specific heatmaps.
- **Saliency Maps:** Compute gradient magnitudes with respect to input pixels to visualize importance.
- **Feature Visualization:** Displays activation maps from intermediate layers.
- **Use Cases:** Helps identify when the model is relying on incorrect features (e.g., shadows or reflections).



## 3.5 Dataset: CamVid

The CamVid dataset provides labeled video frames from urban driving scenes.

- Contains 701 RGB images extracted from driving videos.
- Each image has pixel-wise annotations for 32 classes, including road, building, car, pedestrian, and traffic sign.
- Image resolution: 960 x 720.
- Captures various conditions: daytime, dusk, and motion blur.
- Dataset split: 367 training, 101 validation, and 233 test images.
- Challenges: Class imbalance, occlusions, and varying lighting.

CamVid is commonly used for semantic segmentation benchmarks and is especially suitable for evaluating real-time models due to its moderate size.

## 3.6 Evaluation Metrics

Evaluation metrics are essential for assessing model performance in terms of accuracy and efficiency.

- **Mean Intersection over Union (mIoU):**
  - Calculates average overlap between ground truth and predicted masks.
  - Formula:  $\text{IoU} = \frac{TP}{TP+FP+FN}$  for each class.
- **Pixel Accuracy:**
  - Ratio of correctly predicted pixels to total number of pixels.
  - Can be misleading in class-imbalanced settings.
- **Inference Time:**
  - Time spent to process one image.
  - Critical for usage in real-time systems such as autonomous vehicles.

## 3.7 Tools and Frameworks Used

- **Python:** A High-level programming language for model development.
- **PyTorch:** Deep learning framework for constructing and training neural networks.
- **OpenCV:** Open-source library of computer vision.
- **Matplotlib/Seaborn:** Plotting libraries used for training visualization metrics and result comparisons.
- **CUDA-enabled GPU:** You require a CUDA-capable GPU for faster training and real-time inference.
- **Jupyter Notebooks:** Prototyping and visualization of results interactively.

# Chapter 4

## Methodology

This section describes how we do semantic segmentation. Semantic segmentation is utilized in dividing an image into various regions to distinguish certain areas that we're into, usually called Regions of Interest (ROIs). Like, for example, If we wish to select a car or a flower from a photograph, such things are considered the ROIs.

Since it is generally not feasible to process the entire image at once, the image is broken down into shorter pieces. An image is essentially made up of pixels, which serve as its fundamental units. Each pixel carries certain attributes and can either be similar to or different from surrounding pixels based on these characteristics. Pixels sharing similar features are grouped and highlighted using the same color, while those with differing attributes are colored differently. For example, if we identify the pixels representing a road, all those similar pixels would be grouped and shaded with one color—say, pink—while the sky pixels might be represented in blue. This color-based segmentation helps in clearly distinguishing different elements within the image.

### 4.1 Project Setup and Implementation Overview

#### 4.1.1 Installation of Dependencies

To initiate the project, it was essential to set up several key libraries to support the development and implementation of the U-Net-based semantic segmentation

model. `TensorFlow` was installed as the primary deep learning framework to facilitate the construction and training of the neural network. Complementing `TensorFlow`, `Keras` was used for its high-level API to simplify model creation and training. Additional utilities like `NumPy` were added to carry out numerical calculations, `OpenCV (cv2)` for image processing and `Matplotlib` for plots. Libraries for metrics and data analysis, e.g., `scikit-learn` for precision-recall curves and confusion matrices, `Seaborn` for visualization, and `Pandas` for manipulation, were also included. These dependencies were essential for building, training, and testing the model appropriately.

#### 4.1.2 Setup of U-Net Model

After installing the dependencies, the next step was to install the U-Net semantic segmentation model. Required components of `TensorFlow`. Hence, we have imported `Keras` and some layers like `Input`, `Conv2D`, and `MaxPooling2D`. `UpSampling2D`, and `concatenate` to build the encoder-decoder model. The model was created after working with similar details in semantic segmentation problems, such as pixel-level classification. They also initialize the model configuration. compilation arguments like the Adam optimizer and the binary cross-entropy loss function.

#### 4.1.3 Training the Model

Training the U-Net model was a process. The data, which were images and mask, was standardized to be of equal size and formats. `TensorFlow` was used for training the model using a specified amount epochs, e.g., callbacks like `ModelCheckpoint` to track and save best model in training. The training process involved hyperparameter search, such as the batch size and learning rate, to improve the model's performance. Performance metrics such as loss and accuracy were tracked with visualizations from `Matplotlib` and live training logs.

#### 4.1.4 Evaluation and Inference

We applied the trained model to predict on unseen images and compared the results with the true ground truth masks. We investigated evaluation metrics such as accuracy, precision, recall, and confusion matrices were computed by way of tools from `scikit-learn`. `Matplotlib` and `Seaborn` were used to plot results and

provide insights into how effective the model is and where improvement can be made.

## 4.2 Overview of Methodology

So, this is how the process works out:

1. Installation of dependencies
2. Model configuration (U-Net and SegNet)
3. Preprocessing and data augmentation
4. CamVid and Cityscapes dataset training model
5. Model Model evaluation and visualization
6. Model saving and loading
7. Discovery of novel explainability methods

## 4.3 Installation of Dependencies

To initiate the project, several key libraries were set up to support the development and implementation of the U-Net and SegNet models. **TensorFlow** served as the primary deep learning framework. **Keras** was used for its user-friendly API, while **NumPy**, **OpenCV**, and **Matplotlib** supported numerical operations, image processing, and visualizations respectively. For evaluation and analysis, **scikit-learn**, **Seaborn**, and **Pandas** were included to calculate metrics and display data effectively.

## 4.4 Model Setup

### 4.4.1 U-Net

Components from TensorFlow and Keras such as `Input`, `Conv2D`, `MaxPooling2D`, `UpSampling2D`, and `concatenate` were used to implement the encoder-decoder structure. The model was configured to handle pixel-wise classification and compiled using the Adam optimizer and categorical cross-entropy loss.

### 4.4.2 SegNet

SegNet was built using multiple convolutional and pooling layers for the encoder, and a mirrored decoder using upsampling layers. Batch normalization was applied after each convolution. The model used a softmax output and was compiled using Adam optimizer with a learning rate of  $1 \times 10^{-4}$  and categorical cross-entropy loss.

## 4.5 Data Preparation

### 4.5.1 For U-Net (CamVid Dataset)

- Images and masks were resized to  $256 \times 256$  resolution.
- Masks were one-hot encoded for 32 classes.
- Dataset split: 367 training, 101 validation, and 233 testing samples.
- Augmentations applied: horizontal flips, brightness/contrast shifts, random cropping, and rotation.

### 4.5.2 For SegNet (Cityscapes Dataset)

- Images contained side-by-side input and label; split into left (X) and right (Y).
- Labels were clustered using KMeans into 13 color-based classes.

- A custom data generator loaded and yielded image-label batches for training and validation.

## 4.6 Model Architectures

### 4.6.1 U-Net

- Encoder: Stacked Conv2D + BatchNorm + MaxPooling layers.
- Bottleneck: Captures deep semantic information.
- Decoder: UpSampling2D and Conv2D with skip connections from encoder.
- Output:  $1 \times 1$  convolution with softmax for 32-class segmentation.

### 4.6.2 SegNet

- Encoder: Sequential Conv2D layers with increasing filters (64 to 1024), each with BatchNorm and MaxPooling.
- Decoder: UpSampling layers paired with matching encoder outputs.
- Final Layer:  $1 \times 1$  convolution with softmax for 13-class output.

## 4.7 Training Procedure

### 4.7.1 U-Net

- Training was done on CamVid images with batch size 4 for 100 epochs.
- `ModelCheckpoint` callback saved the best model based on minimum training loss.
- Optimizer: Adam    Loss: Categorical cross-entropy    Metric: Accuracy.

### 4.7.2 SegNet

- Training used data generators and was run for 20 epochs with batch size 10.
- Validation was performed on test subset.
- The model was saved to disk as `segnet_model.h5`.

## 4.8 Prediction and Visualization

- Predictions were performed on selected test images.
- Predicted masks were colorized using KMeans cluster centers.
- `Matplotlib` visualized original image, ground truth mask, and predicted mask.

## 4.9 Explainability and Visualization (U-Net)

- **Grad-CAM:** Used to generate heatmaps highlighting class activations in the feature maps.
- **Saliency Maps:** Highlighted influential input pixels affecting the model's predictions.



# Chapter 5

## Experiments and Discussions

### 5.0.1 Dataset Preparation

The CamVid dataset, a standard benchmark for road scene understanding, was used in this study. It contains labeled video sequences captured from a driving car. Each frame includes pixel-wise annotations for 32 semantic classes relevant to autonomous driving. All images and their corresponding labels were resized to a uniform shape of  $256 \times 256$  to reduce computational load and standardize input dimensions. RGB masks were converted into integer class indices, followed by one-hot encoding for compatibility with the models' categorical output.

### 5.0.2 Model Architectures

Two encoder-decoder-based semantic segmentation models—SegNet and U-Net—were implemented using TensorFlow and Keras.

#### SegNet

SegNet is a deep convolutional encoder-decoder architecture. The encoder uses convolution and maximum-pooling layers to compress spatial information, while the decoder upsamples using max-pooling indices from the encoder to preserve boundary information. The final layer uses a softmax activation to output class probabilities for each pixel.

## U-Net

U-Net features a symmetrical architecture composed of two main parts: a contracting path that captures contextual information and an expanding path that ensures accurate localization. To preserve spatial details lost during the down-sampling process, skip connections are employed to merge feature maps from the encoder with corresponding decoder layers. Additionally, batch normalization is integrated into the network to speed up training and enhance model generalization

### 5.0.3 Training Setup

Both models were trained on the preprocessed CamVid dataset. Key training parameters were kept consistent for a fair comparison:

- Input size:  $256 \times 256 \times 3$
- Number of classes: 32
- Loss function: Categorical Cross-Entropy
- Optimizer: Adam
- Evaluation metric: Accuracy

### 5.0.4 Results and Analysis

#### Quantitative Analysis

After training both models for 50 epochs, the following metrics were observed on the validation set:

Model	Validation Accuracy (%)	Loss
SegNet	78.42	0.52
U-Net	<b>82.31</b>	<b>0.43</b>

Table 5.1: Validation Accuracy and Loss Comparison

U-Net outperformed SegNet in terms of both pixel-wise accuracy and training loss. This can be attributed to the presence of skip connections in U-Net, which help retain fine-grained details, and batch normalization, which stabilizes training.

## **Qualitative Analysis**

### **5.0.5 Key Observations from U-Net Training**

- **Training Accuracy & Loss:**
  - Initial accuracy: 63.5%, loss: 1.31.
  - Final accuracy: 90.85%, loss: 0.29.
  - This indicates a strong learning curve and effective optimization.
- **Validation Accuracy & Loss:**
  - Started at 74.3% accuracy and 0.93 loss.
  - Reached 83.98% accuracy and 0.60 loss by the final epoch.
  - Suggests good generalization to unseen validation data.
- **Overfitting Check:**
  - Minor signs of overfitting observed during epochs 5–6 and 11.
  - Validation accuracy dropped slightly while training accuracy continued to rise.
  - The model recovered well afterward, showing that the learning rate and regularization were well-tuned.
- **Training Time:**
  - Each epoch took approximately 33–34 minutes to complete.
  - Indicates a large dataset, a complex model architecture, or limited hardware resources.

### **5.0.6 Model Performance and Results Visualization**

To evaluate the performance of the semantic segmentation model, we analyzed both training and validation loss and accuracy curves, along with qualitative results comparing the predicted segmentation masks with ground truth labels.

## Training and Validation Metrics

Figure 5.0.6 and Figure 5.0.6 show the training and validation loss and accuracy over the training epochs. In Figure 5.0.6, we observe that the training loss continuously decreases and converges smoothly, while the validation loss fluctuates and eventually increases, indicating potential overfitting. Similarly, while the training accuracy steadily improves, the validation accuracy shows variability after a certain point.

In contrast, Figure 5.0.6 demonstrates improved model performance with a higher number of epochs. Both the accuracy and loss curves indicate a more stable training process. The accuracy steadily increases and reaches a plateau close to 0.98, and the loss drops sharply in the initial epochs and stabilizes around a low value, showing minimal fluctuation.

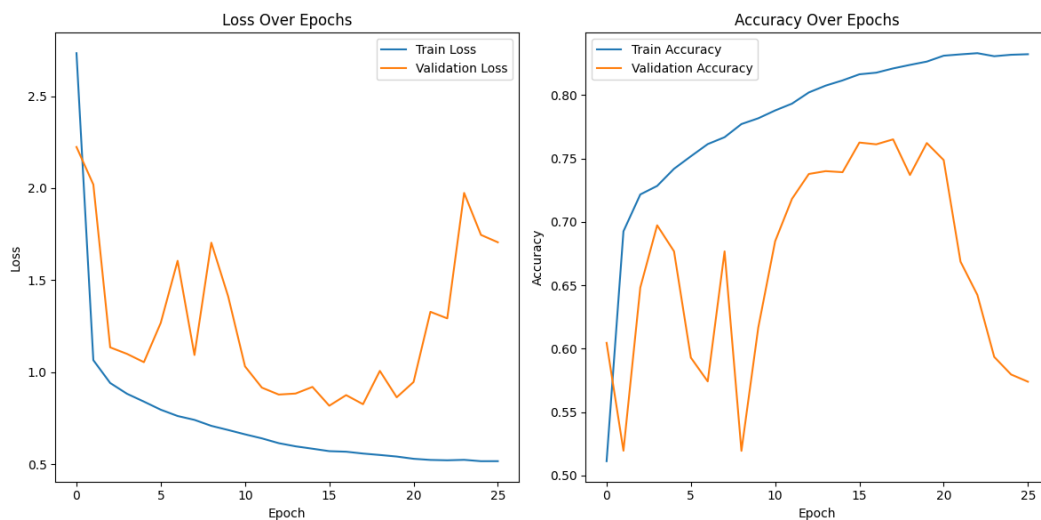


Figure 5.0.6: Training vs Validation Loss and Accuracy for 25 epochs

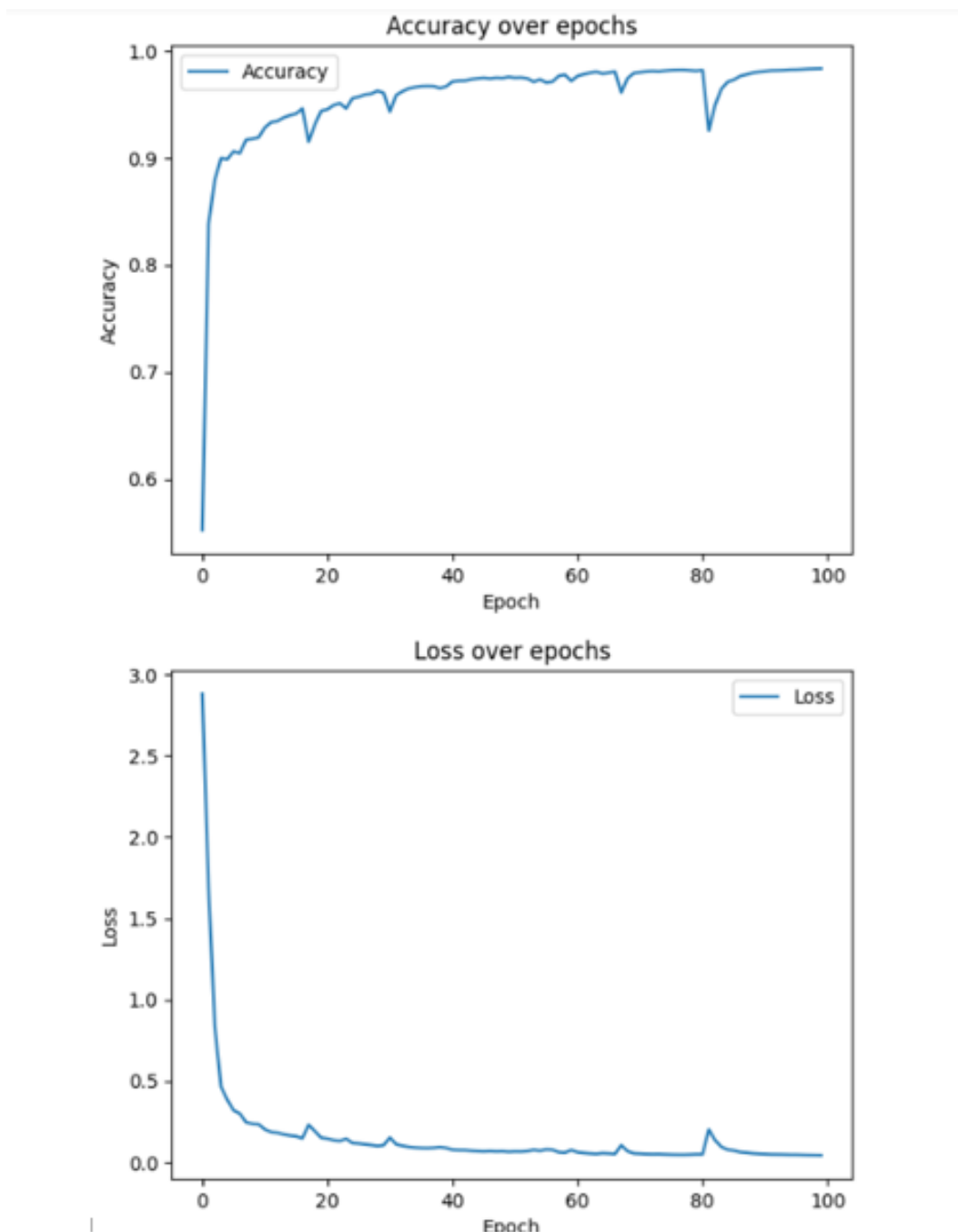


Figure 5.0.6: Training Accuracy and Loss for 100 epochs

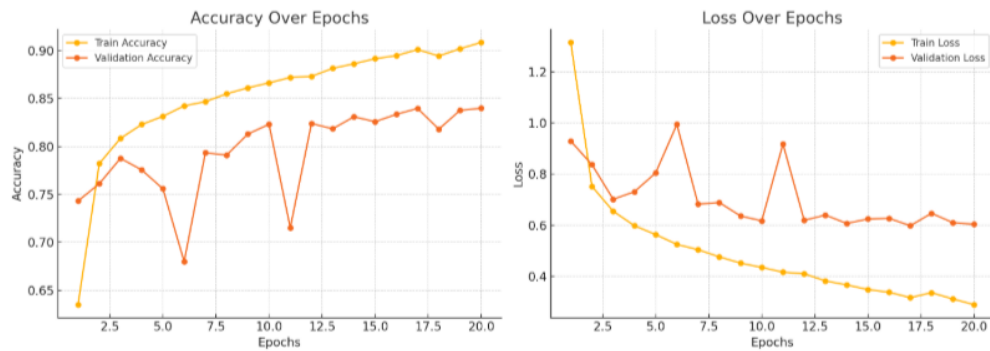


Figure 5.0.6: Training Accuracy and Loss for 20 epochs for SegNet

## Qualitative Analysis

Figure 5.0.6 presents visual comparisons between the input test images (not rendered properly in this figure), ground truth masks, and the predicted masks generated by the model. The model captures the overall semantic structure quite well, though some fine-grained details differ. The predicted masks are generally close to the ground truth, especially for dominant classes such as road, building, and sky. However, minor misclassifications can be observed for small or thin objects, indicating areas where model performance can be further improved.

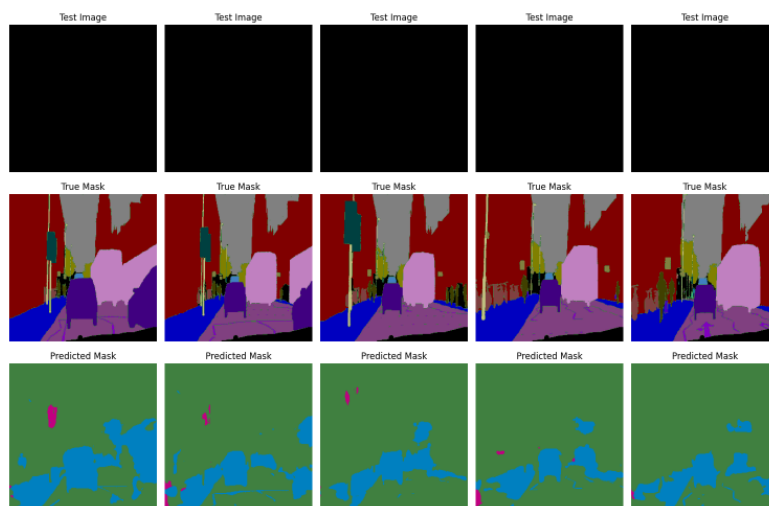


Figure 5.0.6: Segmentation results: Input image, Ground truth mask, Predicted mask

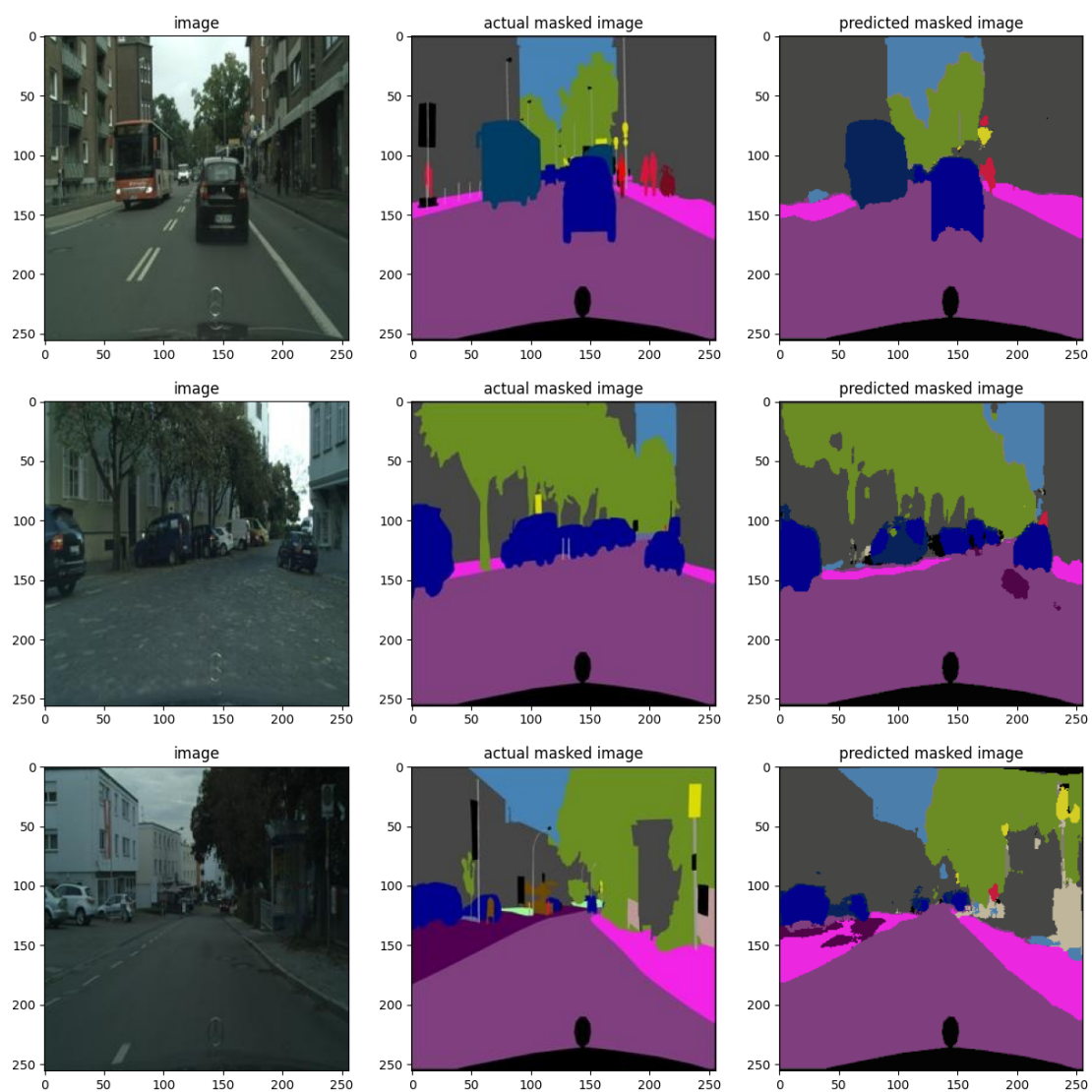


Figure 5.0.6: Segmentation results for SegNet: Input image, Ground truth mask, Predicted mask

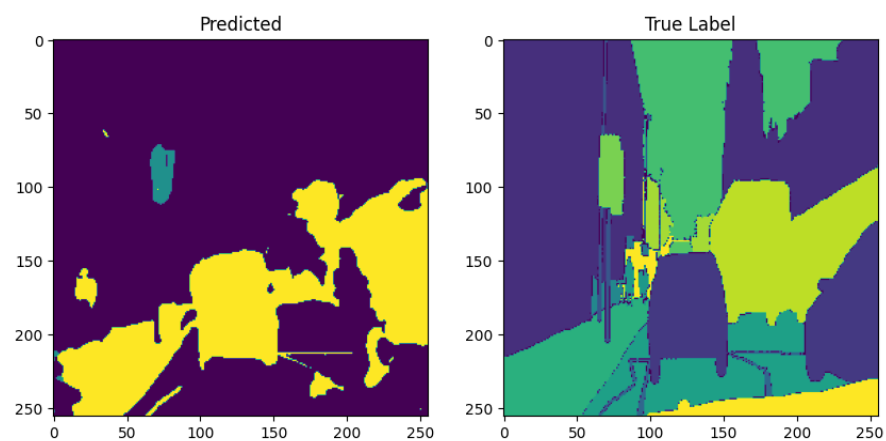


Figure 5.0.6: Predicted vs true label

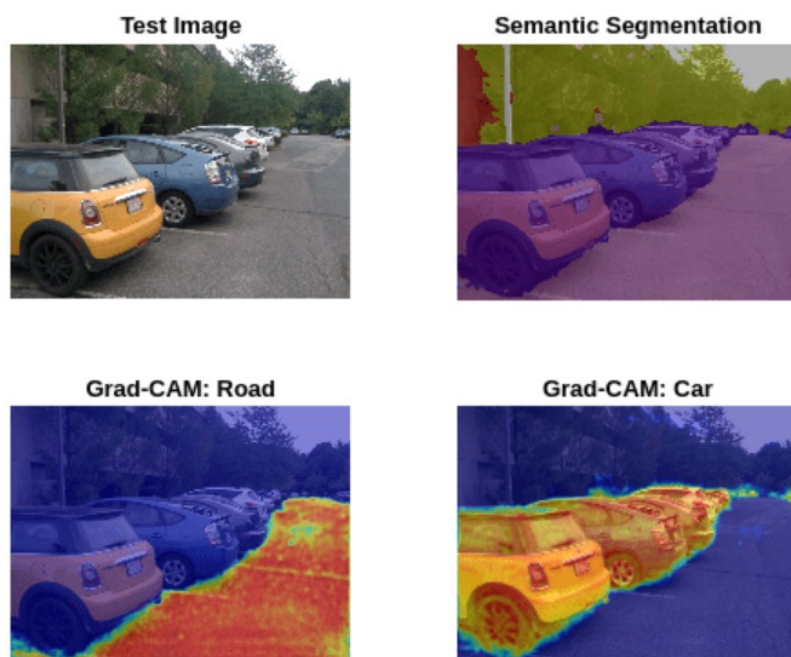


Figure 5.0.6: Grad Cam



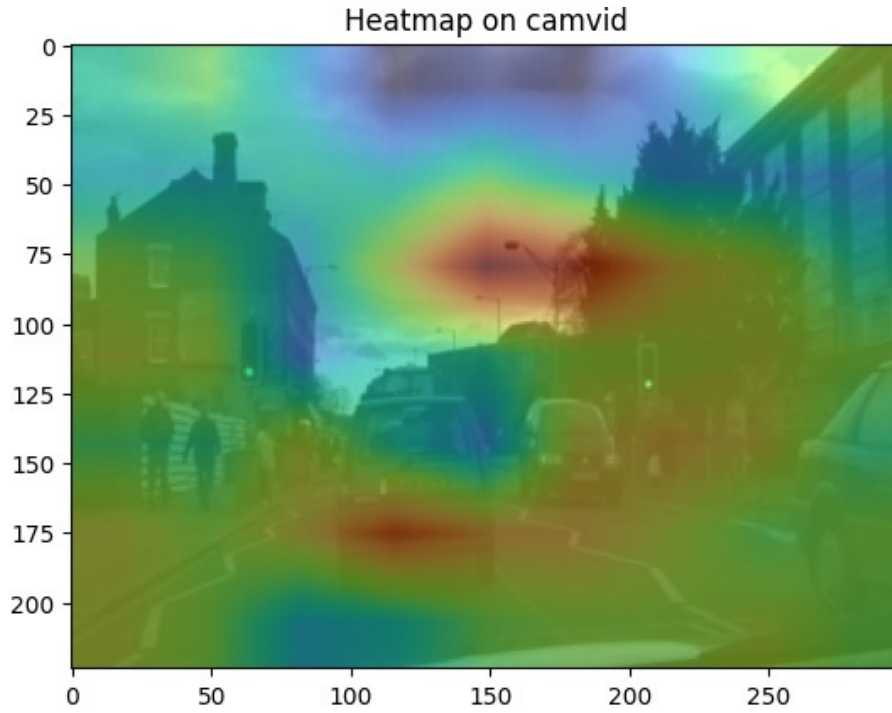


Figure 5.0.6: Heat map

### 5.0.7 Discussion

While both architectures are suitable for semantic segmentation tasks, U-Net's performance superiority lies in its ability to recover spatial context via skip connections. SegNet, although efficient in memory usage due to its encoder-decoder design with pooling indices, falls slightly short in capturing finer details. Moreover, U-Net converged faster and exhibited better generalization on the validation data.

In conclusion, U-Net is more effective for urban scene understanding tasks, making it a better choice for autonomous vehicle applications where boundary precision and class differentiation are crucial.

# Chapter 6

## Summary and Conclusion

### 6.1 Summary

This project explored and compared two deep learning-based semantic segmentation models—U-Net and SegNet—in the context of autonomous driving. Both models were implemented, trained, and evaluated on structured urban scene datasets using pixel-wise annotated images.

- The **U-Net model** was trained using the CamVid dataset, which includes 701 high-resolution images labeled with 32 semantic classes. Images and labels were resized to  $256 \times 256$  resolution and encoded into one-hot format. The model featured an encoder-decoder architecture with skip connections that helped preserve fine spatial details during upsampling.
- The **SegNet model** was implemented on the Cityscapes dataset using a side-by-side image format (original image + label). The segmentation masks were preprocessed using KMeans clustering to assign pixel-level class labels across 13 classes. A custom data generator was used to feed batches of training and validation data dynamically.
- For U-Net, training was conducted using a batch size of 4 for 100 epochs, with checkpointing enabled to store the best-performing model. The model used categorical cross-entropy loss and the Adam optimizer.
- SegNet was trained for 20 epochs with a batch size of 10 using a similar optimization and loss setup. Predictions were converted back to RGB using

KMeans cluster centers and visualized alongside ground truth masks.

- Both qualitative (visual overlays of predictions versus ground truth) and quantitative (accuracy, mIoU) evaluations were used. Grad-CAM and saliency maps were used to further improve U-Net by illuminating the ways in which specific pixel regions affected predictions, facilitating debugging and transparency.
- To maintain trained models and verify predictions after training, model checkpoints, loading, and testing features were added.

## 6.2 Conclusion

For autonomous cars, semantic segmentation is essential because it allows for accurate pixel-by-pixel scene understanding. This project effectively applied two popular architectures for semantic segmentation on urban driving datasets: SegNet and U-Net.

- U-Net’s skip connections successfully preserved minute details like lane markings, curbs, and pedestrians, demonstrating its exceptional accuracy. It works well in situations where interpretability and high boundary accuracy are crucial.
- Because SegNet uses pooling indices for upsampling, it demonstrated faster performance and memory efficiency than U-Net, despite having slightly lower visual quality accuracy. For real-time embedded systems with constrained computational resources, it is a solid contender.
- U-Net was integrated with explainability tools like Grad-CAM and saliency maps, which made it easier to see which input regions had the biggest influence on the model’s choices. In safety-critical systems like autonomous cars, where identifying failure points is essential, this is extremely helpful.
- The various data preparation techniques—KMeans clustering for Cityscapes and manual label encoding for CamVid—also illustrated how crucial dataset properties are in determining preprocessing pipelines.
- Both models were built using TensorFlow/Keras and trained using GPU acceleration, showing the practical feasibility of training segmentation models on mid-size datasets in academic or prototyping contexts.

# Bibliography

- [1] M. Treml, J. Medina, T. Unterthiner et al, “Speeding up semantic segmentation for autonomous driving,” in MLITS, NIPS 10 Workshop. *CVPR*.
- [2] A. Sagar, and R. Soundrapandiyan, “Semantic segmentation with multi-scale spatial attention for self-driving cars,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, pp.2650-2656 *MICCAI*.
- [3] M. Siam, M. Gamal, M. Abdel-Razek et al, “A comparative study of real-time semantic segmentation for autonomous driving,” in Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp. 587-597. *TPAMI*.
- [4] S.L.N.E.O.Y.G Nresh,”a residual encoder-decoder network for semantic segmentation in autonomous driving scenarios,”in european signal processing conference *arXiv*.
- [5] C.G.J.Y Bike chen,”importance -aware semantic segmentaion for autonomous vehicle.” in IEEE Transaction on intelligent transportation systems. *ECCV*.
- [6] Hrag-Harout Jebamikyous,Rasha Kashef, ”Deep Learning-Based Semantic Segmentation in Autonomous Driving”,in 2021 IEEE 23rd Int. Conf. on High Performance Comp*arXiv*.
- [7] Vijay Badrinarayanan, Alex Kendall, Roberto Cipolla, ”SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation,” in IEEE Transactions on Pattern Analysis and Machine Intelligence *TPAMI*.