School of Computing
FACULTY OF ENGINEERING

UNIVERSITY OF LEEDS

Object Segmentation in Autonomous Cars

Neil Abraham

Submitted in accordance with the requirements for the degree of
MSc Advanced Computer Science

2022/2023

The candidate confirms that the following have been submitted*:*

Items	**Format**	**Recipient(s) and Date**
Deliverables 1	*Report*	*SSO (26/08/23)*
Deliverable 2	*Software URL (Github URL)*	*Supervisor, assessor (26/08/23)*

Type of Project: Empirical Investigation

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student): Neil Abraham (26/08/23)

Summary

The project focuses on the implementation of semantic segmentation techniques to enhance the capabilities of autonomous vehicles. The aim is to accurately label and classify pixels within urban scenes, enabling vehicles to understand and navigate complex environments. The project employs a subset of the Cityscapes dataset, which provides high-resolution images of urban landscapes, serving as a real-world representation of the challenges faced by autonomous vehicles.

Problem Statement: Autonomous vehicles require a deep understanding of their surroundings to make safe and informed decisions. Semantic segmentation is crucial for identifying and differentiating various objects, such as roads, pedestrians, vehicles, and traffic signs, within an image. Accurate pixel-level labelling is essential for these vehicles to interpret and respond appropriately to the environment.

Approach: The project employs semantic segmentation models to predict class labels for each pixel in the input images. Two main architectures, UNet and R2UNet, are utilized to capture spatial information and make accurate predictions. UNet, a popular architecture for image segmentation, is compared with R2UNet, an enhanced version with recurrent residual blocks. The models are trained, validated, and tested using a meticulously curated subset of the Cityscapes dataset, comprising urban scenes from different perspectives and lighting conditions.

Data Preparation: The Cityscapes dataset, originally containing images with both the original view and ground truth labels, is divided into two distinct subsets. Images are split into training, validation, and testing sets, ensuring an equitable representation of the dataset's diversity across these subsets. The original images and their corresponding ground truth labels are separated to facilitate pixel-level semantic segmentation.

Implementation: The semantic segmentation models are trained on the training subset using an appropriate loss function and optimization techniques. During training, the models aim to minimize the loss by iteratively adjusting their parameters. The validation subset is used to monitor the model's performance and prevent overfitting. A saturation loss mechanism is introduced to halt training when the validation loss starts to increase, avoiding excessive training.

Evaluation: The models are evaluated on the testing subset using various metrics such as pixel accuracy, precision, recall, F1 score, and IoU. These metrics provide insights into the models' abilities to accurately classify pixels and differentiate objects in complex urban scenes. Both UNet and R2UNet are compared based on their performance across different learning rates.

Acknowledgements

I would like to express my deepest gratitude to the School of Computing for providing me with the invaluable resources and environment to pursue my academic endeavors. The support and guidance I have received throughout my journey have been instrumental in shaping both my personal and professional growth.

I am particularly indebted to my supervisor, Professor Sharib Ali, whose unwavering dedication, insightful guidance, and constructive feedback have been the cornerstones of this project. Your expertise and mentorship have not only enriched the quality of my work but have also inspired me to strive for excellence in all aspects of my academic pursuits.

I am also grateful to my assessor, Professor Toni Lassila, for their valuable insights and thoughtful evaluation of my work. Your constructive criticism and encouragement have pushed me to refine my ideas and present them in the best possible light.

In addition to the academic support, I would like to extend my heartfelt appreciation to my family. Your unwavering belief in me and your constant encouragement provided the emotional foundation I needed to navigate through the challenges of this project. I am especially grateful to my sister, whose steadfast support and understanding helped me endure the tough times and celebrate the successes.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

Image segmentation is a form of analysis that processes an image into its various components/ segments by partitioning it. This allows the complexity of an image to be broken down, remove the background noise and identify the important segments through proper labelling and classification. Simplification of an image through this segmentation mechanism helps identify the key components in the image and contributes to enhancing the overall reliability of image recognition and add the same into the existing data set with new information which further improves readability and analysis of images. With an improvement in data set and constant analysis, the need to read the entire image also reduces, improving the processing capability and deriving faster and better results.
The image segmentation model has long been in use to solve a plethora of use cases in the real world particularly in the field of robotics, automobiles, security, medicine and food & beverage industry. Some key examples of image segmentation and recognition:

1) In the field of medicine, one of the most important use cases is in the ability to detect tumor cells from regular healthy cells in CT/ MRI scans, wherein the reported time for segmentation was between 15s-2mins. (Yang, C. et al., 2022) This is key in the early detection and diagnosis of cancer. This advancement supports medical diagnosis and bio-medical & radiotherapy applications.
2) For the automobile industry, an important application of image segmentation is for the usage of the right techniques and data model to identify objects in the path of autonomous vehicles and locomotives. A more important problem to solve is to also improve the ability to detect objects in low light scenarios and improve the overall reliability and safety through accurate recognition. (Devipriya et al., 2023)
3) Another interesting use case is in image processing in the food industry, where automated factories can use this technique for detection of rotten vs good produce. (Chopra et al., 2022)

The above 3 examples are just few of the real-world use cases, there are innumerable and innovative ways where image segmentation has been implemented.
This project will focus on image segmentation and recognition for autonomous vehicles by exploring the concepts of UNET led semantic segmentation.

1.1 Problem Statement

As we move into a world where self-driven vehicles will become more commonplace, the need of the hour is to ensure that the systems in use have the ability to process the data in real time with low margins of error and therefore have the right data models that are constantly under improvement to ensure a safe and smooth driving experience, minimising the risk of road and in general vehicular accidents (extended and not limited to automobiles and locomotives).
The current detection techniques rely on the available set of data and information, however, with the ever-changing landscapes and accordingly the presence of anomalous background information, accurate detection of the objects within the image becomes extremely crucial. Due to the mentioned challenges along with conditions such as weather, lighting, unavailable/ new data, inaccuracy in data labelling, there are high chances of inconsistencies in accurately identifying the object in an image captured in an autonomous vehicle. Hence the ability to segment the image correctly into distinct regions, where each region is an object for the vehicle within its view becomes critical to the overall success of the image segmentation model. A robust model will provide better ability for the vehicle to distinguish between objects like humans, animals, trees, road and various other objects in its view.
This capability & further enhancements within this capability will ensure that self- driving systems are more accurate, reliable, and safe to use. A crucial aspect of this problem statement is also to understand the existing data models and their respective methodology to review for potential improvements.

1.2 Project Aim

The aim of this project is to perform a comparative study of the existing Semantic Segmentation Models used for image segmentation and recognition for autonomous & self- driven vehicles and review the efficiency and accuracy of these models in identifying the image segments.
To achieve this, the prime focus is to design an experiment that is capable of processing an image captured in the field of view of an autonomous vehicle, segment the relevant portions and identify the objects accurately within the image. The comparison study would be performed between two semantic segmentation models: U-Net and R2U-Net Convolutional Neural Network (CNN).
For the purpose of the experiment and testing the data model, this project utilises the Cityscapes (Anon) dataset, which is a publicly available dataset with a focus on urban street scenes with a total image set of 2975 training images and 500 validation images. Considering the system's ability to process the training and validation data- a subset of Cityscapes has been used to conduct the experiment.

Figure 1: Representation of semantic segmentation in the field view of autonomous vehicle

1.3 Project Objective & Scope

The objectives of the project thesis are as follows:
1) To review the existing literature on semantic segmentation model for image segmentation for autonomous vehicles through CNN models, particularly UNet & R2-UNet and identify existing issues.
2) To familiarise with the Cityscapes dataset
3) To develop and implement the algorithm for image data processing and analysis.
4) Perform a comparison study between U-Net & R2 U-Net and determine accuracy levels through validation of results for various test cases.

1.4 Deliverables

Deliverable 1	Code implementation in 2 models: U-Net & R2-Unet CNN
Deliverable 2	Detailed report of the research, test cases & result

1.5 Data set: Addressing ethical, legal, and social concerns

Since the project involves a large data set of images sourced from a public dataset & is used to enable self-driving concepts, following concerns need to be noted:

1) Data privacy and transparency: Since the images used in the data set are from urban scene surroundings which includes identification of pedestrians and other vehicles in the image, ensuring data privacy and transparency on the data collection process is of high importance.
2) Safety and reliability: Since the primary focus is to utilise the ability of image segmentation for self-driving autonomous vehicles, accuracy & reliability of the output is paramount. Safety concerns and legal questions on the responsibility in case of an accident or similar incident due to a lower accuracy or failure of the model are likely.
3) Bias in identification & fairness: The training data used for the model incorporates multiple scenarios to avoid any possibility of potential bias and wrong identification of object or unfair treatment of certain objects or groups. Making sure the model's decision-making is fair and unbiased is vital to avoid ethical and legal challenges.
4) Algorithm transparency: Since the image segmentation model uses complex deep learning models such as neural networks, discerning how the model identifies and segments data is important. Ensuring segmentation & data transparency and the model's decision making is important to attain and maintain user trust and support for any legal requirements.

1.6 Project Design

To execute this project, the following aspects were taken into consideration, following the agile method of development:

1) Literature review
   - Research on image segmentation techniques, Deep learning, CNN and various methods available under CNN
   - Existing use cases of image segmentation was extensively reviewed to develop a deep knowledge on the subject matter and real-world usability was reviewed.
2) Define
   - Based on literature review and availability of authorised and valid data set, the scope of the project was finalised to concentrate the experiment on autonomous & self- driven vehicles, with the acknowledgement of expanding the usability of the experiment to more relevant scenarios.
3) Design and develop:
   - Experiment design which included and system and software setup.
   - Identification of the data set by scanning through multiple available resources, which match the research criteria.
   - Defining the evaluation criteria for the experiment.
   - Pre-processing the data set.
4) Evaluate:
   - Verification of the code, review of the results and the challenges during the implementation of the code.

1.7 Challenges & mitigation

1. Identifying the right data set for the project
2. Size of the data set: Processing & training time for large data set posed a significant challenge while running the code. Hence the initial sprint results were not very promising. As a mitigation plan, the size of the data was trimmed to support data processing and reduce run

time for the results. The key assumption here is that in actual industrial set up, these issues would be accounted for and hence support data processing and training of larger data sets.

3. Evaluation criteria: While the evaluation metrics were implemented, the results would be more significant in an industrial setup.

Chapter 2
Background Research

2.1 Literature Survey

In this chapter, the main concepts of Deep learning, and specifically Convolutional Neural Networks (CNN) have been covered. We start the Literature survey by reviewing the concepts in detail based on the available research materials, focusing on semantic segmentation through CNN- specifically UNet and R2Unet, since we will be delving further into these concepts for the purpose of this project. While UNet is more widely researched in various domains, through literature survey, it was understood that the applications of R2UNet which also in the same domains as UNet, were still limited, and were more readily available within the medical devices area. Through this project and literature search we recognise the capability of extending the model to also include the image segmentation for autonomous vehicles.
The aim of this section thereby is to provide a comprehensive view of the available research that facilitated the project design.

2.1.1 Deep learning

Deep Learning (a subfield of Machine Learning) has gained tremendous success over the past few years. It works on the concept of Artificial Neural Networks which uses the understanding of how neurons recognise and transmit information to the brain. As already discussed in the introduction, Deep learning has shown a significant improvement in various tasks such as image recognition, natural language processing, biomedical sciences and autonomous driving. (Sharma, M. et al., 2022)
An advantage of Deep Learning over Machine Learning is its ability to extract high-level features from the input which reduces the processing time as well as the dependency on feature engineering, which is no longer required. A Deep learning algorithm contains an input layer, multiple hidden layers, and an output layer.

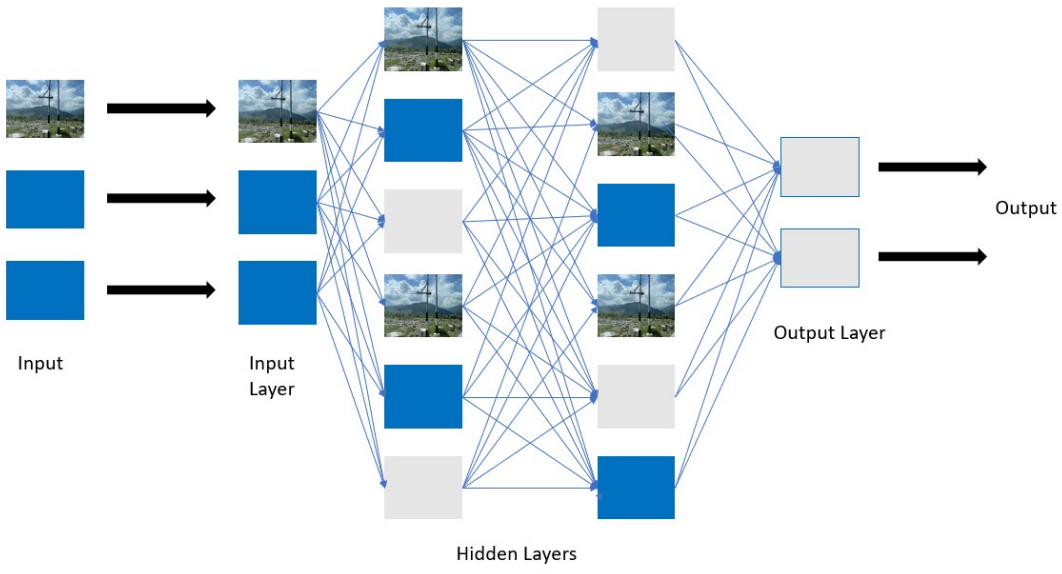

Figure 2. Illustration – Deep Learning algorithm layers

2.1.2 Convolutional Neural Networks

Convolutional Neural Networks are a type of deep learning model designed for analysing visual data such as images and videos. These models are used for tasks such as object detection, image classification, and image segmentation. CNN's have contributed to a remarkable advancement in the field of computer vision. The main reason for being a popular choice for image segmentation is because

of its ability to use 2D images and identify the features and patterns within the image by using many hidden layers and recognising the spatial arrangement of pixels in the input image. The main advantage with CNN is that it does not require much pre-processing.
A classic CNN model comprises of 4 components:

- **Convolution layer**
  - The convolutional layer uses the input data to detect patterns by using filters called kernels. The kernel slides over the input image and the dot product between the input image and the kernel is derived. (Alzubaidi, L. et al., 2021) This is known as the convolution operation (Laith Alzubaidi et al.).

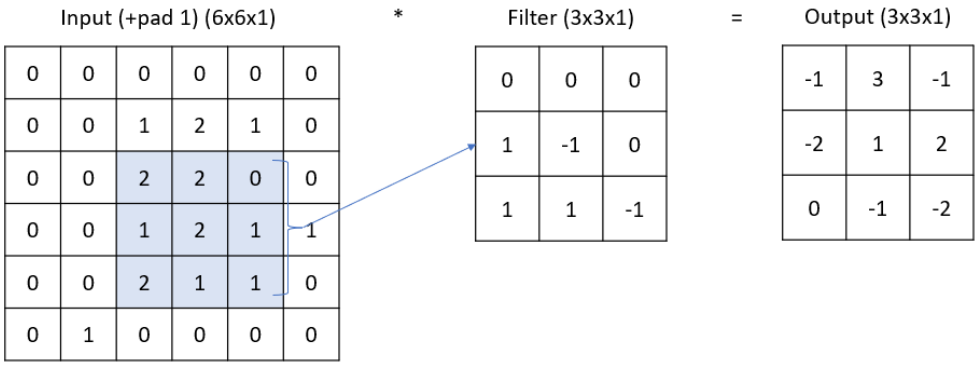

Figure 3. Convolution operation

- **Pooling layer**
  - The main task of the pooling layer is to take the important information from the convolution operation performed as step 1 and reduce the convolved feature maps into smaller sized feature maps. (Alzubaidi, L. et al., 2021)
  - There are many pooling methods present with max, min, and global average pooling being the commonly used in CNNs (Alzubaidi, L. et al., 2021).

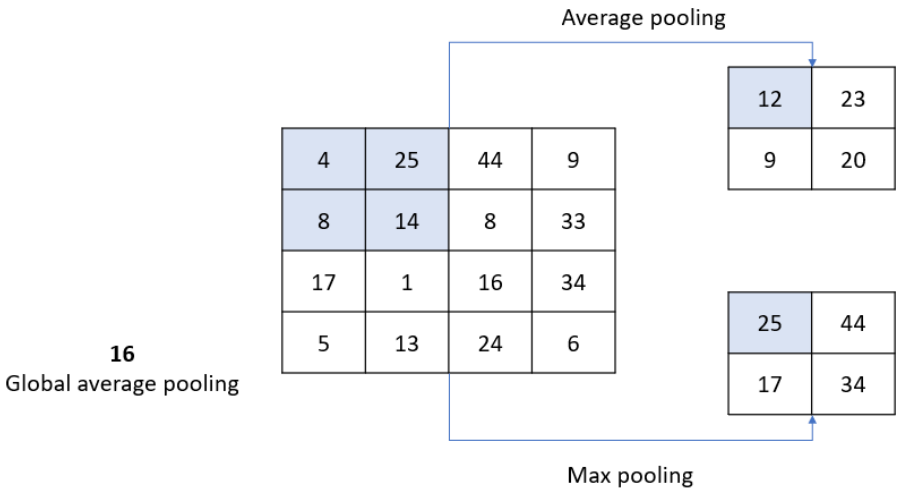

Figure 4. Pooling methods

- **Activation function**
  - Activation function has improved the non-linearity with the neural networks model and hence plays a key role.
  - The neural network works like a network of connected neurons. As the neuron tries to communicate with another neuron through the various layers, the activation function makes the decision if a neuron should "activate" and accordingly pass on the signal it has received or not, based on its input. Then, the activation function takes the input of each neuron and converts it into the neuron's output.
  - By using this function, the neural network is able to handle complex relationships & patterns in data likes curves, which otherwise have been able to understand only simple, straight relationship between input and output like a line. Hence this limitation is overcome with the activation function, the neural network gains the ability to learn more intricate relationships. This makes the network much more powerful and capable of solving a wide range of problems in machine learning.
  - The are many activation functions such as the Sigmoid function, Tanh function, ReLU, and the leaky ReLU function. The ReLU function is one of the most commonly activation functions used in CNN's (Alzubaidi, L. et al., 2021).
    - Sigmoid function:
      $f(x)_{sigm} = \frac{1}{1 + e^{-x}}$
    - Tanh function:
      $f(x)_{\tan h} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$
    - ReLU function:
      $f(x)_{ReLU} = \max(0, x)$
    - Leaky ReLU function:
      $f(x)_{LeakyReLU} = \begin{cases} x, & x > 0 \\ mx, & x \leq 0 \end{cases}$

- **Fully connected layer.**
  - The last component of the CNN architecture is the Fully connected layer.
  - The output of at this layer determines the final output of the CNN model.
  - All the neurons from the previous layers are connected to every neuron in this layer (Laith Alzubaidi et al.).

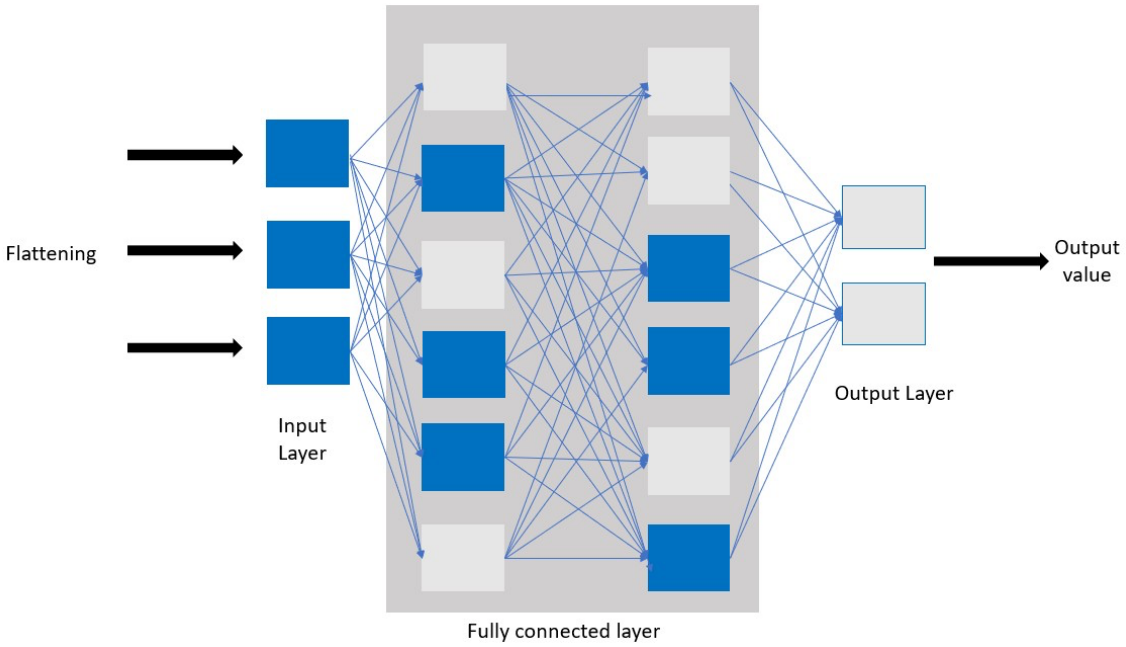

Figure 5. Illustrative image- Fully connected layer

2.1.3 Semantic Segmentation

Semantic segmentation is a computer vision technique that reviews an image at a deeper level than considering it as a simple image. In this technique, rather than tagging the image as a single class, the image is actually broken down to pixels, and each pixel in the image is labelled with a corresponding class. This pixel level classification breaks the image into proper regions and which allows further analysis of the image (Busra Emek Soylu et al.).

Semantic segmentation also comes with major real-world applicability like autonomous driving, medical imaging and for surveillance (Xiaolong Liu et al.). However, for these applications the task is challenging considering the requirement of precise pixel-level labelling, identifying overlapping objects, and taking into account the influence of lighting and viewpoints.

An important problem statement with semantic segmentation has been on the ability to design efficient feature representation to differentiate objects that are part of different classes. The use of Deep learning and with the advancement in Convolutional Neural Networks (CNNs), it has significantly improved semantic segmentation performance by learning the features in an image and hence capturing finer details in the image and context.

Different CNN architectures, such as Fully Convolutional Networks (FCNs), U-Net, DeepLab, and SegNet, can be used for semantic segmentation. Each architecture looks into the specific challenges and addresses those. To tackle the issue of limited labelled data, data augmentation techniques are often used to generate more training examples.

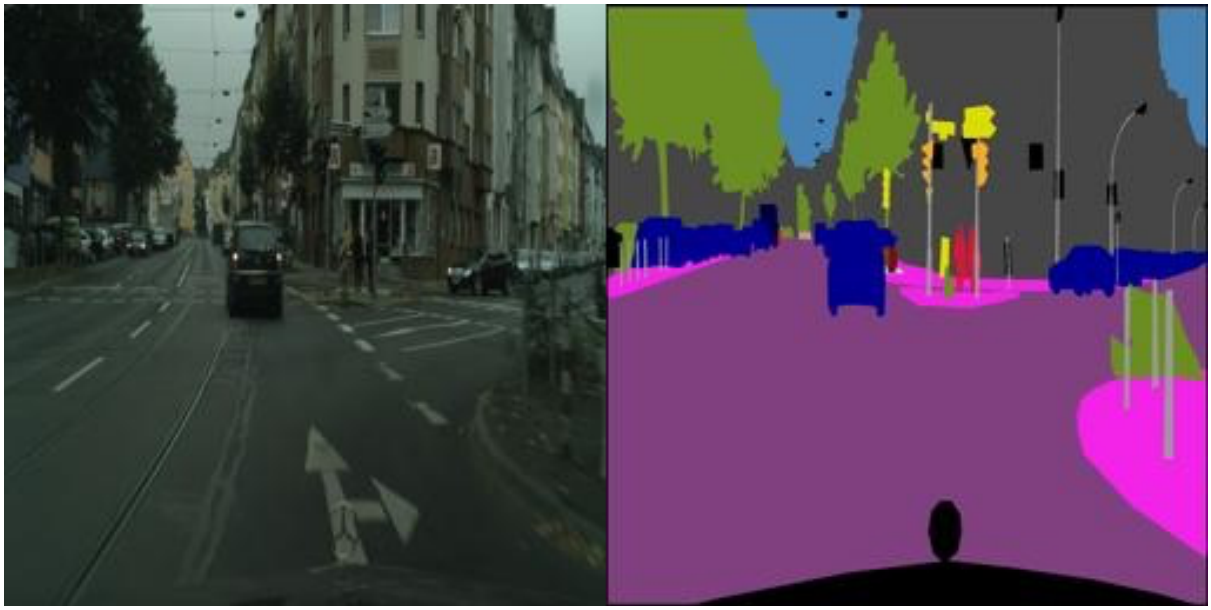

Figure 6. Illustrative representation of semantic segmentation

The complexity of the application is high for Semantic segmentation using Deep learning, and generally the number of images required at a pixel level are in thousands, which increases the complexity of the program but also improves the output of the model.

2.1.4 UNet

UNet is a widely used type of neural network architecture specifically designed for tasks involving semantic segmentation. It was first introduced in 2015, in the paper titled "U-Net: Convolutional Networks for Biomedical Image Segmentation," authored by Olaf Ronneberger, Philipp Fischer, and Thomas Brox.

UNet was developed to address image segmentation needs in the field of biomedical devices, with a primary focus on the image segmentation of cell nuclei in microscopic images. With time, its effectiveness, adaptability, and applicability to other scenarios has been widely recognised in various research domains and computer vision tasks (Alom, M.Z. et al., 2021). Today, UNet's application is extended to diverse fields like natural scene images, satellite imagery, and even autonomous driving applications and according to the application, many versions of this model are constantly proposed.

UNet becomes even more useful when there is limited training data and can create detailed segmentations, which reduces the requirement of manual intervention and impact improperly classified data labels. Hence UNet is usually a favoured choice for semantic segmentation tasks.

The key features of UNet are as follows:

- **U-shaped architecture**
  - UNet's name is derived from the fact that its architecture is U-shaped.
  - UNet has 2 components:
    - Encoding path: this is where the input image is down-sampled to gather contextual information.
    - Decoding path: here the spatial resolution is increased, gradually, to generate the segmentation map.
- **Contracting and Expanding path**
  - The main purpose for the contracting path is to capture the multi-scale features and abstract representations from the input image.
  - This is done through repeated additions of convolutional layers and max pooling operations on the encoding path which follows a contracting pattern.

- Expanding pattern process is followed for the decoding path, where the image is up-sampled by using up-convolutional layer (transpose convolution/ deconvolution). This helps with the feature maps and allows recovery of spatial information.

- **Skip connections**
  - Skip connections are used to link the encoding and decoding layers.
  - This helps in enhancing the finer details and low-level features from the encoding path to the decoding path.
  - The network can then combine the information of the local level to the global level, providing better and detailed segmentation.

- **Multi-resolution fusion**
  - Fusion step in UNet involves linking of the feature maps created during the up-sampling process in the encoding path with corresponding up-sampled feature maps from the decoding path.
  - This combines the high-level semantic information with fine-grained spatial details and improves segmentation accuracy.

- **Loss function**
  - Dice loss or Cross Entropy loss (pixel wise loss function) are used for training UNet.
  - The loss function compares the predicted segmentation with ground-truth label and helps the network learn accurate pixel level predictions.

Below is a diagrammatic representation of a UNet model. UNet ensures that the entire image is processed in the forward pass and produces the segmentation map. This preserves the entire input image and becomes a major advantage of this model. (Alom, M.Z. et al., 2021)

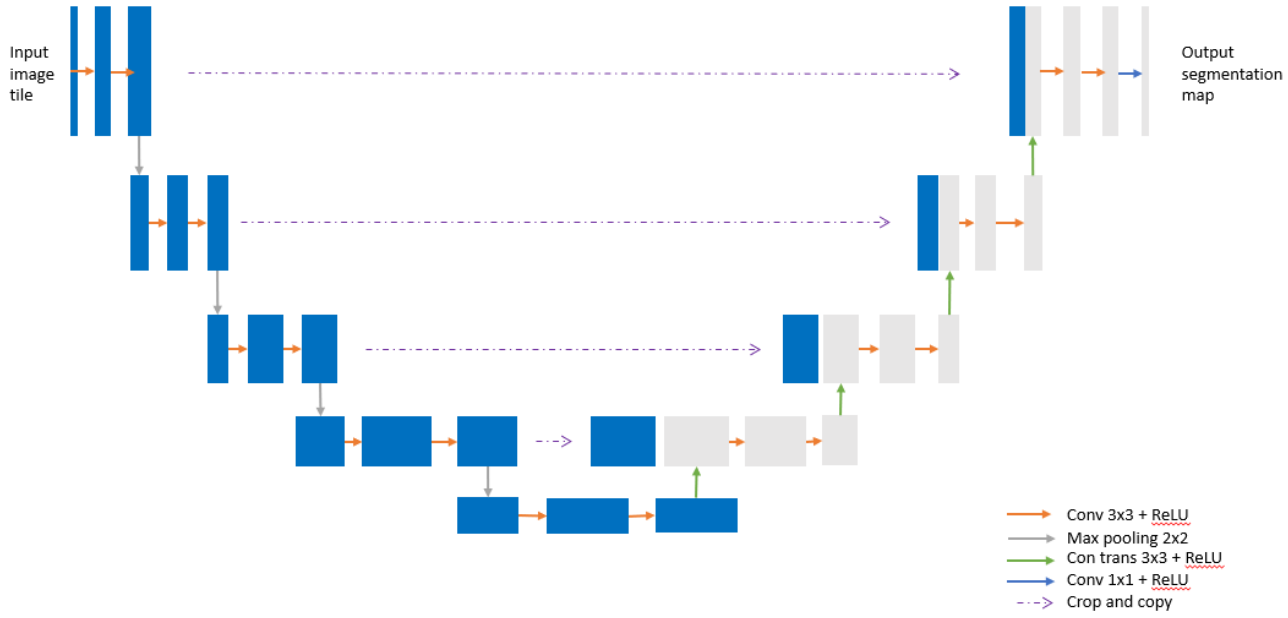

Figure 7. Illustration created to represent the UNet model

2.1.5 R2UNet

R2UNet- Recurrent Residual Convolutional Neural Network (RRCNN) is a deep learning model based on UNet. Its applicability runs in the same domains as that of UNet, which gained its popularity with medical imaging.

Within R2UNet framework same number of network parameters helps better image segmentation, thus the performance expectation is generally superior to that of UNet. The three main aspects that are used in R2UNet:[10]

- **UNet**
  - As discussed in the above section, UNet uses skip connections in the encoder and decoder layers to extract and fuse the relevant image segments.

- **Residual Network (ResNet):**
  - This helps with better training of deep architecture by skipping some layers that can impact the architecture performance through batch normalisation and account for any gradient level issues.
  - Residual Network works by connecting the activation of a layer with more layers by skipping some layers in between, hence this technique is called skip connections.
  - By doing this a residual block is created, stacking residual blocks together forms ResNets
  - This allows the network to identify the residual mapping and not work to understand the underlying mapping.
  - Function:
    - $F(x) = H(x) - x$, where $H(x)$ would have been the underlying mapping. Therefore $H(x) = F(x) + x$

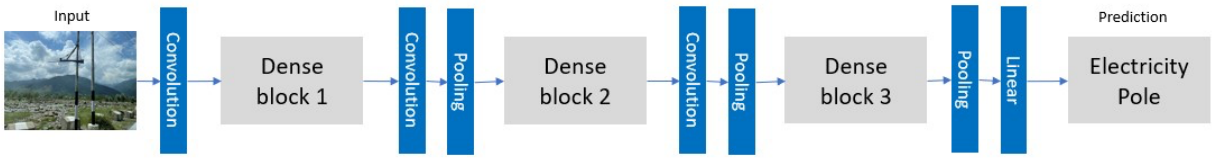

Figure 8. Illustration – Residual Network

- **RCNN (Recurrent Convolutional Neural Network):**
  - RCNN feature accumulations helps with better presentation for segmentation task. It works through a selective search to identify which are the regions of interest in an image and then extracts features from that image that can be used to classify the features and identify the image
  - Selective search looks for patterns in terms of scale, texture and color and identifies regions in the image. At the base level multiple regions are identified and then these regions are combined to form larger regions based on color, size etc. Using this a region of interest is identified.
  - RCNN has its limitations in that it is expensive to train the model and also slower since it identifies 2000 regions for each image for the selective search.

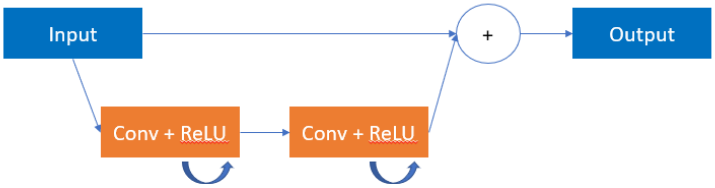

Figure 9. Illustration representation of R2UNet model though the additional of a Residual Network

2.1.6 Deep Convolutional Neural Network (DCNN)

With the understanding that deep neural networks have a shortcoming when trying to analyse high resolution remote sensing image segmentation (which is used for natural/ geological disaster warnings, agriculture, military etc), especially when these networks are trained only in natural scenes there has been research done to further the UNet approach by using DenseNet, UNet, DeconvNet and dilated convolution. This model is the Deep Convolutional Neural Network.
The improvement in this approach over UNet was a better accuracy in the evaluation indexes, increased segmentation speed. This was achieved by using combined dilated convolutions for feature extraction, reducing the number of layers.
This method allows the prediction of pixels which allows the model to develop a fine-grained reasoning for computer vision tasks.
Key components of this framework:

- **Dilated convolution:**
  - This component in the DCNN framework allows the take in a greater amount of the reception field to get enough amount of the global information without losing any of the image resolution.
  - Formula, here rfi = receptive field, ki= kernel size, di,= dilation factor and si = stride of dilated convolution in the i layer

$$rf_i = (rf_i - 1) + (k_i + 1) \times d_i \times \prod_{n=1}^{i-1} s_n$$

$$rf_i = t_i - i + 1 + t_2 \sum_{n=1}^{i-1} (ks)^n$$

$$g_i^{-1} = \frac{t_i - i + 2}{t_2 d_1 (bs)^{i-1}} + \frac{ks((ks)^{i-1} - 1)}{d_1 (bs)^{i-1} (ks - 1)}$$

  - These formulas help conclude that as the network layer increases, receptive field also increases without increasing the number of calculations required.

- **Dense connected convolutional network (DenseNet):**
  - With the use of DenseNet, the issue of vanishing gradient is resolved, and gradient propagation is improved, thereby also reducing the need to learn feature maps that do not contribute to the image (reduces redundancy). Using this component, (the dense block with holes) primary features of the image can be extracted.
  - Here there is a merging of the input of block with output of convolutions.

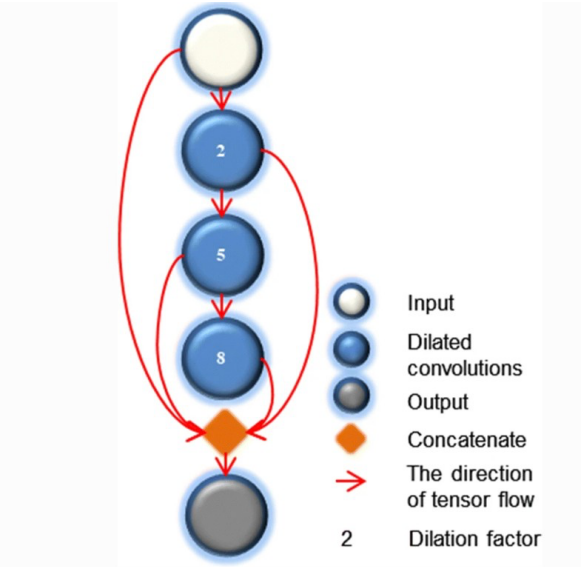

Figure 10. Dense block using dilated convolution

- **U-Net:**
  - As discussed in the above section, UNet uses skip connections in the encoder and decoder layers to extract and fuse the relevant image segments.

- **Transposed convolution:**
  - This component of DCNN helps with recovering the size of the feature maps and pixel location information.
  - The presence of 2 transition layers helps in the below scenario to reduce the size of the feature map by ¼ th of the original. Post feature extraction using UNet, reducing the vanishing gradient using Conv-conv mode, the 2 transposed convolution layers can bring back the feature map size to the input image size.

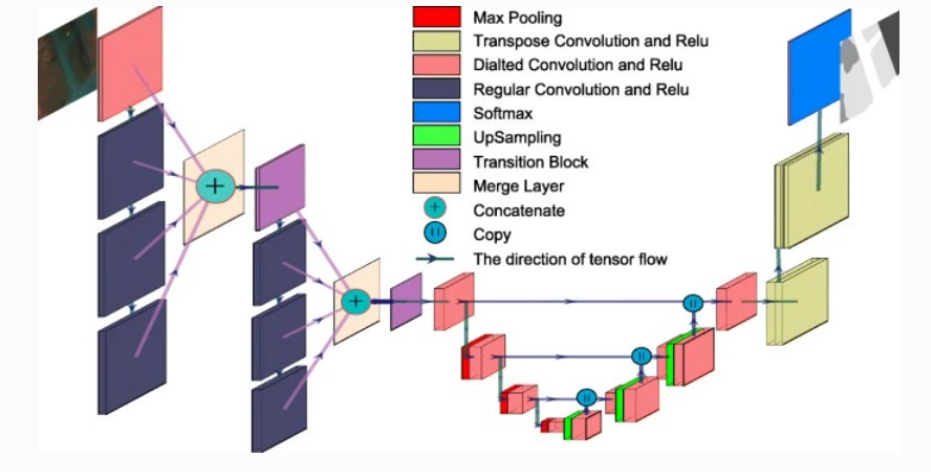

Figure 11. DenseUDeconvNet

There is continuous ongoing research on UNet and many models have been developed from different real world requirements that have come in, these following methods have not been convered in the background research but can provide sufficient inspiration to be used in combination with the methods

described above for better application and results of a hybrid model: H-DenseNet, MutiResUnet, ResUnet (partially covered), UNet++, Attention UNet.

2.2 Methods and techniques

The following are a few methods can be utilised to implement semantic segmentation in autonomous vehicles using Deep Learning based models. These methods can be used on their own, or in combination to leverage the unique capabilities in each method.

- **Efficient CNN based methods**
  - Deep learning has revolutionized semantic segmentation in autonomous vehicles. Convolutional Neural Networks (CNNs), particularly Fully Convolutional Networks (FCNs), are the choice for pixel-wise segmentation.
  - These methods leverage the hierarchical features learned by deep networks to accurately segment objects in real-time. The accuracy of image classification is improved through pre-trained models, combining many layers.
  - Transpose convolution methods help with up sampling and Dilated convolution help with spatial preservation.
- **Encoder-Decoder Architectures**
  - Encoder-decoder architectures, like UNet, relies on the down sampling and up sampling through an encoding path to capture context and a decoding path to recover spatial information. These models excel at maintaining fine details in the segmented output while capturing global context from the input image to create a full resolution segmentation output.
- **Transfer learning**
  - Transfer learning involves using pre-trained models on large-scale datasets, such as ImageNet, and fine-tuning them for semantic segmentation in autonomous driving scenarios. This approach benefits from the learned features in the initial layers of the network.
- **RCNN based network – using Residual connections**
  - The main advantage with this method is target detection in instance segmentation.
  - Traditional UNet already takes into account skip connections. Additional residual blocks create smaller skip connections inside the blocks. This allows the models training to complete faster. Residual connections are used in R2UNet.
  - A mask R-CNN network has been proposed by He et al. for instance segmentation network that has shown good results in many computer vision challenges. This model uses a faster R-CNN approach and can complete high quality segmentation through regression branching. A major advantage here is the ability of distinguishing 2 individuals within the same class.
- **Feature Pyramid Network**
  - This method which is a famous multi scale neural network was proposed by Lin et al. and is used for target detection & image segmentation. This model proposes building of feature pyramids.
  - Key components of this method:
  - Bottom-up pathway
  - Top-down pathway
  - Lateral connections- for combining high- and low-resolution information in small target objects within remote sensing capabilities. Usually a lot of feature information gets lost during down-sampling tasks, which causes an error in the small target object detection. Therefore as proposed by Liu et al. using pyramidal supervision along with multiscale architecture and auxiliary loss detection, spatial context information can be maintained.

- **Attention based mechanism network**
    - Chen et al. proposed a multiscale feature extraction technique that allows the model to assign weights to each pixel.
    - The attention-based mechanism network uses average and maximum pooling and allows the model to assess and place weights on each item in various placements.
    - The main advantage for this method is the capability for feature representation and spatial localisation precision.

- **Other methods:**
    - GAN (Generative adversarial network) Based Network: this a new technique that uses generator & discriminator. This method uses 2 neural networks against each other to produce synthetic data and can be used for creation of videos and images.
    - L-GCNN: Learnable gated network, suggested by Guo et al. for improved segmentation.

While semantic segmentation through Deep learning tends to be a favourite choice and provides excellent results, a major pain point which impacts results is the proper merging of global context with fine grained spatial data. Large variations in the data set also makes semantic segmentation a more challenging task.

2.3 Choice of methods and techniques

For the purposes of this research and after reviewing the available literature on the various methods and techniques available for CNN models for image segmentation, the choice of conducting this empirical investigation was narrowed down to using the following techniques:
CNN based Encoder Decoder architecture:
Through the literature study, it is understood that CNN based models are widely popular choice for semantic segmentation and would be the perfect choice to understand the real-world results of this method for image segmentation of autonomous vehicles.

1) Model 1: UNet
It is by far the most popular and widely researched CNN model especially considering the autonomous vehicle domain and well established as a choice for semantic segmentation and can provide a good comparator, being a traditional model.

UNet's flexibility to be combined with other techniques to produce hybrid models also makes this an important choice for the project experiment. Hence, studying this technique would be an important from a learning perspective.

Apart from autonomous vehicles, the research in other domains such as medical devices, remote sensing etc is also widely available and hence makes this model a fit candidate.

2) Model 2: R2UNet
R2UNet is an updated version of UNet and while it is researched well in the field of medical devices, the autonomous vehicle segment remains largely less researched. R2UNet has proved to be an efficient model for remote sensing as well. Hence in order to test the applicability of R2UNet in autonomous vehicle segment, as well as comparison with the results of UNet to see if there are advantages vs a traditional model, R2UNet posed as the right model for the experiment's purpose.

Chapter 3
Datasets and Experimental Design

3.1 Datasets

3.1.1 Dataset Description

3.1.1.1 Cityscapes Dataset

The Cityscapes dataset is a comprehensive and widely used dataset designed for urban scene understanding and computer vision research. It focuses on semantic segmentation, instance segmentation, and various other tasks related to urban environments. This dataset has gained significant popularity in the field of deep learning and computer vision due to its realism, diversity, and complexity.

Dataset Composition:

The Cityscapes dataset contains an extensive collection of high-resolution images captured in various cities across different countries. It includes images of street scenes, captured from the perspective of a vehicle's driving view. The dataset is composed of a staggering 25,000 frames, making it an order of magnitude larger than previous attempts.

The dataset includes:

1. 5,000 frames with high-quality pixel-level annotations for tasks like semantic segmentation and instance segmentation.
2. 20,000 frames with weak annotations, which provide additional data for various research purposes.

Annotation Information:

The Cityscapes dataset stands out due to its high-quality pixel-level annotations. Each image is densely annotated, with every pixel labeled with a specific semantic class, providing a detailed understanding of urban scenes. In addition to pixel-level annotations, the dataset also provides annotations for instance-level segmentation, enabling differentiation between individual instances of the same object class.

Object Classes:

The dataset defines 19 object classes, encompassing a wide range of urban objects and structures. These classes include vehicles, pedestrians, roads, buildings, and more. The diversity of object classes makes the Cityscapes dataset well-suited for a variety of tasks such as semantic segmentation, object detection, and instance segmentation.

Usage and Benchmarks:

The Cityscapes dataset has been a cornerstone of various computer vision tasks, including semantic segmentation, instance segmentation, object detection, and depth estimation. Many cutting-edge algorithms and models have been developed, trained, and evaluated on this dataset, cementing its status as a benchmark for urban scene understanding tasks. It has led to substantial advancements in fields such as autonomous driving, robotics, and urban planning.

3.1.1.2 Cityscapes Image Pairs

For the comparison of two models for the segmentation of autonomous vehicles, we are utilizing the "cityscapes-image-pairs" dataset, which is a subset of the well-known Cityscapes dataset. This subset is specifically designed for semantic segmentation tasks. The "cityscapes-image-pairs" dataset contains 2975 training image files and 500 validation image files.

Each image file in the dataset has dimensions of 256x512 pixels. Notably, every image file is a composite, where the left half of the image corresponds to the original photo captured from the real-world driving environment. In contrast, the right half of the image represents the labelled image, obtained as the output of semantic segmentation.

By utilizing this composite setup, we can efficiently evaluate the performance of our segmentation models. The original photo provides the real-world context, while the labelled image serves as the ground truth for comparison against the segmentation results generated by the models.
The "cityscapes-image-pairs" dataset facilitates a comprehensive evaluation of the segmentation models' accuracy and generalization capabilities in autonomous driving scenarios, enabling us to make informed decisions in selecting the most suitable model for enhancing the perception capabilities of autonomous vehicles.

Despite its success, the Cityscapes dataset continues to present challenges like occlusions, varying lighting conditions, and intricate urban scenes. As part of its future directions, the dataset's creators are actively enhancing annotations, exploring new tasks such as panoptic segmentation, and considering the inclusion of an even more diverse array of cities and scenarios.

Figure 12. Illustration of the Cityscapes Image Pairs Dataset on Kaggle

3.1.2 Data Pre-processing

The original Cityscapes dataset contains 19 different labels, representing various objects and regions of interest in the urban environment. However, due to the limitations of computational power and the desire for better optimization, we will be reducing the number of labels to 10.

To achieve this reduction, we will employ K-Means clustering, a widely-used unsupervised learning technique, to group similar objects together. K-Means clustering will enable us to identify and merge labels that share similar characteristics and visual appearances. By clustering the original 19 labels into 10 distinct clusters, we aim to simplify the semantic segmentation task while still capturing crucial information about the objects in the driving scenes.

This reduction in label classes will not only enhance computational efficiency but also help in training and evaluating the segmentation models effectively. Our goal is to create a more streamlined and manageable dataset that retains essential information about the urban environment, enabling us to compare the models' performance and make informed decisions regarding the optimal model for enhancing the perception capabilities of autonomous vehicles.

3.2 Research Design

3.2.1 Research Methodology

In this experiment, we aim to compare the performance of UNet and R2UNet for semantic segmentation in autonomous vehicles using the "cityscapes-image-pairs" dataset. The dataset contains urban driving scenes with 19 original labels representing various objects and regions of interest. Due to computational

constraints, we will reduce the number of labels to 10 using K-Means clustering, grouping similar objects together while retaining crucial information.

First, we pre-process the dataset by reducing the labels and split it into training and validation sets. Next, we implement the UNet architecture in PyTorch, adapting the output layer to accommodate the reduced 10 labels. The model is trained using the training set with Cross-Entropy Loss as the loss function and tuned hyperparameters.
Following the training of UNet, we proceed with implementing R2UNet in PyTorch, which extends UNet with residual connections for improved feature flow and reuse. The output layer of R2UNet is adjusted to match the reduced-label task.

Both UNet and R2UNet models are trained on the reduced-label dataset, and their performances are evaluated on the validation set. We calculate metrics like Intersection over Union (IoU), F1 Score, and AUC to compare the segmentation results of both models.

3.2.2 Identification of Metrics

For evaluating the segmentation models, we will utilize three essential metrics: F1 score, Intersection over Union (IoU), and the Area Under the Curve (AUC).

1. Pixel-Accuracy: Pixel accuracy is a fundamental metric in semantic segmentation, offering valuable insights into the model's performance at a pixel level. This metric quantifies the proportion of correctly classified pixels in relation to the total number of pixels in the image dataset. Essentially, pixel accuracy provides a direct measure of how accurately the model assigns class labels to individual pixels, thus evaluating the model's ability to differentiate between distinct objects and regions within an image.

   A high pixel accuracy indicates that the model is capable of accurately segmenting objects, boundaries, and features, resulting in a faithful representation of the ground truth labels. On the other hand, a lower pixel accuracy suggests that the model struggles with identifying subtle variations in the image or is prone to misclassifications.

2. F1 Score: The F1 score is a widely used metric for evaluating the trade-off between precision and recall in binary classification tasks. In the context of semantic segmentation, we interpret each class as a binary classification problem, where each pixel is either correctly classified (positive) or misclassified (negative). The F1 score considers both false positives and false negatives, providing a balanced measure of the model's accuracy.

$$F1\ Score = 2 \times \frac{Precison \times Recall}{Precision + Recall}$$

3. Intersection over Union (IoU): IoU, also known as the Jaccard Index, quantifies the overlap between the predicted segmentation and the ground truth. It measures the proportion of correctly segmented pixels relative to the total number of pixels for each class. IoU is particularly useful for multi-class segmentation tasks, providing an aggregate assessment of the model's segmentation accuracy across all classes.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

4. Precision: Precision is a vital evaluation metric in semantic segmentation that assesses the accuracy of positive predictions made by the model. It quantifies the proportion of correctly predicted positive pixels in relation to all pixels classified as positive by the model. In essence, precision focuses on the model's ability to avoid false positive errors, which occur when the model predicts a pixel as belonging to a certain class when it actually doesn't.

5. Recall: Recall, often referred to as sensitivity or true positive rate, is a pivotal metric in semantic segmentation that assesses a model's ability to identify all relevant instances of a particular class within an image. It measures the ratio of correctly predicted positive pixels to all pixels that truly belong to the target class. In other words, recall quantifies the model's capacity to avoid false negatives – instances where the model fails to detect pixels that should have been classified as belonging to the target class.

Collectively, these metrics provide a comprehensive evaluation of a semantic segmentation model's effectiveness in capturing class regions accurately while managing trade-offs between different aspects of performance.

3.2.3 System Architecture

The system architecture for the semantic segmentation model in autonomous vehicles is designed to accurately and efficiently segment objects and regions of interest in real-time driving scenes. The architecture follows a deep learning approach, leveraging Convolutional Neural Networks (CNNs) and the U-Net architecture, which incorporates skip connections for improved feature propagation.

1. Encoding and Decoding Path: The system architecture consists of an encoding path, responsible for down-sampling the input image to capture context and high-level features, and a decoding path, which gradually up-samples the feature maps to generate the final segmentation map. The skip connections between the encoding and decoding layers allow for the transfer of fine-grained details, enhancing the accuracy of the segmentation.
2. Backbone CNN: The architecture incorporates a pre-trained CNN as the backbone, such as ResNet or VGG, which serves as a feature extractor. This enables the model to leverage learned hierarchical features from the backbone, enhancing its ability to recognize objects and patterns in the input image.
3. Semantic Segmentation Head: The system architecture features a semantic segmentation head that comprises a series of convolutional layers. The head refines the features extracted by the backbone and predicts the class labels for each pixel in the image, producing a dense pixel-wise segmentation map.
4. Loss Function: For training the model, a suitable loss function, such as the Dice Loss or Cross-Entropy Loss, is employed to measure the discrepancy between the predicted segmentation and the ground-truth labels. This helps guide the model towards accurate and precise pixel-wise predictions during training.
5. Deployment and Inference: Once trained, the model is deployed in the autonomous vehicle's perception system. It takes in real-time input from cameras and other sensors and then performs inference on the input image to generate the semantic segmentation output. The model's efficiency and real-time performance enable timely decision-making and safe navigation by the autonomous vehicle.

3.2.4 Software Tools

Pytorch: PyTorch is renowned for its flexibility and research-oriented approach, PyTorch equips developers with a rich set of tools and modules tailored for constructing, training, and deploying neural network models. Specifically, when dealing with tasks like semantic segmentation, PyTorch's capabilities shine through. It efficiently supports the construction of convolutional neural networks (CNNs), which are indispensable for carrying out pixel-wise classification tasks, such as segmentation.

NumPy: In scientific computing, NumPy assumes a pivotal role. This fundamental package is crucial for numerical operations and array manipulation. Often working in tandem with PyTorch, NumPy handles the intricacies of data processing and transformation. When preparing data for input into the neural network, NumPy's array manipulation capabilities prove invaluable.

DataLoader: The DataLoader utility by PyTorch, is essential during the training phase. Responsible for orchestrating dataset management, DataLoader empowers seamless iteration over data batches. What sets it apart is its knack for efficient loading of data in these batches, resulting in optimized

memory usage during the training process. Such memory management is pivotal in the era of data-hungry neural networks.

scikit-learn (sklearn): scikit-learn introduces a library renowned for its utility in machine learning tasks, offering various modules to aid in model development and evaluation. The sklearn.metrics module, for instance, stands poised to calculate essential performance metrics. This roster includes accuracy, precision, recall, and the F1 score—critical factors when evaluating the effectiveness of a model's segmentation prowess.

tqdm: tqdm's progress bars provide a dynamic representation of the model's advancement. This simple visual cue can greatly enhance the training experience, fostering a more intuitive grasp of progress. This makes it easier to visualize the progress and the loss as neural networks train and iterate.

matplotlib (plt): Through matplotlib, graphs are plotted and images are displayed, bridging the gap between raw data and human comprehension.

imageio: The imageio library is crucial for reading and writing image files. imageio facilitates the seamless manipulation of images, serving as a key intermediary between raw image data and the model.

Chapter 4
Results of the Empirical Investigation

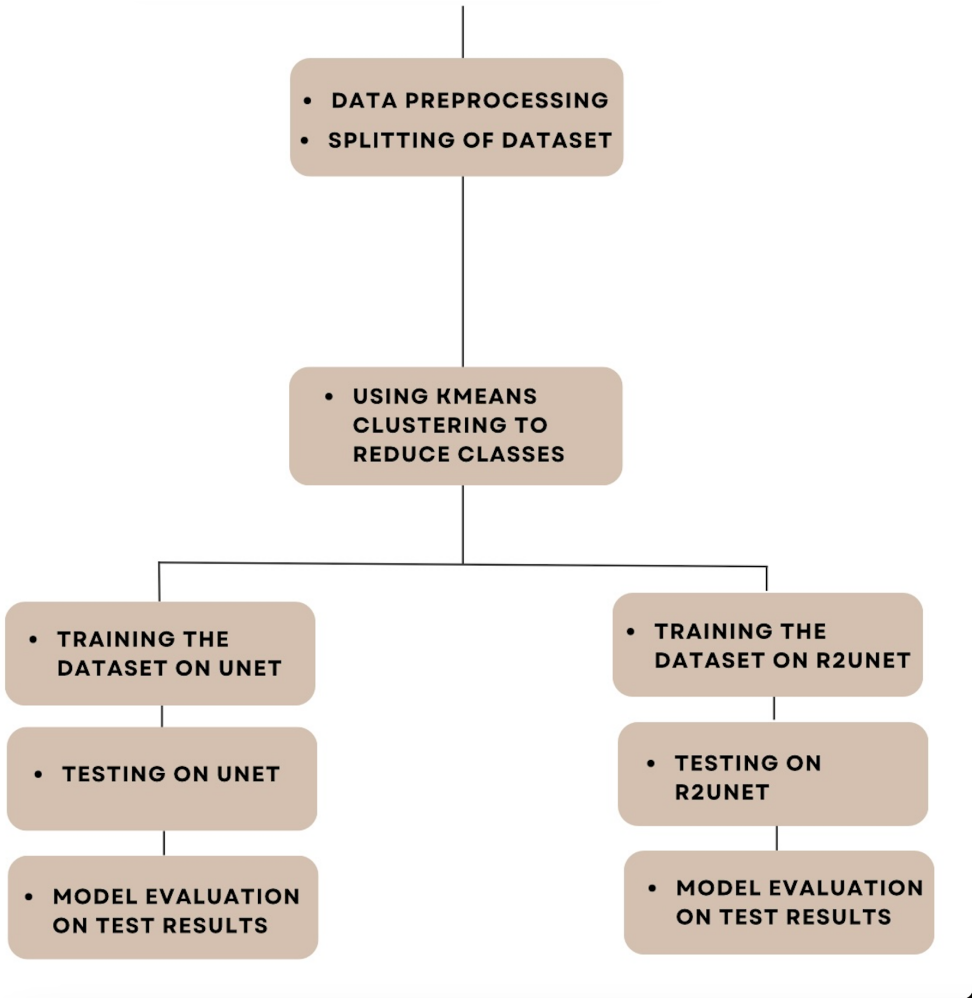

Figure 13. Flowchart of the Implementation

This chapter delves into the practical implementation of the semantic segmentation models, offering a detailed account of how the UNet and R2UNet architectures were built using PyTorch. It explains the preparation of the dataset for training and testing, encompassing data loading, splitting, and transformation steps. The chapter further outlines the training procedure, including the optimization process, loss functions, and the integration of a saturation loss mechanism. The main focus is on the evaluation and comparison of the models, with an emphasis on metrics such as accuracy, precision, recall, F1 score, intersection over union (IoU), and loss. Finally, the chapter touches upon the generalization capability of the models and highlights any computational constraints encountered during experimentation.

4.1 Dataset

Generally data pre-processing is an important step at the initial phase of the project for implementing a UNet/ R2UNet model. Data pre-processing is the method of transforming unstructured data into a more structured form. Raw data sets are generally inconsistent and have either information lacking and may contain certain errors that can impact the training and testing results. This can especially happen as the data is gathered through multiple sources and hence depending on the source there are possibilities of

source format anomalies. Therefore through a pre-processing step, the data is reviewed and errors such as duplicates and inconsistencies are cleaned, thereby making the dataset now ready for implementation. In context of this project, since the data source was from Kaggle, the pre-processing step was already completed, and not required to be performed for this project.

The dataset originally comprised two primary folders: one for validation and another for training. In order to facilitate model assessment, I amalgamated the data from these folders into a consolidated dataset. Subsequently, I meticulously divided this merged dataset into three distinct subsets: 70% for training, 20% for validation, and the remaining 10% for testing purposes. This stratified allocation of data ensured that each subset contained an equitable representation of the original dataset's diversity.

The initial dataset contained images where the original image and its corresponding ground truth annotation were combined within a single image. As a pre-processing step, The images needed to be separated into two different images: the original image capturing the visual scene and the ground truth image representing pixel-wise annotations.

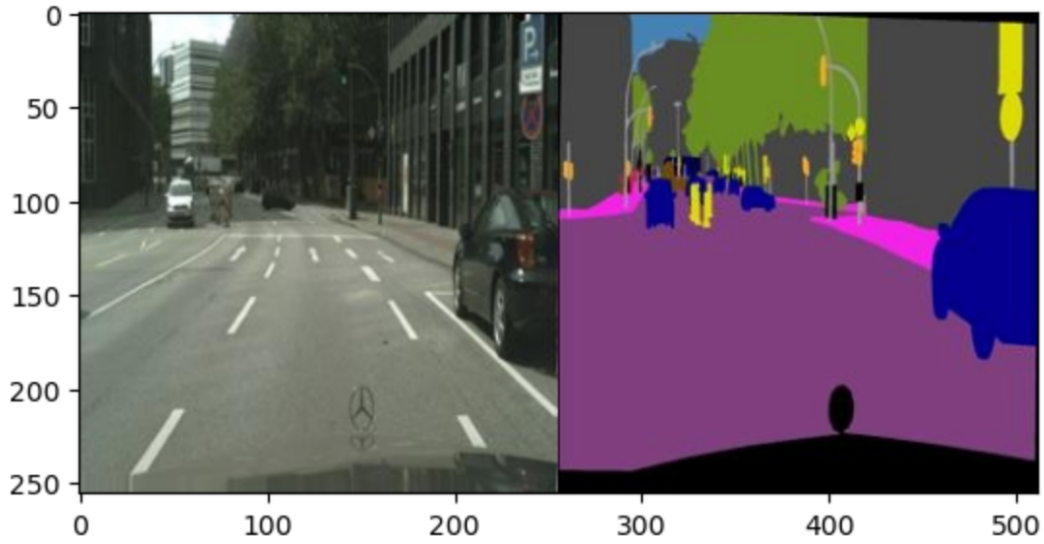

Figure 14. Initial Dataset

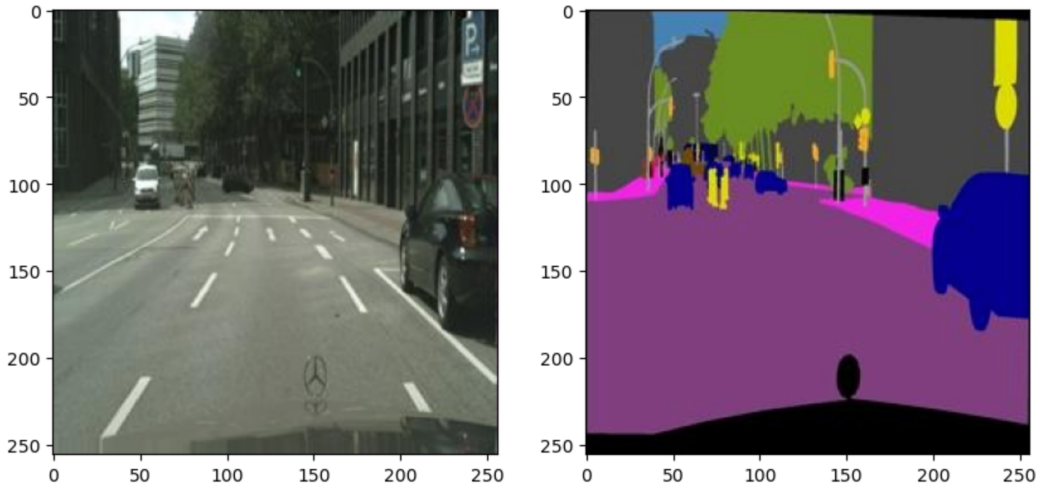

Figure 15. Dataset after splitting

4.2 Dataset Class

The CityscapeDataset class is a custom dataset class designed for data loading and preprocessing in the context of semantic segmentation tasks, particularly for Cityscapes dataset images. This class inherits from the torch.utils.data.Dataset class and encapsulates the necessary functionalities to efficiently handle and manipulate the dataset.

Initialization: The class is initialized with two primary arguments: image_paths and label_model. image_paths represent a list of file paths pointing to the images within the dataset, while label_model is a machine learning model (such as KMeans in this context) used for image label generation.

Data Loading: The __getitem__ method enables data loading and preprocessing for a specific index. It reads an image from the provided file path, converts it into an RGB format, and splits the image into two parts: the actual cityscape image and the label image. The label image is then processed using the label_model to generate label class predictions. These predictions are reshaped into a 2D format of size 256x256. The cityscape image is transformed using predefined transformation operations, including converting it into a PyTorch tensor and normalizing its values.

Data Splitting and Transformation: The split_image method is responsible for splitting the provided image into two halves: the left half containing the cityscape and the right half containing the label. The transform method applies a series of transformations to the input image. These transformations include converting the image to a tensor format and normalizing its pixel values based on pre-defined mean and standard deviation values.

Data Length: The __len__ method provides the length of the dataset, which is determined by the number of image paths present in the image_paths list.

4.3 DataLoader

The DataLoader utility is an essential component in our model implementation, serving as a powerful tool for managing and processing our datasets during the training and evaluation phases. With its capability to efficiently load data in batches, DataLoader helps optimize memory utilization while streamlining the overall training pipeline. In the provided code snippet, we have exemplified its utility by utilizing it to create three distinct data loaders: one each for training, validation, and testing.

In the training phase, the `train_data_loader` is constructed to facilitate the iterative processing of our training dataset. By specifying a batch size of 4, we dictate the number of samples processed in parallel during each training iteration. This efficient batching enhances computational efficiency while allowing the model to benefit from the inherent parallel processing capabilities of modern hardware. Furthermore, the `shuffle` parameter set to `True` ensures that the data is randomly shuffled after each epoch, preventing the model from inadvertently learning sequence-related biases and promoting generalization.

Similarly, for validation and testing, the `val_data_loader` and `test_data_loader` are created with the same batch size. However, since shuffling is not necessary during these phases, the `shuffle` parameter is set to `False`. The DataLoader's ability to divide the dataset into manageable batches and efficiently load them into memory enables seamless integration with our model, ultimately contributing to the overall robustness and accuracy of the semantic segmentation task.

4.4 K-Means Clustering

In the context of our semantic segmentation implementation, K-means clustering assumes an unconventional role as a preprocessing step to facilitate our semantic segmentation task. Unlike its typical usage, where it categorizes data points into clusters, here it is employed to derive class labels for our segmentation model.

The advantages of K-means clustering are as follows:

- It is easy to implement and makes it a go-to choice for this task.
- K-means clustering can handle large datasets efficiently.
- K-means is a scalable clustering that can handle large datapoints.
- K-means can be easily used for different applications and can be used with different initialized methods.

There are also certain disadvantages with K-means:

- It is sensitive to the initial selection of centroids.
- They may converge might not be optimal.
- It requires a specific set of clusters.
- Anomalies in the data might have an impact in clusters.
- The K-means algorithm finds clusters through various methods with one of them being a trial and error method which specifies the value of 'k'. The second method to find K-means to use clusters is the elbow technique. Once the 'k' value is obtained, the system will randomly assign the centroids and measure the distance between the datapoint and the centroid. Through this points are assigned to corresponding centroids from which the distance is minimum. Therefore, we will be able to find the 'k' number of the initial clusters.

$$WSS = \sum_{i=1}^{m} (x_i - c_i)^2$$

Where, WSS is the within sum of squares measure,
x_i , is the data point
c_i , is the closest point to the centroid.

In our implementation, we apply K-means clustering to the colour values present in our ground truth labels. This unconventional approach involves grouping colour values into a specified number of clusters, determined by the parameter 'num_classes'. Subsequently, these clusters are treated as distinct class labels for our semantic segmentation task. This unique application is particularly useful when ground truth labels are not natively available in categorical format, and instead, they are encoded using colours. By utilizing K-means clustering, we effectively convert these color-coded labels into discrete class labels, creating a link that enhances the learning process of our segmentation network.

In our project, the number of classes have been reduced to 10 and the colour has been chosen at random. K-Means, renowned for its utility in unsupervised machine learning, endeavours to segregate a given collection of colour data, represented as an array of RGB values, into distinct clusters based on colours. Through these methods, K-means can be used for image compression by reducing the number of colours in the input image while ensuring that the quality of the image remains intact. Herewith the algorithm can utilize the colours in the input image, cluster them and replace the pixels with the centroid colour within each cluster. This creates a compressed image.

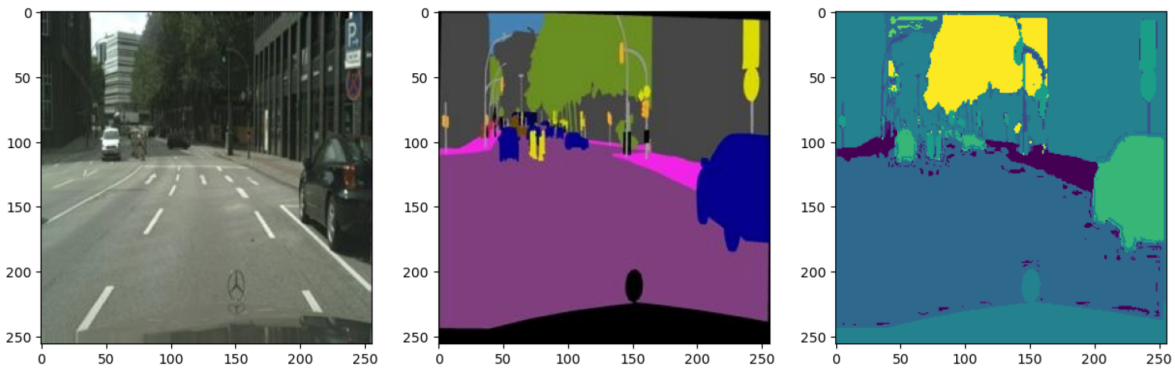

Figure 16. Applying KMeans Clustering to the Dataset

4.5 Batch Size

Batch size is a crucial hyperparameter in machine learning and deep learning that determines the number of training samples presented to the model in each iteration of the training process. When training a model, the dataset is divided into smaller subsets called batches. During each training iteration, the model processes one batch of data, updates its internal parameters based on the computed loss, and repeats this process until all batches have been utilized, constituting an epoch.

For this experiment, a batch size of 4 was used based on the computational resources available. Given that dataset was computationally intensive, using a smaller batch size helped manage memory constraints while still allowing the model to make frequent updates during training. This balance between efficient memory usage and training progress was likely a key consideration in achieving successful training outcomes.

4.6 Loss Function

For this model I have chosen the Cross Entropy Loss. The Cross Entropy Loss is a commonly used loss function for classification tasks, including semantic segmentation. It measures the dissimilarity between the predicted class probabilities and the true class labels. The goal is to minimize this dissimilarity during training.

For semantic segmentation, where each pixel needs to be classified into one of multiple classes, the Cross Entropy Loss calculates the error between the predicted pixel-wise class probabilities and the ground truth labels. This loss function encourages the model to assign high probabilities to the correct classes and lower probabilities to incorrect ones.
The formula for the Cross Entropy Loss is:

$$L(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log\left(\hat{y}_{ij}\right)$$

Where:
- N is the number of pixels or samples.
- C is the number of classes.
- y_{ij} is the ground truth label.
- $\hat{y}_{ij}$ is the predicted probability.

The goal during training is to minimize this loss, which involves adjusting the model's parameters using the optimizer. This process aims to enhance the model's ability to accurately segment and classify objects within the input images.

4.7 Optimizer

We have employed the Adam optimizer in our model due to its effectiveness in optimizing neural networks. Adam stands for "Adaptive Moment Estimation," and it combines the benefits of both stochastic gradient descent (SGD) and momentum-based optimization methods. The optimizer adjusts the learning rate dynamically for each parameter, adapting to the parameter's historical gradient information and the magnitude of past gradients. This adaptability enables Adam to navigate complex optimization landscapes and converge more efficiently compared to fixed learning rate methods. Additionally, the momentum term in Adam helps accelerate convergence by considering previous gradients' influence. These characteristics make Adam a popular choice for optimizing deep learning models, as it efficiently balances exploration and exploitation during training, leading to faster convergence and improved generalization.

4.8 Learning Rates

Learning rates are crucial hyperparameters in machine learning that determine the step size during model optimization. They play a pivotal role in shaping the training process by influencing how quickly a model converges to an optimal solution. A higher learning rate can lead to faster convergence, but it runs the risk of overshooting and instability. Conversely, a lower learning rate may ensure stability but could slow down convergence. Striking the right balance is essential, as an improperly set learning rate can hinder a model's ability to learn effectively and generalize well to new data. Thus, tuning learning rates is a fundamental aspect of model development to ensure efficient and accurate learning.

In this implementation, we have explored 3 different learning rates:

- **0.1:** This higher learning rate led to rapid convergence, but the model's accuracy exhibited erratic behaviour. Overshooting of the optimal solution caused instability during training.
- **0.01:** At this learning rate, the model's convergence was more stable. Accuracy improved consistently without dramatic oscillations, offering a favourable trade-off between stability and convergence speed.
- **0.001:** The lowest learning rate resulted in steady convergence with a focus on accuracy. Although the training process was slower, the model exhibited consistent improvements.

4.9 Kernel Size

In both the UNet and R2UNet architectures, the choice of kernel size for convolutional operations plays a critical role in feature extraction. In the UNet model's contracting blocks, convolutional layers employ a 3x3 kernel size. This size is a common choice as it allows capturing local spatial information while considering immediate neighboring pixels. Similarly, the middle layer and the expansive blocks also utilize a 3x3 kernel for convolutional operations. In the R2UNet architecture, the Up_Sample_Conv class employs a 3x3 kernel for convolution after the up-sampling operation. This kernel size balances capturing local features and allowing spatial expansion during the up-sampling process. The Repeat class and the RR_Conv class in the R2UNet model also use a 3x3 kernel for their convolutional layers, facilitating feature refinement and adjustment of channel dimensions.

4.10 Stride

The stride parameter determines the step size of the convolutional filter as it slides across the input image. In the UNet architecture, convolutional layers in the contracting blocks and middle layer use a stride of 1. This choice ensures that neighboring pixels are considered, maintaining spatial resolution and capturing detailed features. Conversely, the expansive blocks in the UNet model employ a stride of 2 in their transposed convolution operations, effectively achieving up-sampling by skipping alternate pixels. Similarly, in the R2UNet architecture, the Up_Sample_Conv class employs a stride of 1 after up-sampling, while the RR_Conv class's 1x1 convolution maintains a stride of 1 to retain feature information.

4.11 Padding

Padding involves adding extra pixels to the input image before convolution to preserve spatial dimensions and mitigate the reduction in feature map size. In the UNet architecture, contracting blocks, middle layers, and expansive blocks use a padding of 1 for their convolutional layers. This padding maintains the input's spatial dimensions, preventing loss of information during convolution. In the R2UNet model, the Up_Sample_Conv class employs a padding of 1 after the up-sampling operation to preserve spatial information, and the RR_Conv class does not require explicit padding due to the 1x1 convolution's nature. The choice of padding ensures that convolutional operations do not lead to the reduction of feature map size and maintain accurate information flow throughout the network.

4.12 Model Classes

4.12.1 UNet

The UNet class, an extension of PyTorch's nn.Module, is equipped with a series of components, each tailored to a specific phase of the architecture:

Contracting Blocks: These initiate the architecture. Each block encompasses two convolutional layers. These layers have a kernel size of 3x3, stride of 1, and padding of 1, maintaining spatial dimensions. A Rectified Linear Unit (ReLU) activation function follows each convolutional layer, enhancing non-linearity. Batch normalization is applied after each activation. The number of input and output channels evolves across the blocks: from 3 to 64, 64 to 128, 128 to 256, and 256 to 512.

Middle Layer: Positioned between contracting and expansive phases, this layer employs a convolutional block with a kernel size of 3x3, stride of 1, and padding of 1. It maintains the spatial dimensions and enhances feature complexity.

Expansive Blocks: These orchestrate the upscaling process. Transposed convolutions are used with a kernel size of 3x3, stride of 2, and padding of 1. This achieves up-sampling, and an output padding of 1 ensures spatial dimensions match expectations. Each expansive block's convolutional layers mirror the contracting blocks, with similar parameters. The number of input and output channels varies, progressing from 1024 to 512, 512 to 256, 256 to 128, and 128 to 64.

Output Layer: The final output layer employs a convolutional layer with a kernel size of 3x3, stride of 1, and padding of 1. It generates the segmentation mask with the specified number of output channels (classes).
The forward pass through the network coordinates the contracting and expansive phases, employing max-pooling for down-sampling and transposed convolutions for up-sampling.

4.12.2 R2UNet

The R2UNet model class contains of the following components. This architecture excels in semantic image segmentation tasks by effectively capturing both local and contextual information.

‘Up_Sample_Conv’ Class: This class encapsulates an up-sampling operation followed by a convolutional layer. The up-sample operation increases spatial resolution, and the subsequent Conv2d layer employs a 3x3 kernel, stride of 1, and padding of 1. Batch normalization and ReLU activation follow the convolution. The module enhances feature representations during the up-sampling process.
Repeat: This class defines a repeating convolutional block that aids in feature refinement. The block comprises a 3x3 convolution, followed by batch normalization and ReLU activation. It repeats this process twice, with an important "in-place" setting that directly modifies the input to save memory.

‘RR_Conv’ Class: This class combines the Repeat block with a 1x1 convolutional layer. The Repeat block refines features using the Repeat module described above. The 1x1 convolutional layer adjusts channel dimensions to match the desired output.

R2UNet: This class assembles the entire architecture. The architecture contains five encoding layers, each consisting of an 'RR_Conv' block. The channel dimensions increase progressively, with each layer doubling the channels of the previous one. The max-pooling operation follows each encoding layer to reduce spatial dimensions.

'DeConvLayer' Class: These layers are responsible for up-sampling the feature maps and decoding. 'Up_Sample_Conv' class layers perform the up-sampling and convolution, while the 'Up_Layer' block employs 'RR_Conv' blocks to refine features during the up-sampling process.

'Conv': This final layer performs a 1x1 convolution to generate the segmentation output, with the number of output channels determined by the specified output channel.
Overall, the R2UNet architecture effectively combines the U-Net's powerful segmentation capabilities with Recurrent Residual Blocks, enhancing the network's ability to capture complex spatial relationships and achieve superior semantic segmentation results.

4.13 Training Function

Our training function serves as the core of our model development process. It optimizes the model's parameters to fit the training data and enhance its performance. We have introduced a crucial enhancement by implementing a saturation loss mechanism, which ensures that training halts when the validation loss stops decreasing.

The training function iterates through the dataset, updating the model's parameters using the Adam optimizer. During each iteration, the model's predictions are compared to the ground truth labels, and the loss is calculated. This loss is then used to adjust the model's parameters via backpropagation, facilitating learning and improving accuracy.

During the course of model training, it's imperative to assess the model's performance on a separate validation dataset to gauge its generalization capabilities. The validation loss, often computed as a metric of the disparity between predicted outputs and ground truth labels, serves as a vital indicator of the model's predictive accuracy. In certain scenarios, as the training progresses, the validation loss might exhibit fluctuations or a gradual decrease, signifying improved learning.

To enhance efficiency, we've incorporated a saturation loss criterion. This safeguard triggers when the validation loss fails to decrease substantially over a threshold of 0.1. This threshold is used to compare the current epochs loss with the previous epoch. This strategy prevents the model from needlessly continuing training when little improvement is observed, saving computational resources and time.

Incorporating the saturation loss criterion empowers our training process to terminate gracefully, ensuring we maximize the model's performance while avoiding unnecessary training iterations that yield diminishing returns.

4.14 Testing Function

Our testing function serves as a step in assessing the performance of our trained semantic segmentation model. It provides insights into how effectively the model generalizes to new, unseen data.
The testing function takes the trained model and a testing dataset as input. It iterates through the dataset, making predictions for each input sample using the trained model. These predictions are then compared to the ground truth labels to calculate various evaluation metrics, such as accuracy, precision, recall, F1 score, and intersection over union (IoU).

By evaluating these metrics, we gain a comprehensive understanding of the model's performance across different aspects of semantic segmentation. Accuracy reveals the overall correctness of predictions, while precision and recall measure the model's ability to correctly identify positive instances. The F1

score provides a balanced measure between precision and recall. IoU offers insights into the overlap between predicted and true segmentations.

4.15 Model Limitations

The models developed in this study, UNet and R2UNet, offer valuable insights into urban scene segmentation. Both models demonstrate commendable generalization across diverse landscapes, highlighting their potential for real-world application. However, it's important to recognize that the advanced architecture of R2UNet comes at a computational cost. The intricacies of this architecture lead to extended training times and heightened computational requirements, which could restrict its real-time feasibility in time-sensitive applications.

Moreover, both models may encounter challenges when faced with intricate urban scenes or scenarios not encountered during training. Variations in lighting, occlusions, and unusual objects could affect segmentation accuracy. Class imbalance within the data could also impact the models' performance, particularly for less frequent classes.

To mitigate these limitations, potential strategies include augmenting the training data, exploring transfer learning, and optimizing R2UNet's efficiency. Addressing these challenges would enhance the models' adaptability to a wider array of scenarios, ensuring their effectiveness in practical urban scene segmentation applications.

Chapter 5
Validation of Results

5.1 Results

In this chapter we will be delving into the empirical results of our two data models – UNet and R2UNet. The structure of this chapter will showcase the the results of UNet followed by R2UNet. The third section will summarize the findings of our two models.

5.1.1 UNet

5.1.1.1 Metrics

Table 1. Metrics for UNet

Learning Rate	Accuracy	Precision	Recall	F1 Score	IoU	Loss
0.1	0.66	0.65	0.66	0.63	0.48	1.43
0.01	0.78	0.76	0.78	0.76	0.76	6.27
0.001	0.86	0.85	0.86	0.84	0.75	0.50

5.1.1.2 Validation and Training Loss Plots

Learning Rate: 0.1

In the training process, it seems that the validation loss started to saturate and became significantly larger after just one epoch. This suggests that the model's performance on the validation dataset deteriorated dramatically after a single epoch of training. The validation loss value being much larger than the training loss value indicates a potential overfitting issue.

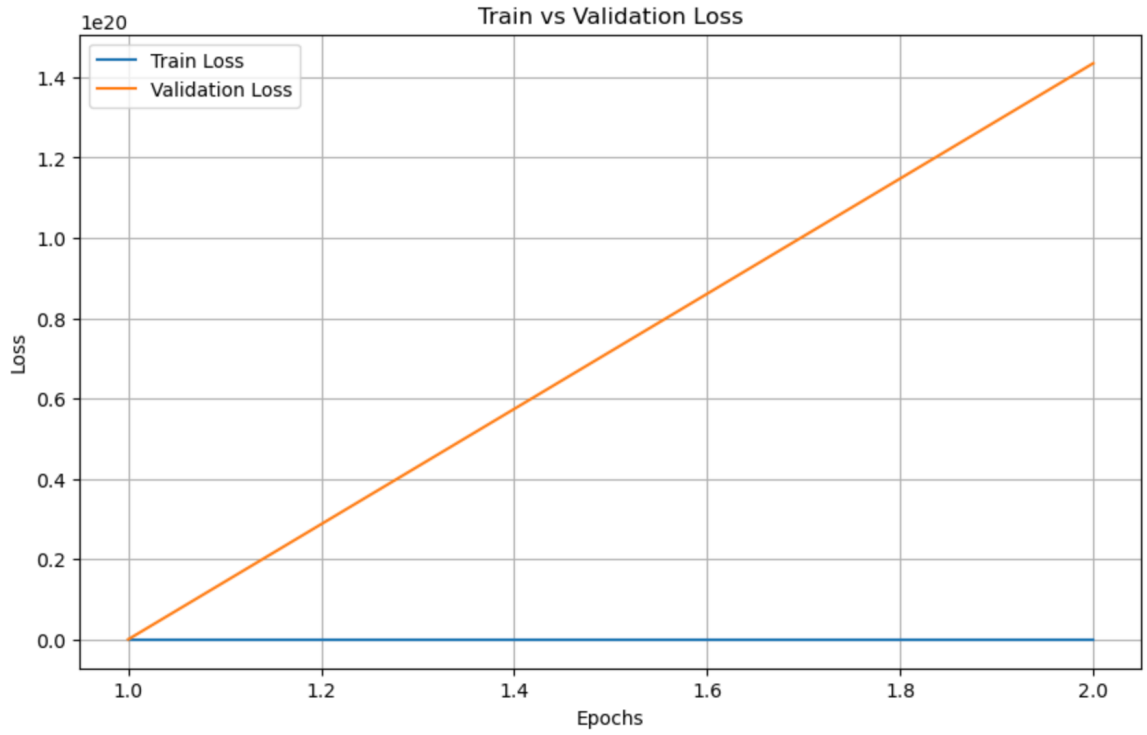

Figure 17. Training vs Validation Loss for Learning Rate of 0.1

Learning Rate: 0.01

- This model generally shows lower training losses across epochs compared to the previous model. This indicates that the current model is learning the training data more effectively.
- This model also demonstrates lower validation losses across most epochs. This suggests that the higher learning rate might have allowed the model to converge more quickly to a better solution.

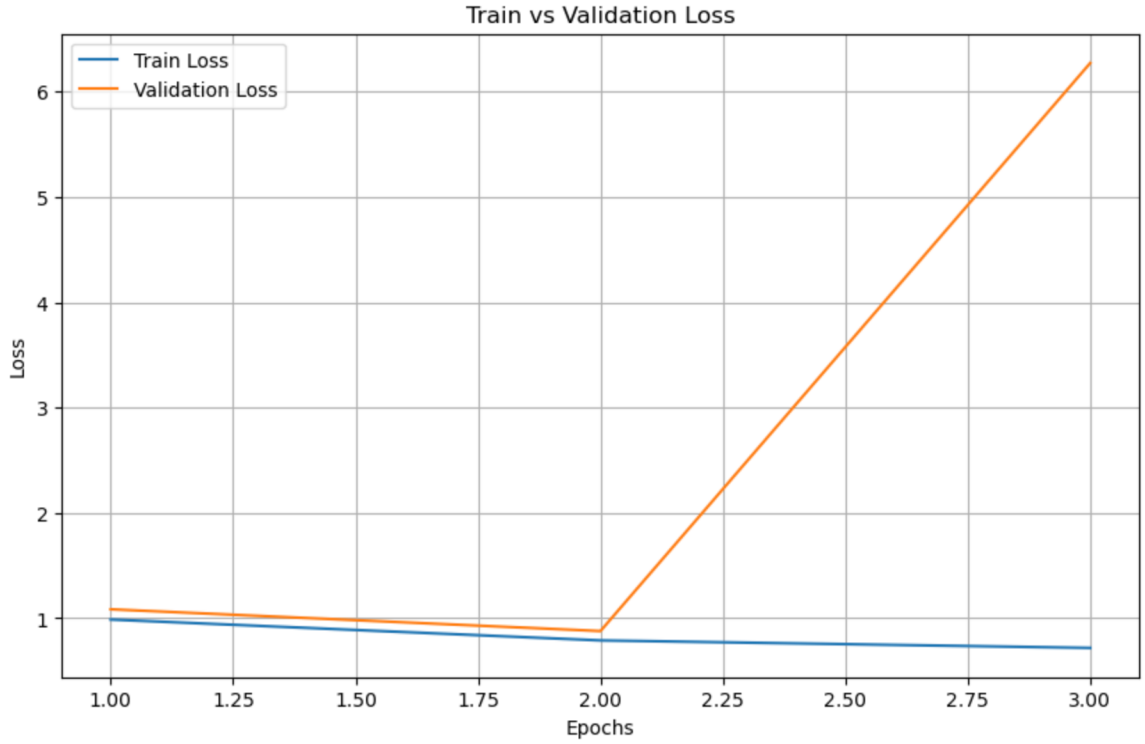

Figure 18. Training vs Validation Loss for Learning Rate of 0.01

Learning Rate: 0.001

- This UNet model shows lower training losses compared to the the models trained at a learning rate of 0.1 and 0.01. This indicates improved learning in subsequent models.
- The learning rate of 0.001 out of all the models generally demonstrates lower validation losses compared to the first set, indicating better generalization ability.
- This model, which uses more epochs, shows significant improvement in validation losses over time.

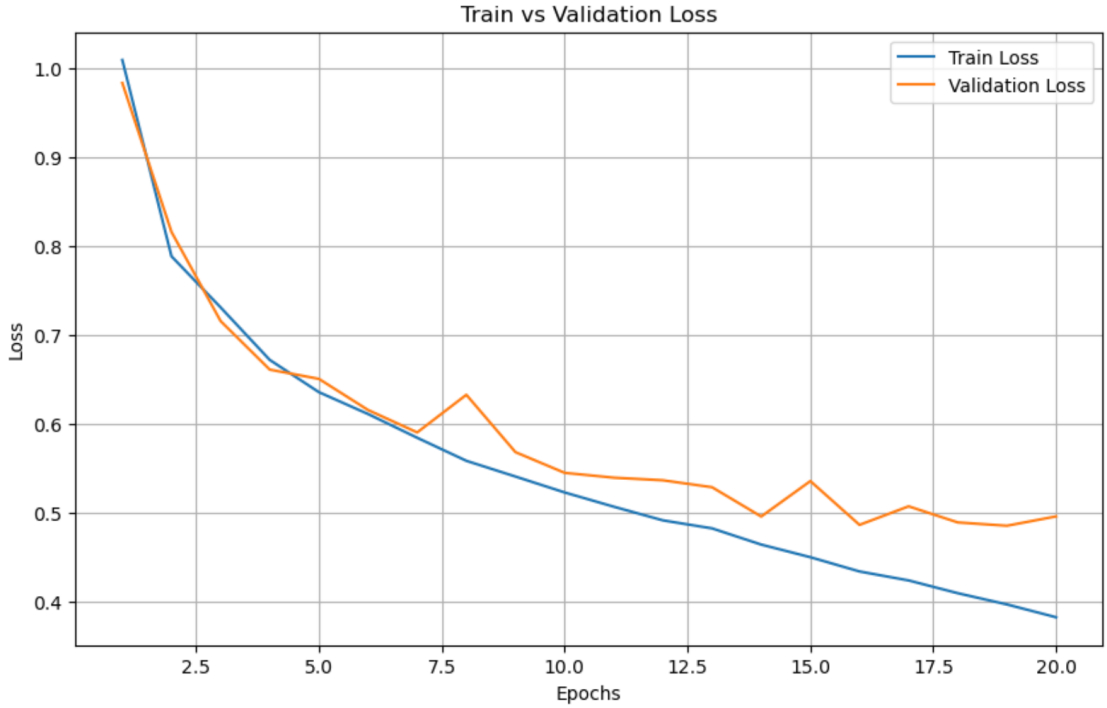

Figure 19. Training vs Validation Loss for Learning Rate of 0.001

5.1.1.2 Output Images

Learning Rate: 0.1

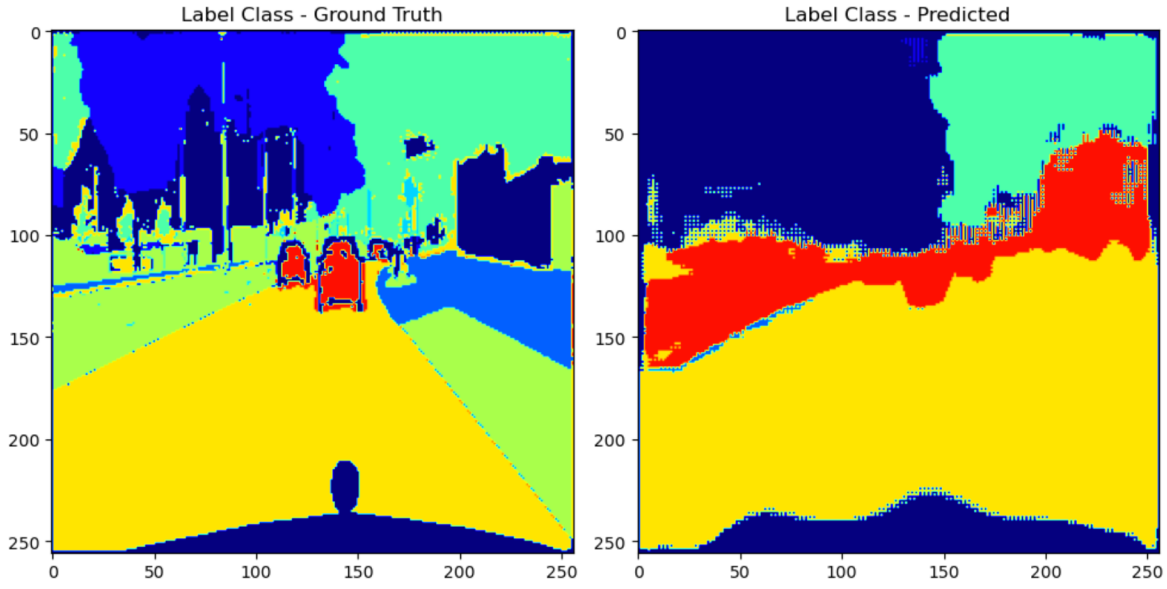

Figure 20. Ground Truth vs Predicted Image for Learning Rate of 0.1

Learning Rate: 0.01

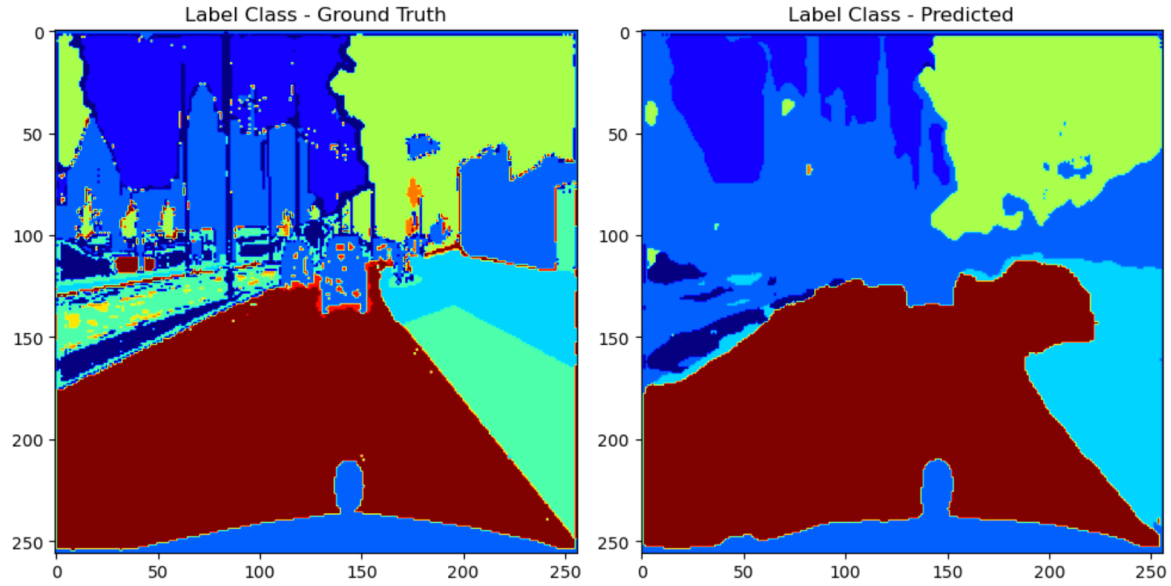

Figure 21. Ground Truth vs Predicted Image for Learning Rate of 0.01

Learning Rate: 0.001

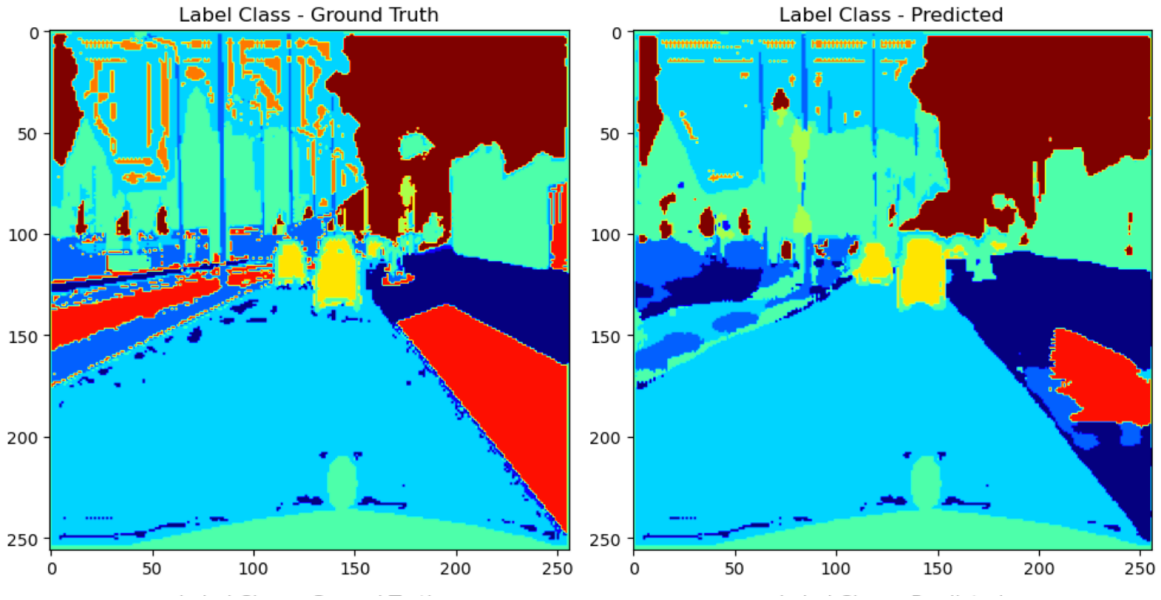

Figure 22. Ground Truth vs Predicted Image for Learning Rate of 0.001

5.1.2 R2UNet

5.1.2.1 Metrics

Table 2. Metrics for R2UNet

Learning Rate	Accuracy	Precision	Recall	F1 Score	IoU	Loss
0.1	0.47	0.45	0.47	0.44	0.30	2.40
0.01	0.81	0.78	0.81	0.78	0.70	1.28
0.001	0.86	0.85	0.86	0.85	0.76	0.87

5.1.2.3 Validation and Training Loss Plots

Learning Rate: 0.1

With the learning rate as 0.1, the validation loss started to saturate and even slightly increased after the third epoch. This suggests that the model's performance on the validation dataset began to stabilize and potentially degrade after a few epochs. However, this would be due to overshooting of the model due to having a high learning rate.

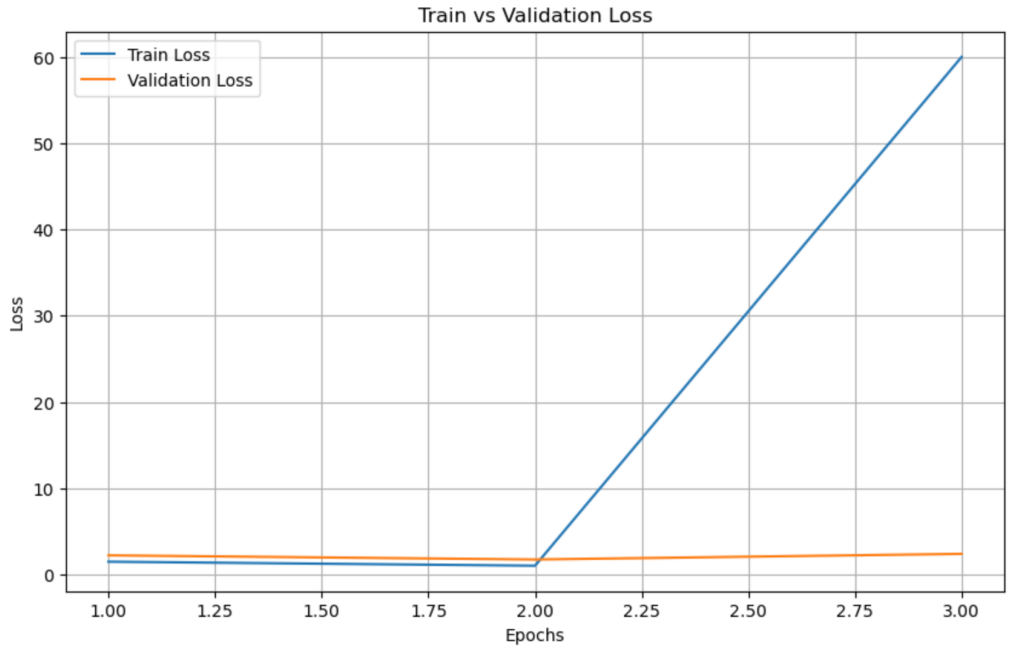

Figure 23. Training vs Validation Loss for Learning Rate of 0.1 for R2UNet

Learning Rate: 0.01

- The model when trained with a learning rate of 0.01 generally shows lower training losses across epochs compared to the previous model. This indicates that the current model is learning the training data more effectively.
- This model also demonstrates lower validation losses across most epochs. This suggests that the higher learning rate might have allowed the model to converge more quickly to a better solution.
- The previous learning rate which was 0.1 had extremely high validation loss values in the second epoch indicating a significant issue, possibly due to divergence or data-related problems.

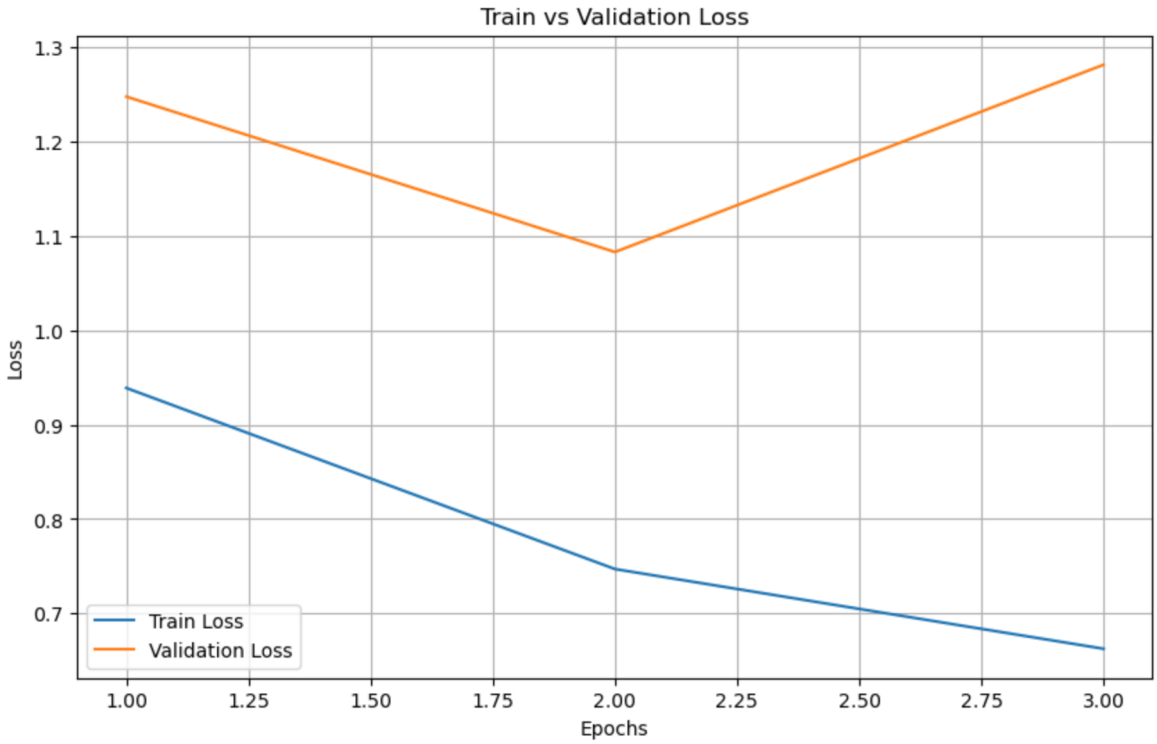

Figure 24, Training vs Validation Loss for Learning Rate of 0.01 for R2UNet

Learning Rate: 0.001

- The learning rate of 0.001 of UNet models demonstrates consistently lower training losses compared to both the first and second sets. This indicates improved learning and optimization.
- The third set of models showcases notably lower validation losses across most epochs compared to the previous sets. This suggests improved generalization to the validation data.

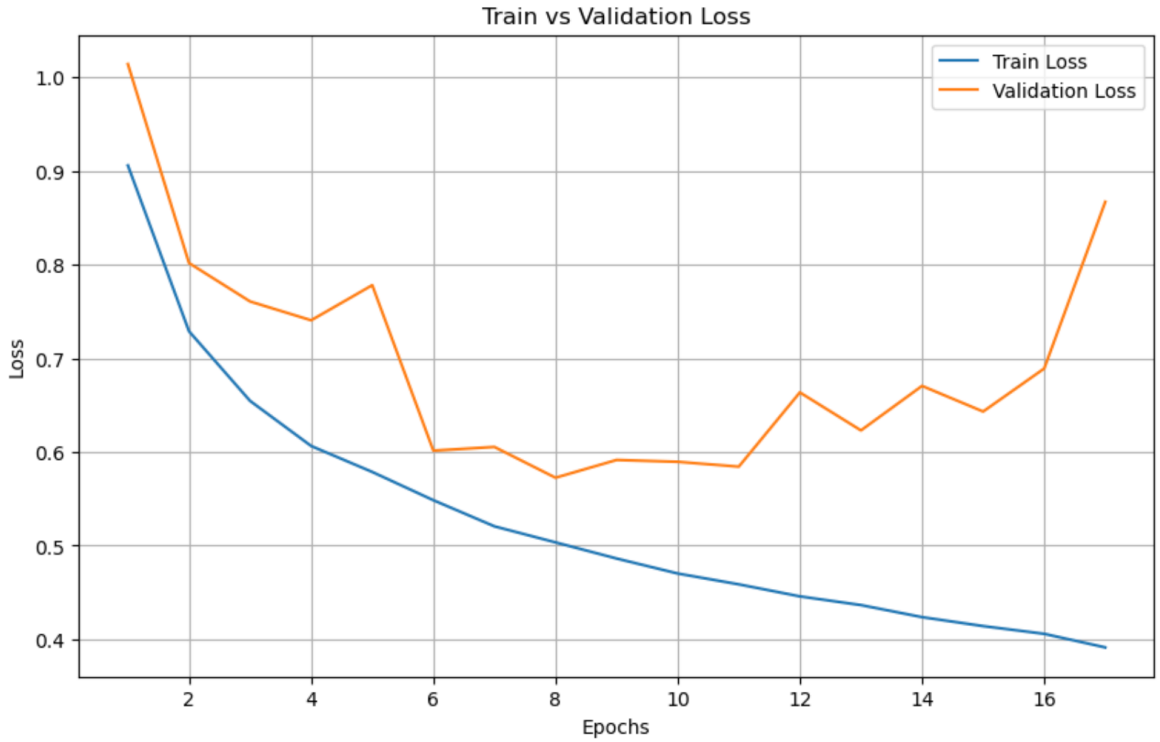

Figure 25. Training vs Validation Loss for Learning Rate of 0.001 for R2UNet

5.1.2.3 Output Images

Learning Rate: 0.1

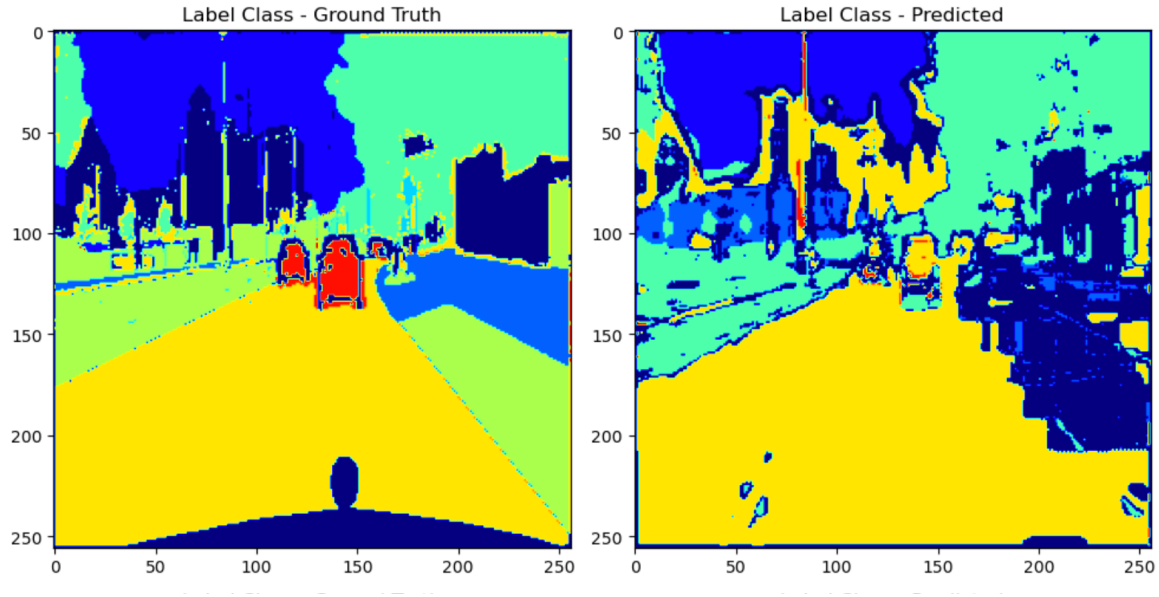

Figure 26. Ground Truth vs Predicted Image for Learning Rate of 0.1 for R2UNet

Learning Rate: 0.01

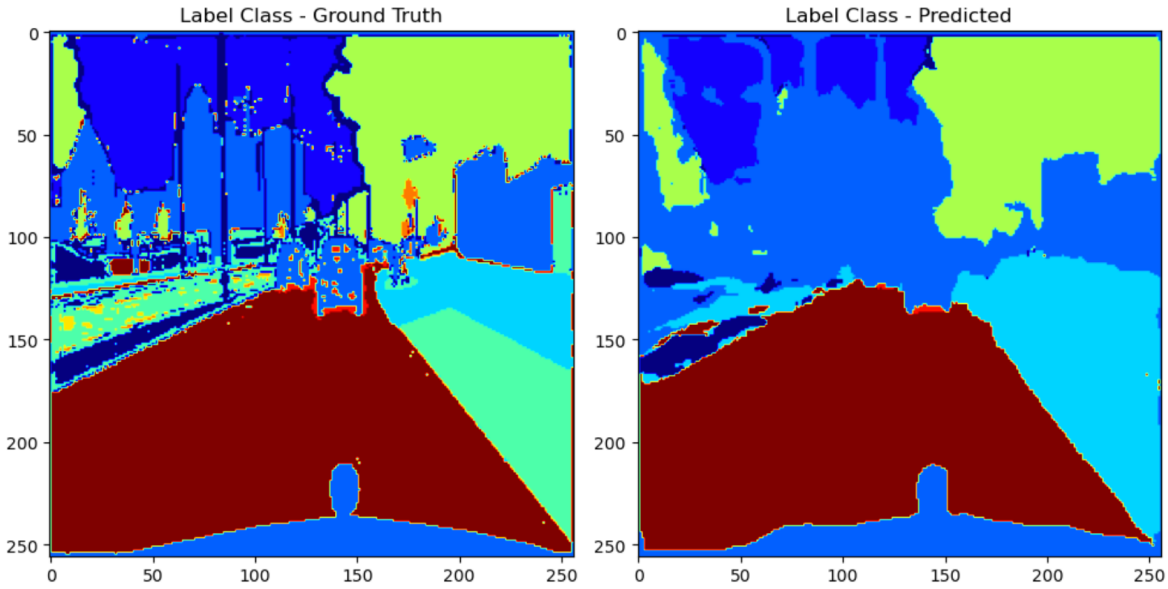

Figure 27. Ground Truth vs Predicted Image for Learning Rate of 0.01 for R2UNet

Learning Rate: 0.001

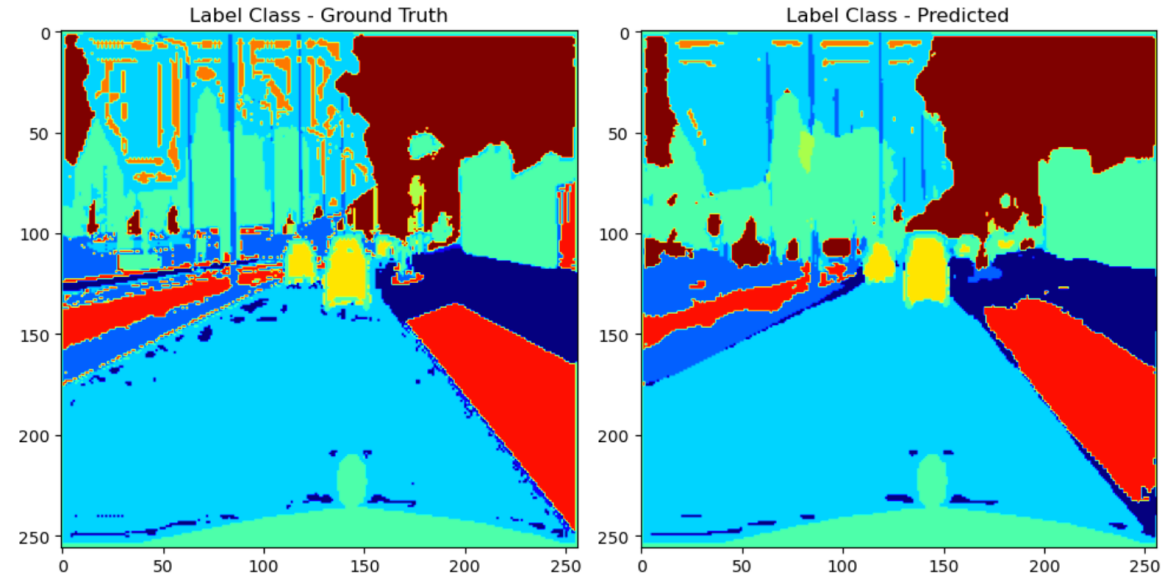

Figure 28. Ground Truth vs Predicted Image for Learning Rate of 0.001 for R2UNet

5.1.2 Comparison of the Models

We can observe from the above results that both the models stop the training early for higher learning rates. This is likely to be due to overshooting during training. The comparison of the metrics for the two models helps us understand a lot about their performance and efficiency.

- **Accuracy:** The R2UNet model tends to have a higher accuracy compared to the UNet model across all learning rates. The UNet model seems to achieve a better accuracy, especially at higher learning rates like 0.1.
- **Precision, Recall, and F1 Score:** In terms of precision, recall, and F1 score, both models show similar trends. The UNet model outperforms the R2UNet model, with consistently higher scores at

all learning rates. This indicates that the UNet model has a better balance between correctly identifying positive instances and minimizing false positives and false negatives.

- **IoU (Intersection over Union):** The IoU metric measures the overlap between predicted and ground truth masks. Here again, the UNet model demonstrates superior performance, maintaining higher IoU scores across learning rates. This signifies better spatial alignment between predicted and actual segmentations.
- **Loss:** The loss value measures the disparity between predicted and ground truth labels. A lower loss indicates better convergence of the model during training. Notably, the UNet model consistently achieves lower loss values compared to the R2UNet model at all learning rates.

Chapter 6
Conclusions and Future Work

6.1 Conclusion

In this project, we have delved into the concepts of Deep Learning and Convolutional Neural Networks in detail through literature study on the various models and techniques available in this area. The primary focus of the project's empirical research was targeted towards 2 models: UNet as the go-to model which is known to have excellent results and R2UNet which has limited research but high potential beyond the field of biomedical image segmentation.

A framework was developed for comparing these models to perform image segmentation of city road landscapes to evaluate the ability of these models to provide a suitable result based on the identified performance metrics, the use of which is evaluated for autonomous vehicles.

The research methodology followed the steps of literature review, defining the aim and objective of the project, designing the experiment and data requirements for the empirical research, developing the experiment & data model training and evaluation of the empirical results.

We used the following metrics to evaluate and compare the 2 models for different learning rates (0.1, 0.01, 0.001) :

- Accuracy
- Recall
- Precision
- F1 score
- IoU (intersection over Union)

As per the results UNet performed better than R2UNet in 0.1%, while R2UNet has performed marginally better in 0.01% & 0.001% than UNet. The reason for observing a marginal improvement is due to a lower learning rate.

Hence we can conclude that R2UNet has the potential to perform better than UNet for the above metrics, if run for more epochs which will improve the model training time and ability to perform better in validation settings. Also by setting the learning rate at less than 0.001% we should be able to observe even better results with R2UNet. However the drawback here is that with an increased learning rate, the time taken for processing also increases with R2Unet and as per the literature review cost is also factor that will be impacted due to this.

We can safely conclude that the through the project implementation, the following objectives were performed:

We were successfully able to establish that CNN based deep learning models are of high value and importance in the present world scenario through an in depth understanding of these models. While there are numerous methods, for the purpose of this project we have covered only the relevant models. These methods have been widely studied, and continue to be researched.

We have managed a thorough review of Cityscapes data set and ensured the implementation through necessary split in the data set.

Semantic segmentation led models of UNet and R2Unet can be used for image segmentation purposes especially in case of urban scene segmentation. However, for rural, low light settings, occlusions and unusual objects, we have not established the efficiency as it was out of scope in this project, and can be further explored for these models.

The accuracy and efficiency parameters can be improved through continuous training and introduction of variation in the data set and transfer learning.

R2UNet as a model seems to work marginally better than UNet. Hence we see that further investigation, which was beyond the scope of this project has to be done. R2UNet as a model should be further studied and developed through better training and validation data set.

Both models while coming with a lot of advantages, also come with a significant computational cost, especially R2UNet considering its complex architecture.

The results of the experiment can be improved through further cycles.

6.2 Future work

Autonomous vehicle segment has seen a rapid improvement over the last decade and has caught the eye of the generation. This has been made possible due to the feats achieved in Deep learning.

UNet has proved to be a significant development in the field of Deep learning and has improved in scope beyond its original application for medical image segmentation. Many variations have been brought into the architecture framework such as RCNN, R2UNet, LadderNet, SegNet which prove to be significant improvements to the original framework. Autonomous vehicle providers have been using CNN in various capacities for self-driven cars (HydraNet, ChaufferNet).

Through this project we have navigated 2 models UNet and R2UNet in image segmentation. Using the learnings and further enhancement in the design principals of these models and reviewing additional possibilities to improve components of the model is an area of exploration.

In my opinion, there is still significant work required in improving R2Unet to make it a robust model for autonomous vehicles and therefore the scope for further research and improvements is greatly present. With these improvements we can aim to target better accuracy and make the model more robust and real-time.

Data is the most important aspect of any deep learning model used in autonomous vehicles. Without excellent training data materials, any model is bound to be sub-par with its performance. Hence, improving the training data quality is paramount. The algorithms written today don't take into account scenarios where there could be a deviation from the class information that is presently available. For example, inaccuracy in the markings on the road, lane segregation etc may cause an inaccurate reading of the segmented image and impact the image analysis. A major challenge in this scenario also is the amount of data that is actually collected through an autonomous vehicle is very high and hence the chances of unclassified segments increases.

Another important aspect, in any project or software is the ability to identify and solve for the corner cases that we cannot immediately predict or have insufficient knowledge of. Sufficient work needs to be done to address scenarios to enable decision making and avoid any road accidents or related events.

To summarise, areas of future work in UNet inspired models include improvements in:

- accuracy
- perception
- depth
- real time analysis
- data classification

Improvements in Deep Learning & CNN based encoder decoder techniques like UNet and R2UNet will not only improve the accuracy in identifying the objects in an input image, but also take the scope further into understanding the perception and depth of the objects in the image. The existing semantic segmentation takes into account only 2D or 3D arrangements. Further data classification & labelling, perception and depth analysis will help with real time segmentation and tracking. Improvements in these areas will allow better decision making for autonomous vehicles.

List of References

Bibliography

1. Alom, M.Z., Hasan, M., Yakopcic, C., Taha, T.M. and Asari, V.K. 2018. Recurrent Residual Convolutional Neural Network based on U-Net (R2U-Net) for medical image segmentation. arXiv [cs.CV]. [Online]. [Accessed 26 August 2023]. Available from: http://arxiv.org/abs/1802.06955.
2. Alzubaidi, L., Zhang, J., Humaidi, A.J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M.A., Al-Amidie, M. and Farhan, L. 2021. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. Journal of big data. 8(1).
3. Anon 2022. 1D Convolutional Neural Networks in TensorFlow. CityofMcLemoresville. [Online]. [Accessed 26 August 2023]. Available from: https://cityofmclemoresville.com/1d-convolutional-neural-network-tensorflow/.
4. Anon n.d. Dataset overview. Cityscapes-dataset.com. [Online]. [Accessed 26 August 2023a]. Available from: https://www.cityscapes-dataset.com/dataset-overview/.
5. Anon n.d. Wikimedia.org. [Online]. [Accessed 26 August 2023b]. Available from: https://upload.wikimedia.org/wikipedia/commons/d/d4/Image_segmentation.png.
6. Chen, Y., Li, L., Li, W., Guo, Q., Du, Z. and Xu, Z. 2023. AI computing systems: An Application Driven Perspective. Oxford, England: Morgan Kaufmann.
7. Chopra, M. and Purwar, A. 2022. Recent studies on segmentation techniques for food recognition: A survey. Archives of Computational Methods in Engineering. State of the Art Reviews. 29(2), pp.865–878.
8. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S. and Schiele, B. 2016. The Cityscapes dataset for semantic urban scene understanding. arXiv [cs.CV]. [Online]. [Accessed 26 August 2023]. Available from: http://arxiv.org/abs/1604.01685.
9. Devipriya, A., Prabakar, D., Singh, L., Oliver, A.S., Qamar, S. and Azeem, A. 2023. Machine learning-driven pedestrian detection and classification for electric vehicles: integrating Bayesian component network analysis and reinforcement region-based convolutional neural networks. Signal, image and video processing.
10. Han, P.&. T., Nisbet, M.&. Y., Buyya, C.&. V.D., Linstedt & Olschimke and Doan, H.&. I. n.d. Data mining - edition 4 - by Ian H. witten, eibe Frank, Mark A. hall and Christopher J. pal, ph.D.elsevier health inspection copies. Elsevier.com. [Online]. [Accessed 26 August 2023]. Available from: https://inspectioncopy.elsevier.com/book/details/9780128042915.
11. Jiang, B., An, X., Xu, S. and Chen, Z. 2022. Intelligent image semantic segmentation: A review through deep learning techniques for remote sensing image analysis. Journal of the Indian Society of Remote Sensing.
12. Kniazieva, Y. 2022. What is semantic segmentation in computer vision? Labelyourdata.com. [Online]. [Accessed 26 August 2023]. Available from: https://labelyourdata.com/articles/semantic-segmentation.
13. Ma, L., Liu, Y., Zhang, X., Ye, Y., Yin, G. and Johnson, B.A. 2019. Deep learning in remote sensing applications: A meta-analysis and review. ISPRS journal of photogrammetry and remote sensing: official publication of the International Society for Photogrammetry and Remote Sensing (ISPRS). 152, pp.166–177.
14. Parekh, D., Poddar, N., Rajpurkar, A., Chahal, M., Kumar, N., Joshi, G.P. and Cho, W. 2022. A review on autonomous vehicles: Progress, methods and challenges. Electronics. 11(14), p.2162.
15. Sharma, M., Kumar, A., Supriya, M., Singh, V. and Kishore, S. 2023. Machine learning in remote sensing data—a classification case study In: Atmospheric Remote Sensing. Elsevier, pp.413–428.
16. Su, Z., Li, W., Ma, Z. and Gao, R. 2022. An improved U-Net method for the semantic segmentation of remote sensing images. Applied intelligence. 52(3), pp.3276–3288.

17. Yang, C., Qin, L.-H., Xie, Y.-E. and Liao, J.-Y. 2022. Deep learning in CT image segmentation of cervical cancer: a systematic review and meta-analysis. Radiation oncology (London, England). 17(1).
18. Yu, H., Yang, Z., Tan, L., Wang, Y., Sun, W., Sun, M. and Tang, Y. 2018. Methods and datasets on semantic segmentation: A review. Neurocomputing. 304, pp.82–103.
19. Zhang, Liangpei, Zhang, Lefei and Du, B. 2016. Deep learning for remote sensing data: A technical tutorial on the state of the art. IEEE geoscience and remote sensing magazine. 4(2), pp.22–40.

Appendix A
Cityscapes Image Pairs Dataset

This code leverages the Cityscapes image pairs dataset, enabling the exploration and analysis of urban environments through advanced image processing techniques.
https://www.kaggle.com/datasets/dansbecker/cityscapes-image-pairs

The dataset contains 2975 training images and 500 validation images of vehicles driven in Germany.

Appendix B
Ethical Issues Addressed

B.1 Data Privacy

The images used in this project are sourced from Cityscapes- freely available source for academic purposes. [Cityscapes, "Cityscapes Data Collection," https:// www.cityscapes-dataset.com/, M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes Dataset for Semantic Urban Scene Understanding," in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.] & Kaggle open source data set [https://www.kaggle.com/datasets]

No personal information of any individuals have been personally requested, collected or used for the purposes of this project and hence this project is not in violation of data privacy or consent. All images part of the data source has been ethically sourced by the data set providers and have the required consent to use these images for academic purposes.

The images in the data set ensures that individuals in the images cannot be recognised and lead to a loss of anonymity.

B.2 Data Bias

The training and validation data sets have been used and partitioned through softwares without manual intervention, hence avoiding any scope for bias.

B.3 Data Transparency and Fairness

The nature of use of the data has been for purely academic fulfilment and hence has been used fairly and responsibly.

B.4 Data Sharing

Data sharing through the sources cited in this project should be for reference purpose only. If this project is referenced in the future, the individual referencing should abide by the terms and conditions of the data source provider.