University of the Witwatersrand
School of Mechanical, Industrial and Aeronautical Engineering

Semantic Segmentation for Autonomous Vehicles

Research Project
MECN4006A

Name: Tshabalala Kgaogelo
Student Number: 675111
Supervisor: John Jones

11 September 2023

Declaration

I, the undersigned, am registered for the course MECN4006A - Research Project in the year 2023. I herewith submit the following task “Semantic Segmentation for Autonomous Vehicles” in partial fulfillment of the requirements of the above course.

I declare the following:

- This report is my own, unaided work, except where otherwise acknowledged.
- It is being submitted as partial fulfilment to the course MECN4006A.
- It is being submitted for the degree of Bachelor of Science in Mechanical Engineering at the University of the Witwatersrand, Johannesburg.
- It has not been submitted before for any degree or examination at any other university.
- I are aware that plagiarism (the use of someone elses work without their permission and/ or without acknowledging the original source) is wrong.
- I understand that the University of the Witwatersrand, Johannesburg may take disciplinary action against me if it can be shown that this task is not my own unaided work, or that I have failed to acknowledge the source of the ideas or words in my writing in this task.

Name: Kgaogelo Tshabalala
Student Number: 675111
Date: 11 September 2023

Signature

Executive Summary

In this study, the performance metrics (i.e., intersection over union, accuracy, recall, precision, and fscore) of three segmentation networks were evaluated using a dataset captured under different illumination conditions (sunny and cloudy conditions) to qualitatively asses the effect of illumination on the performance metrics. In addition, the performance metrics (i.e., intersection over union, accuracy, recall, precision, and fscore) of three semantic segmentation networks were evaluated using the same dataset to qualitatively explain the relationship between perception performance and network architecture for the three semantic segmentation models (i.e., resnet152DeepLabV3Plus, resnet101DeepLabV3Plus, and resnet101Unet). The conclusions are as follows: (1) The training hyperparameters, particularly the learning rate, was only effective for the resnet101Unet segmentation model but too high for the resnet152DeepLabV3Plus and resnet101DeepLabV3Plus semantic segmentation models. (2) The performance of the resnet101UNet segmentation model was not generalizable for all the training epochs considered (i.e., only the results for epochs 4, 8, and 10 were deemed acceptable), whereas the perfomance of the resnet152DeepLabV3Plus and resnet101DeepLabV3Plus models was generalizable for all epochs considered. (3) The execution time was longest for the resnet152DeepLabV3Plus semantic segmentation architecture (7–8 s) and lowest for the resnet101DeepLabV3Plus and resnet101UNet architectures (5–6 s). That is, only changing the encoder of the architecture affected the execution time. This is attributed to the greater number layers (152 layers) in ResnNet152 compared to those in ResNet101 (101 layers). The increased parameters in the former encoder results in an increased execution time. (4) resnet101UNet architecture exhibits decreased performance under sunny conditions compared to cloudy conditions, whereas resnet152DeepLabV3Plus and resnet101DeepLabV3Plus exhibit consistent performance. (5) resnet152DeepLabV3Plus and resnet101DeepLabV3Plus architectures generally exhibit superior (i.e. IOU, precision, recall, fscore, and dice loss) and consistent performance compared to the resnet101UNet architecture under sunny and cloudy conditions. This difference in performance is attributed to the superior DeepLabV3Plus architecture (i.e., atrous spatial pyramid pooling and atrous convolutions) in comparison to the UNet architecture. To improve the predicted semantic segmentation maps and performace metrics obtained in this study, it is recommended to use a lower learning rate for the resnet152DeepLabV3Plus and resnet101DeepLabV3Plus, increase the number of images used during training, use higher resolution images, use images with higher color depths, use images with lossless image formats, and apply donoising techniques to the images. This study provides providing a reference for future semantic segmentation applications of encoder-decoder architectures under varying illumination conditions.

Contents

1 Background **1**

2 Literature Review **2**

2.1 Digital Images 2
2.2 Segmentation Architectures 3
2.3 Artificial Neural Network 4
2.4 Convolutional Neural Network 6
2.5 Encoder-Decoder Semantic Segmentation Architectures 12
  2.5.1 DeepLabV3Plus 14
  2.5.2 Unet 16
  2.5.3 Resnet 17
2.6 Metrics for Semantic Segmentation 18
2.7 Effect of Illumination 20
2.8 Considerations for Training a Semantic Segmentation Algorithm 21

3 Significance of the study **24**

3.1 Objectives 24

4 Methods and Apparatus **25**

5 Results and Discussion **30**

5.1 Training and Validation 30
5.2 Testing 32

6 Conclusions and Recommendations **40**

List of Figures

1 Semantic segmentation example [1] 2
2 Artificial neuron [11] 4
3 Artificial neural network [13] 5
4 Sigmoid activation function [12] 6
5 Convolutional neural network [9] 7
6 Example of color image (RGB) [36] 7
7 Example of a filter [7] 8
8 Example of convolution operation [7] 8
9 Padding operation [7] 9
10 Stridding operation [7] 9
11 Max and average pooling [7] 10
12 Features extracted by filters in the various convolutional layers of the CNN [7] 11
13 Dilation example [41] 14
14 DeeplabV3Plus architecture [48] 15
15 Unet architecture [34] 16
16 Demonstration of intersection over union [20] 19
17 Driving scene images 25
18 (a) Original image and pixels corresponding to (b) unlabelled (c) sky, (d) road, (e) cones, (f) white road barricade, and (g) orange road barricade (f). Labels for each category shaded in yellow 26
19 Augmentation techniques used on the training dataset 28
20 Metrics monitored during testing and validation 31
21 Metrics monitored during the testing stage of the study 34
22 Predicted segmentation map after 2 epochs 37
23 Predicted segmentation map after 6 epochs 38
24 Predicted segmentation map after 10 epochs 39

List of Tables

1 Models implemented in the study [17] 29

2 Testing times for architectures considered: frame 0850 sunny and 0850 cloudy 32

1 Background

Semantic segmentation is a machine learning technique where a digital image is labelled at the pixel level to identifying and classify objects in a scene, such as road signs, pedestrians, and road markings in the context of automotive vehicles, thereby enabling a computer to interpret the content in the image [27]. Digital images are created using devices such as digital cameras, scanners, or graphic design software. These devices capture or create images by measuring the intensity and color of light at various points in the scene and converting that information into digital data [16]. The resulting digital file contains numerical values that represent the color and intensity of each pixel in the image [15]. The goal of semantic segmentation is to group pixels according to the class/object they belong to from an input image. After an image is input into a semantic segmentation algorithm, the algorithm should be able to delineate the different objects in the image, as shown in Figure 1. segmentation is the foundation for tasks such as object detection and scene understanding in autonomous vehicles and utilizes different algorithms, which rely on the texture and color information to identify objects and regions of interest [27]. Consequently, the performance of semantic segmentation algorithms is affected by factors such as rain, snow, fog, and illumination [14]. Semantic segmentation algorithms are commonly evaluated based on two criteria: accuracy, that is, the success of the algorithm in computing the segmentation task, and computation complexity, that is memory requirements and speed. Overall, a good semantic segmentation algorithm should achieve high scores in both metrics [43]. Changes in illumination can affect the accuracy and robustness of semantic segmentation algorithms, as changes illumination alters texture and color of objects and regions [14]. Therefore, the investigation of the perception performance metrics of semantic segmentation algorithms under different illumination conditions is necessary to assess the applicability of machine vision for autonomous-vehicle applications. The architecture of a semantic segmentation network can have a significant impact on its performance. This is because different architectures have different strengths and weaknesses and are optimized for different types of tasks [2]. Therefore, investigating the effect of semantic-segmentation-algorithm architecture on the performance metrics is important for classifying semantic segmentation networks according to application suitability. This paper is structured as follows. Sections 2.1 to 2.6 highlight concepts related to semantic segmentation. Section 3 highlights the significance of the study and the research objectives. Section 4 explains the methods and apparatus used to meet the research objectives. Section 5 is a discussion of the results in line with the research objectives. Lastly, Section 6 concludes the study and highlights recommendations.

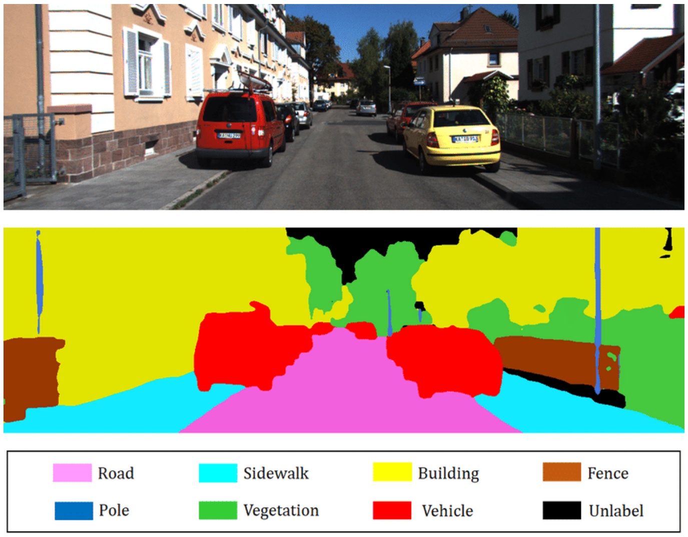


Figure 1: Semantic segmentation example [1]

2 Literature Review

Computer vision, a branch of computer science, is concerned with the understanding and identification of people and objects from videos and images [18]. Computer vision techniques can be broadly categorized into classification, object tracking, object detection, instance segmentation, and semantic segmentation, which is the focus of this study, and instance segmentation [22]. Before the signifin concepts related to digital are images are explained to highlight how they can be converted into meaning information for autonomous vehicle applications.

2.1 Digital Images

Digital images are visual representations of objects, scenes, or designs that are stored and transmitted in digital format. They consist of a grid of picture elements, commonly known as pixels, which are tiny squares of color that collectively form the image. Digital images are created using devices such as digital cameras, scanners, or graphic design software. These devices capture or create images by measuring the intensity and color of light at various points in the scene and converting that information into digital data [16]. The resulting digital file contains numerical values that represent the color and intensity of each pixel in the image [15]. Several properties of digital images can significantly impact image processing techniques and algorithms, including image resolution, color depth, image noise, image compression, image color space, image orientation, sharpness, contrast, brightness, distortion, chromatic aberration, and dynamic range [21]. Before these factors are explained further, some of the important concepts for semantic segmentation are explained [33, 45].

2.2 Segmentation Architectures

Image segmentation is a process that accepts images as inputs and generates an outcome, typically in the form of a mask, as shown Figure 1, or a matrix. This result contains diverse elements indicating the category or instance to which each pixel within the image belongs. Several relevant heuristics, or high-level image features, can prove valuable in the context of image segmentation. These heuristics serve as the foundation for established image segmentation algorithms, which employ clustering methods such as edge detection and histogram analysis [10, 42, 5]. Conventional image segmentation techniques, such as thresholding, clustering, and region growing, that rely on these rules of thumb are often rapid and uncomplicated [10]. However, they often demand substantial fine-tuning to accommodate specific use cases [10, 42, 5]. Moreover, their accuracy may not always be adequate for intricate images [10, 42, 5].

Modern approaches to image segmentation, particularly those based on deep learning, such as convolutional neural networks (CNNs), have significantly outperformed traditional semantic segmentation techniques in terms of both performance and adaptability [42]. These newer segmentation methods leverage machine learning and deep learning to enhance accuracy and flexibility. Machine learning-based image segmentation methods involve training models to improve their capability to recognize crucial features. Deep neural networks excel in image segmentation tasks [42]. Furthermore, there is a wide array of neural network architectures and implementations suitable for image segmentation [42]. Therefore, the primary focus of this study will be on semantic segmentation networks that are based on deep learning, particularly encoder-decoder architectures. However, before delving into deep neural networks, several fundamental concepts that are central to these types of networks are discussed in Sections 2.3–2.4.

2.3 Artificial Neural Network

Artificial neural networks (ANNs), as depicted in Figure 2, comprise several tiers of nodes, encompassing an initial layer for input, one or more intermediate hidden layers, and an output layer [11]. These nodes, also known as artificial neurons, are interconnected and characterized by their weights and thresholds. When the output of a node exceeds the predefined threshold, it becomes activated and transmits information to the subsequent layer of the network. Conversely, if the output falls below the threshold, no data is forwarded to the following layer [46]. The connection weight represents the parameter in a neural network responsible for modifying input data as it traverses through the hidden layers of the network. Within every node, there exists a collection of inputs, weights, and a bias value. When an input enters a node, it undergoes multiplication by a weight value, and the resultant output is either observed or forwarded to the subsequent layer within the neural network [46].

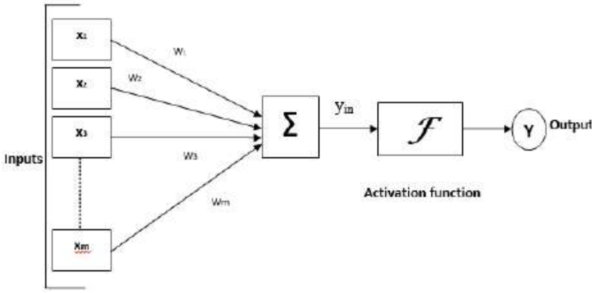


Figure 2: Artificial neuron [11]

Frequently, the neural network's weights are situated within its hidden layers, as shown in Figure 3. Both weights and biases are adjustable parameters within the network. In a trainable neural network, the initial stages involve randomizing both weight and bias values. As the training progresses, these parameters are fine-tuned towards the desired values and correct output. The distinction between these two parameters lies in their impact on input data. In simple terms, bias gauges the disparity between predictions and their intended values [11]. Biases bridge the gap between the function's output and its intended target. A minor bias implies that the network assumes more about the output's form, while a substantial bias value indicates fewer assumptions about the output's form. Conversely, weights can be likened to connection strength. Weight determines the extent to which a change in input affects the output. A smaller weight value has negligible impact on the input, whereas a larger weight value significantly alters the output. Subsequently, these weighted inputs are aggregated, and an activation function is applied to compute the neuron's input [11].

Neural networks have an input layer, output layer, and hidden layers, as shown in Figure 3. In the neuron of each layer, the input is multiplied by a weight and, in some instances, a bias is added to the product. This is a linear transformation, which is acceptable if the problem can be predicted by a linear model [6]. In cases that involve complex patterns, multiple hidden layers are necessary. However, multiplying a weight by a bias is not sufficient to model the prediction of complex patterns and, thus, requires an activation function, which applies a nonlinear transformation, thereby allowing the model to make more complex predictions [6].

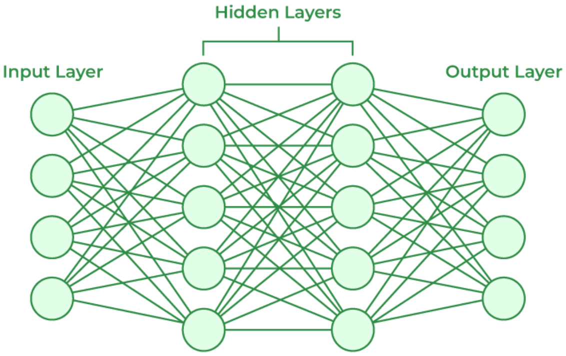


Figure 3: Artificial neural network [13]

Activation functions can be linear or nonlinear. Their function is to apply a transformation and determine whether a neuron should be activated or not [30]. Types of activation functions include sigmoid activation function, hyperbolic tangent function, rectified linear unit (ReLu), radial basis function, hard limiter function piecewise linear function, and linear function. The application of activation functions is illustrated by using a sigmoid activation function in Figure 4. When the input to an activation function, consisting of the product of a weight and input to the artificial neuron, is positive, the output of the activation function approaches the threshold of the neuron. Thus, the neuron will be activated for inputs that are sufficiently large [30]. In the context of semantic segmentation, the product of the weight and artificial neuron will be high if the pattern of the input is similar to the pattern that the segmentation model is trained to recognise. This is explained further in Section 2.4 using the concept of filters and convolution. Neural network models come in various forms, such as feedforward artificial neural networks, perceptrons, multilayer perceptron neural networks, convolutional neural networks (CNNs), radial basis function artificial neural networks for time series prediction, recurrent neural networks, and modular neural networks [29]. CNNs, owing to their significance for semantic segmentation applications, are subsequently explained.

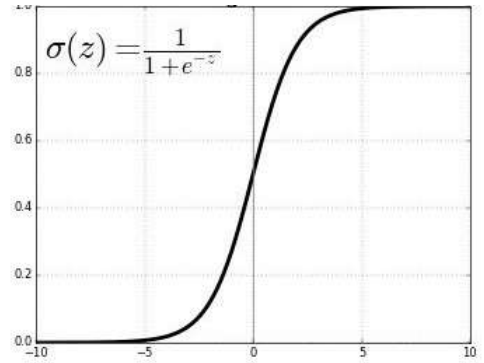


Figure 4: Sigmoid activation function [12]

2.4 Convolutional Neural Network

A basic CNN can be understood as a series of layers, as shown in Figure 5, with each layer aiming to transform a set of activation volumes into new volumes using a differentiable function (i.e., activation function) [9]. Essentially, this implies that every layer works to convert information from preceding layers' values into more intricate data, which is then relayed to subsequent layers for additional abstraction. This abstraction process is presented in Figure 12. The convolution block encompasses both the convolution layer and the pooling layer, representing a critical element in extracting features [9]. The fully connected block entails a straightforward architecture of a neural network with interconnected nodes. Its role primarily involves classifying data, drawing from input provided by the convolutional block [9].

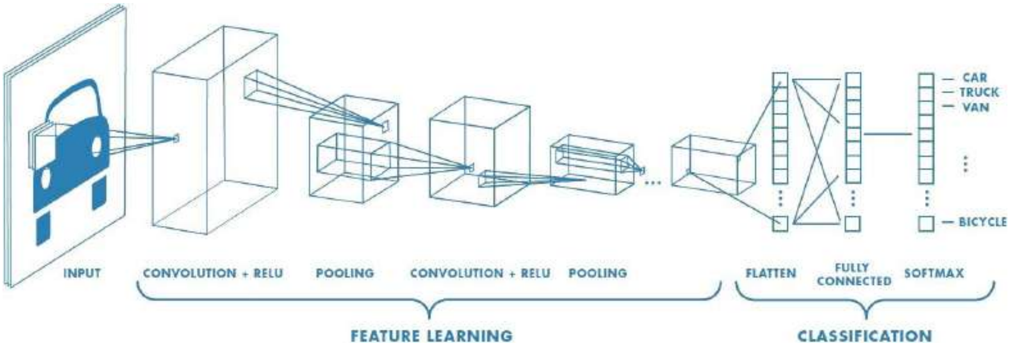


Figure 5: Convolutional neural network [9]

First, the concepts of 'filters' and 'convolution' are explained, followed by their implementation in the layers of CNNs. Grayscale images can be depicted using a single matrix that represents the brightness level, as illustrated in 21. In the case of a colored image, this can be expressed as a combination of three such matrices, with each one signifying the red, green, and blue intensity, respectively, as displayed in Figure 6[36]. We can extract valuable insights and uncover relationships in images by adjusting these parameters. In the context of CNNs, filters and convolution operations are employed to capture this valuable information. For simplication purposes, greyscale images are using to explain concepts that follow. Extensions of these concepts to 2D RGB images are highlighted where appropriate. Filters or 'kernels' are also images that depicts a particular feature. As an example, the picture of a curve and its filter for a grey scale image are shown in Figures 10b and 7a, respectively. Subsequently, the process of recognizing a feature (i.e., determine its presence in an image) in an input image is explained using the concept of convolution.

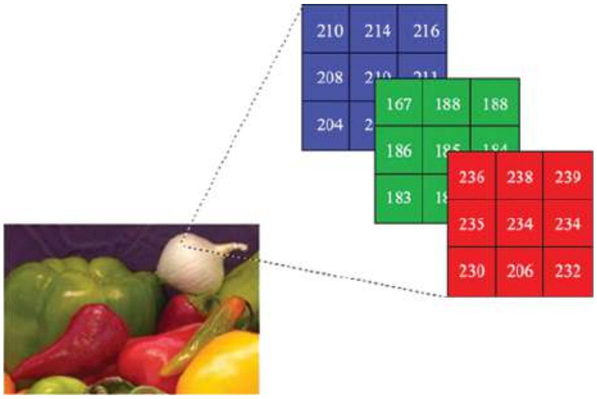
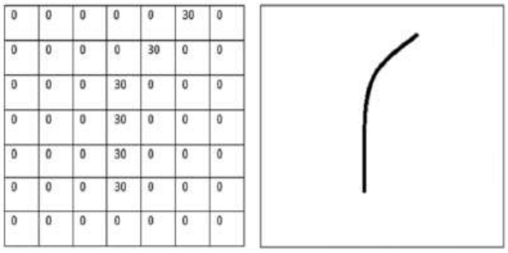


Figure 6: Example of color image (RGB) [36]

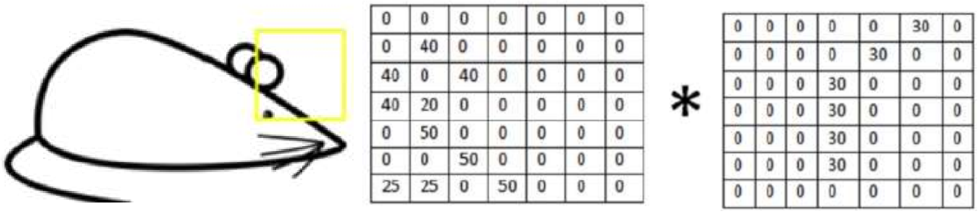


0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

(a) Pixel representation of filter (b) Visual representation of filter

Figure 7: Example of a filter [7]

Convolution is a process performed on a specific matrix, typically the image matrix, by utilizing another matrix, often referred to as the filter matrix [7]. This process entails the multiplication of the cell values at specific rows and columns within the image matrix with the values of corresponding cells in the filter matrix. This is done for the values of all the cells within the span of the filter matrix, and the results are added together to form an output. If this filter is convoluted with matrices representing different areas of the mouse, as shown in Figure 8, the output of the convolution will be largest for the image matrix section that contains the feature of interest [7]. Thus, this value is passed by the activation function to the next layer of the network. When dealing with a colored image, represented as a 2D RGB image matrix, there exits three filter matrices, which are 2D matrices representing certain features [3]. In this process, convolution is applied independently over the corresponding regions of the image matrix using each filter matrix for all the red, green, and blue color channels [3]. Subsequently, the values obtained from each channel are added to determine the value of the cell in the output matrix. Essentially, this operation identifies whether a specific feature is present in the image that we are trying to recognize, as previously explained [3]. Convolution involves a few other essential operations, namely padding, stridding, and pooling, which are subsequently explained.



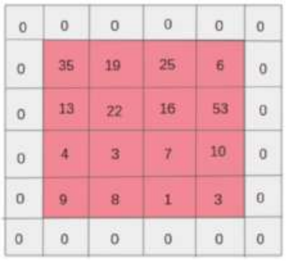
0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

(a) Visualization of filter on greyscale image (b) Convolution operation

Figure 8: Example of convolution operation [7]

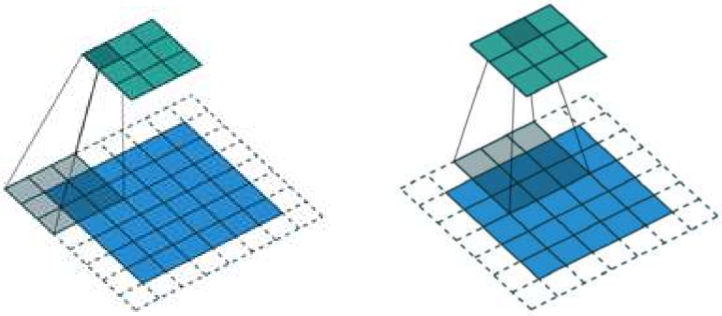
During the convolution operation, the filter matrix traverses the entire image. However, the cells' values within the filter matrix are considered more times than the cells near the corners or borders of the image. This means that the values around the image's edges are not given equal importance [7]. To address this imbalance, an additional row and column filled with zeros are added to all sides of the image matrix, as shown in Figure 9. This technique is known as padding. In practice, these added '0' values do not provide any additional information, but they help ensure that the previously underemphasized values near the edges are given more weight in the convolution process [7].



0	0	0	0	0	0
0	35	19	25	6	0
0	13	22	16	53	0
0	4	3	7	10	0
0	9	8	1	3	0
0	0	0	0	0	0

Figure 9: Padding operation [7]

The following operation is striding. In "strided" convolution, instead of moving the filter one row or one column at a time, it is shifted by a greater distance, such as 2 or 3 rows or columns at each step, as shown in Figure 10. This is typically done to decrease the number of computations and the size of the output matrix. When dealing with large images, this approach does not lead to data loss but significantly reduces the computational cost [7].



(a) First stride

(b) Second stride

Figure 10: Stridding operation [7]

The operation of a CNN consists of a convolution operation, which incorporates a stridding operation. The convolution product becomes an input for the activation function. These processes form the convolution block of the CNN. The basic intuition to derive for the activation function is that, after convolution, if a particular convolution function results in '0' or a negative value when processed by the activation function, the feature the filter is trained to recognize is not present in the section of the input image represented by the pixel matrix [7].

The pooling layer carries out the task of selecting a specific value from a set of values, typically the maximum or average value from the set. This action results in a reduction in the dimensions of the output matrix. To illustrate, in the case of max-pooling, the maximum value within a 2 × 2 section of the matrix extracted. This allows us to isolate the values indicating the existence of a feature in that segment of the image, thus eliminating extraneous information related to feature presence in a specific portion of the image and focusing solely on essential data. The pooling layer undertakes the task of extracting a specific value from a collection of values, typically either the maximum or average value among all the values [7]. This operation reduces the dimensions of the resulting matrix. To illustrate, in the case of max-pooling, the highest value within a 2 × 2 segment of the matrix is utilized, as depicted in Figure 11. Consequently, this process captures the information signifying the existence of a feature in that region of the image, effectively eliminating extraneous details pertaining to feature presence in a specific part of the image, and focusing solely on essential data. In CNN architectures typically incorporate a pooling layer between consecutive convolutional blocks [7]. This layer serves the purpose of gradually decreasing the spatial dimensions of the representation, thereby reducing the volume of parameters and computational load within the network. Figure 11 shows an example of average and max pooling after a convolution operation, where a max pooling takes the maximum value of a portion of the convolution product, while average pooling takes the average value takes the maximum value of a portion of the convolution product.

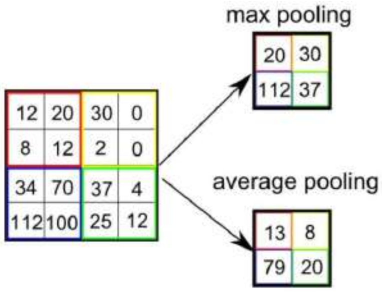


Figure 11: Max and average pooling [7]

Now that the concepts related to the operation of a CNN (i.e., artificial neurons, artificial neural network, activation function, convolution, filters, pooling, and stridding) have been explained, the operation of a CNN can be summarized. Fundamentally, the important concept to grasp is that, during training, the filters at different layers undergo adjustments to identify distinct image features. The filters in the initial layers capture fundamental elements like edges or basic shapes. As we progress to the next layer, the filters start capturing more complex textures. In the deeper layers, the filters become capable of recognizing specific components that together form an object, as illustrated in Figure 12 [7]. Ultimately, the fully connected layer, comprising a neural network, is responsible for categorizing the image into different object classes or categories [7].

A CNN model can be conceptualized as a fusion of two key components: a feature extraction segment and a classification segment. The convolution and pooling layers primarily handle feature extraction tasks. For instance, when presented with an image, the convolution layer identifies features like pairs of eyes, elongated ears, quadrupedal limbs, a concise tail, and so forth. Subsequently, the fully connected layer assumes the role of a classifier for these identified features and assigns a probability indicating whether the input image corresponds to a dog. The convolution layers serve as the primary workhorse within a CNN model. They progressively learn intricate features by building upon each other. In the initial layers, they discern basic edges, while in subsequent layers, these edges are combined to recognize shapes. Further layers then consolidate this information to deduce more complex features, such as a nose. Through exposure to numerous instances of these features in images, the model becomes adept at identifying them [7]. The fully connected layers, in turn, learn how to effectively utilize these features extracted by the convolutions to make accurate classifications within the specified classes or categories in images. Now that the operation of a CNN has been explained, encoder-decoder architectures, which utilize CNNs for semantic segmentation tasks, will be explained in Section 2.5.

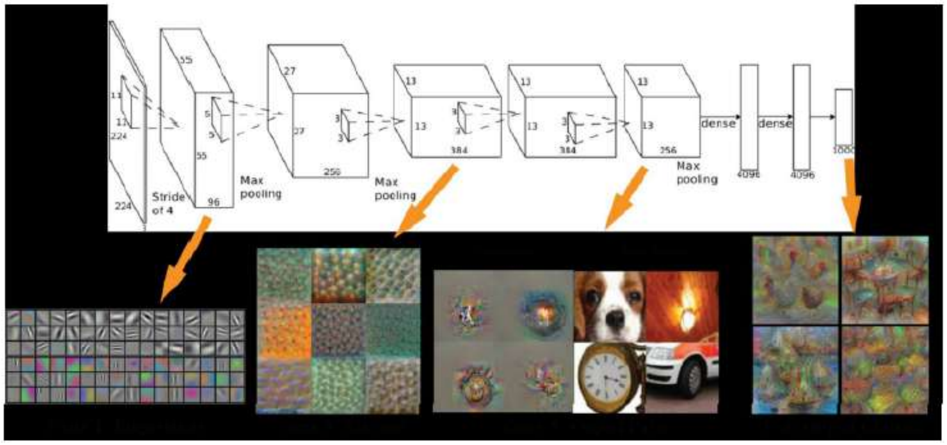


Figure 12: Features extracted by filters in the various convolutional layers of the CNN [7]

2.5 Encoder-Decoder Semantic Segmentation Architectures

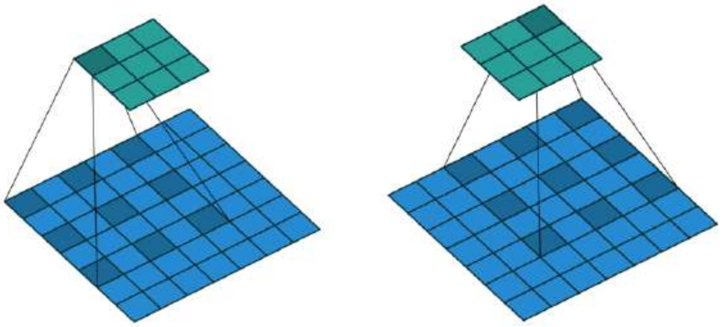
Encoder-decoder semantic segmentation networks are a class of deep learning models designed for pixel-wise semantic segmentation tasks in computer vision. They consist of two main components: an encoder network and a decoder network. The encoder network is typically a convolutional neural network (CNN) that processes the input image and gradually reduces its spatial dimensions while extracting high-level features [47, 5, 51]. It consists of several convolutional layers followed by pooling or strided convolutions to downsample the feature maps (i.e., the output maps generated by a neural network during the process of semantic segmentation). This downsampling helps in capturing abstract and invariant features at multiple scales. The encoder network's purpose is to encode the input image into a lower-dimensional feature representation that preserves semantic information [47, 5, 51]. The decoder network takes the encoded feature representation from the encoder and aims to recover the original spatial resolution of the input image [47, 5, 51]. It performs upsampling or transposed convolutions to progressively increase the feature map size. Furthermore, the decoder network also incorporates skip connections, which connect corresponding layers from the encoder to the decoder. The function of these skip connections is to combine low-level and high-level features, enabling the model to leverage both local and global context information [47, 5, 51]. The decoder network gradually refines the feature maps, producing pixel-wise predictions. The encoder-decoder architecture allows the model to capture both local and global information, enabling accurate segmentation. Furthermore, it captures high-level semantic features through downscaling, while the decoder uses upsampling and skip connections to recover spatial details and generate dense predictions for each pixel in the input image [47, 5, 51].

To train encoder-decoder segmentation networks, pixel-level annotations are required, where each pixel in the training images is labeled with a specific class/category. This is explained further in Section **??**. During training, the network learns to minimize the semantic segmentation loss, typically computed using metrics such as dice loss, intersection over union, accuracy, fscore, recall, and precision (explained in Section 2.6 , which measure the similarity between the predicted segmentation map and the ground truth annotations (i.e., manually created or verified pixel-level labels) [47, 5, 51]. Encoder-decoder architectures have gained significant popularity in the field of semantic segmentation due to several advantages they offer, including (1) semantic understanding, (2) spatial resolution preservation, (3) flexibility, (4) multi-scale features, (4) efficiency, (4) transfer learning, (6) state-of-the-art performance, (7) real-time applications, and (8) semantic features [47, 39].

(1) Encoder-decoder architectures excel at capturing semantic information within images. The encoder part learns to extract hierarchical features from input images during the training stage, effectively encoding information about object boundaries, textures, and context. Subsequently, the decoder uses this encoded information to generate pixel-wise semantic labels [19]. (2) These architectures maintain the spatial resolution of the input image throughout the processing. By using skip connections that transfer information from the encoder to the decoder, they allow for precise localization of objects in the segmentation map [19]. (3) Encoder-decoder architectures are highly adaptable and can be used for a wide range of image sizes and tasks. They can handle both scene-level and pixel-level segmentation, making them versatile for various computer vision applications. (4) The encoder extracts features at different levels of abstraction, while the decoder combines them to generate pixel-wise predictions. This enables the network to handle objects of various sizes within the image [19]. (5) Despite their depth, many encoder-decoder architectures are designed to be computationally efficient. Techniques like dilated convolutions and depth-wise separable convolutions are often used to reduce the computational load while maintaining high performance [19]. (6) Encoder-decoder architectures benefit greatly from pre-trained models on large datasets (e.g., ImageNet). Transfer learning allows fine-tuning on smaller, task-specific datasets, which can significantly boost performance, especially when labeled data is limited [19]. The concept of trafnsfer learning is futher expplained in Section **??** (7) Encoder-decoder architectures have consistently achieved state-of-the-art results in semantic segmentation and have become a benchmark for comparison [19]. (8) Moreover, many encoder-decoder architectures can be optimized for real-time applications. This is crucial for tasks like autonomous driving and robotics, where low-latency segmentation results are essential [19]. (9) These architectures are adept at capturing high-level semantic features, making them suitable for tasks that require understanding the content of an image, such as object detection and scene parsing [19]. Sections 2.5.1-2.5.3 highlight the encoder-decoder architectures utilized in this study.

2.5.1 DeepLabV3Plus

The first architecture to be considered is DeepLab, which is a convolutional neural network (CNN) architecture designed for semantic image segmentation. It was developed by researchers at Google for the purpose of accurately labeling every pixel in an image with the corresponding class label [25]. It achieves this by utilizing atrous (dilated) convolutions, which allow for larger receptive fields without increasing the number of parameters [8]. Atrous convolutions are a type of convolutional operation used in deep learning models, particularly in the field of computer vision. They were originally introduced to effectively capture multi-scale information while maintaining a large receptive field. In a standard convolutional operation, a kernel (filter), explained in Section 2.4, slides over the input image or feature map with a fixed stride, typically 1. Each element of the kernel interacts with the corresponding elements in the input to produce an output feature map. The receptive field of a convolutional layer is determined by the kernel size and stride. In contrast, atrous convolutions introduce the concept of "dilation," which refers to the spacing between the values of the kernel elements, to increase the receptive field without increasing the number of parameters or computations significantly. Instead of directly sliding the kernel, the kernel is dilated by inserting zeros between its values, creating gaps or holes, as shown in Figure 13. By controlling the dilation rate, which determines the spacing between the values, atrous convolutions allow for the integration of larger context information. A larger dilation rate increases the receptive field of the convolutional operation. This is particularly useful for capturing multi-scale information, as it enables the network to learn features at different scales simultaneously. The advantage of using atrous convolutions is that they can capture fine-grained details and contextual information from a larger region while keeping the computational cost relatively low [8]. This makes them useful in tasks such as image segmentation, where capturing both local details and global context is crucial. The architecture also includes an encoder-decoder structure, where the encoder captures high-level semantic information and the decoder recovers the spatial resolution of the output.



(a) First stride

(b) Second stride

Figure 13: Dilation example [41]

There have been several versions of DeepLab developed over the years, each with its own improvements and modifications. DeepLabv3, shown in Figure 14, for example, introduced atrous spatial pyramid pooling (ASPP) to capture multi-scale information and better handle objects at different scales. The ASPP module serves the purpose of gathering context information at multiple scales [48]. The final predictions are generated through an up-sampling process. Within the ASPP network, in addition to the feature map derived from the backbone, four separate atrous convolutions are employed in parallel, each with distinct atrous rates, to effectively segment objects at various scales. To incorporate global context information, image-level features are integrated by applying global average pooling to the last feature map of the backbone [48]. ASPP helps to increase the field of view for extracting multi-scale features. It is used to consider objects at different scales and segment with much improved accuracy [48].

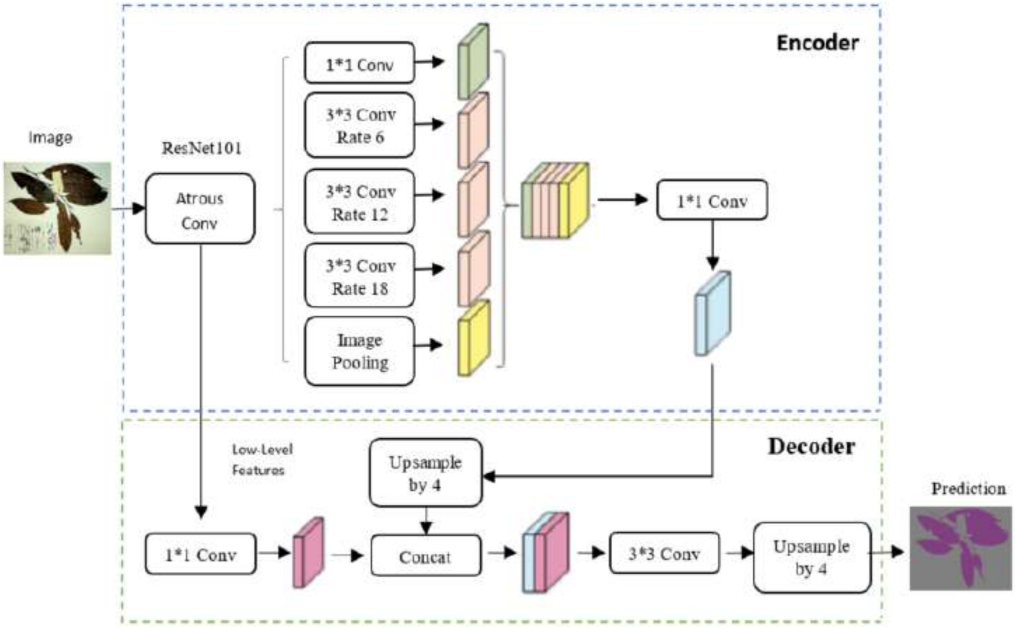


Figure 14: DeeplabV3Plus architecture [48]

2.5.2 Unet

The UNet architecture is a popular architecture designed for semantic segmentation tasks. It is particularly well-suited for tasks where precise pixel-wise segmentation is required. Similar to Deeplab, it follows an encoder-decoder structure, where the encoder is responsible for capturing contextual information from the input image, while the decoder is responsible for generating pixel-wise segmentation maps. The encoder consists of multiple convolutional layers followed by pooling layers. These layers progressively reduce the spatial resolution while increasing the number of feature channels, as explained in Section 2.4. Similar to Deeplab, this process effectively captures hierarchical features from the input image. The decoder is the counterpart of the encoder and is responsible for upsampling the feature maps to size of the original input image. It uses transposed convolutions (also known as deconvolutions or upsampling layers) to increase the spatial resolution. In addition, UNet uses skip connections that connect corresponding layers in the encoder and decoder [23]. These connections allow the decoder to access high-resolution features from the encoder, facilitating precise localization of objects in the segmentation map. Skip connections are typically concatenated or added element-wise [23]. The final layer of the decoder typically uses a convolutional layer with a softmax activation function to generate pixel-wise class probabilities [23]. This results in a segmentation map where each pixel is assigned a class label or probability distribution over classes. The contracting (encoder) and expansive (decoder) paths have a symmetric architecture, which helps in retaining spatial information. UNet typically has more feature channels in its encoder portion, which helps it capture fine-grained details in the input image [23].

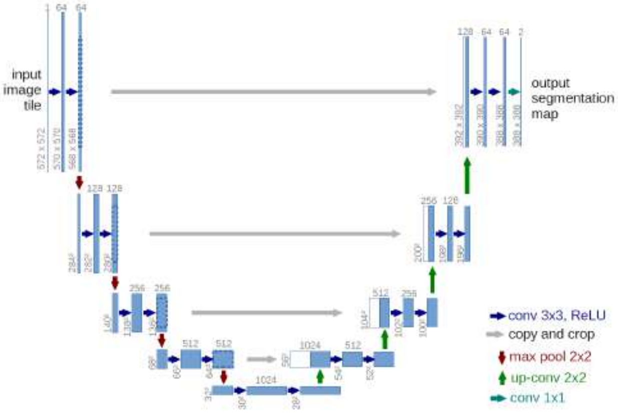


Figure 15: Unet architecture [34]

The UNet architecture offers several advantages, including (1) precise pixel segmentation, (2) skip connections, (3) symmetric structure, (4) efficient use of data, (5) adaptability, (6) state-of-the-art performance. These are subsequently discussed [34]. (1) UNet is designed for precise pixel-wise segmentation tasks, making it highly effective in applications where detailed object delineation is crucial, such as medical image segmentation and object detection [34]. (2) The use of skip connections allows UNet to capture both high-level context and fine-grained details from the input image, leading to accurate segmentation results [34]. (3) The symmetric encoder-decoder structure helps in preserving spatial information, which is essential for tasks like image segmentation [34]. (4) UNet can perform well even with limited labeled data due to its ability to transfer knowledge from pre-trained models through transfer learning [34]. (5) Researchers and practitioners can adapt and modify the UNet architecture to suit various domains and tasks. This adaptability has made it a go-to choice for many segmentation problems [34]. (6) UNet has consistently achieved state-of-the-art results in many segmentation benchmarks and competitions, demonstrating its effectiveness [34]. However, UNet has certain limitations, including (1) lack of global context , (2) boundary artifacts, (3) limited scalability, (4) memory and computational requirements, and (5) need for large datasets [34]. These are subsequently discussed. Moreover, contrast to DeeplabV3plus, (1) while UNet is excellent at capturing local context, it may struggle with capturing extensive global context information. This limitation can affect its performance on tasks where long-range dependencies are crucial [34]. In addition, (2) UNet can sometimes produce artifacts along object boundaries, particularly when objects are closely spaced or overlapping. This may require post-processing steps to refine segmentation maps [34]. (3) The Unet architecture is designed for relatively small input sizes, which can be a limitation in tasks requiring high-resolution image segmentation [34]. (4) Training and deploying deep U-Net models can be memory-intensive and computationally demanding, which may limit their use on resource-constrained devices [34]. (5) While U-Net can perform well with limited data thanks to transfer learning, it may still require substantial amounts of labeled data for optimal performance, especially in domains with significant variability [34].

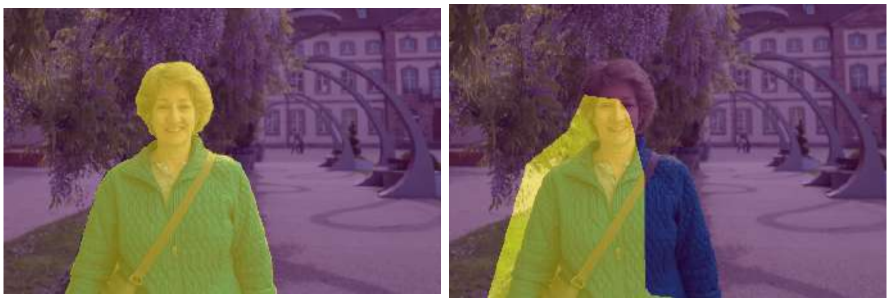
2.5.3 Resnet

Residual Network (ResNet) is a deep convolutional neural network architecture that has been widely used for various computer vision tasks, including semantic segmentation. While ResNet was initially designed for image classification, it can be adapted for segmentation tasks with a few modifications. This modification involves incorporating the ResNet into an encoder-decoder architecture, where the encoder part consists of the ResNet backbone, which extracts features from the input image, and the decoder part generates pixel-wise segmentation predictions [50]. In this study, decoders used include UNet and DeeplabV3Plus. The initial layers of the ResNet backbone (usually up to the final pooling layer) serve as the encoder. They capture hierarchical features from the input image. Furthermore, to facilitate precise object localization and to recover spatial information, skip connections are incorporated in the corresponding layers in the encoder and decoder, as explained in explained in Section 2.5 [50]. ResNet152 is a deep convolutional neural network architecture that is part of the ResNet (Residual Network) family. It is popular for its exceptional depth, making it one of the deepest architectures at the time of its introduction. ResNet101 and ResNet152 share many similarities, including the basic building block of residual connections, but differ mainly in terms of depth, model size, and capacity [50, 49, 31] . Some of the factors that distinguish ResNet152 from ResNet101 are subsequently highlighted.

ResNet101 consists of 101 layers, including convolutional and fully connected layers, whereas ResNet152 is deeper and consists of 152 layers. It includes more convolutional layers and building blocks. ResNet101 typically has fewer parameters compared to ResNet152 due to its smaller depth. Thus, ResNet152 has more parameters, which can make it more powerful but also requires more computational resources [50, 49, 31]. Due to its smaller number of layers and parameters, ResNet101 has a smaller model size, making it more memory-efficient and faster to train compared to ResNet152 [50, 49, 31]. ResNet152 has a larger model size, which can be a drawback in terms of memory and training time; however, it may offer better performance on more complex tasks or larger datasets [49, 31]. ResNet101 is often used as a good balance between performance and computational resources. It is suitable for a wide range of computer vision tasks and provides excellent results [50]. However, ResNet152, with its increased depth and capacity, may achieve slightly better performance compared to ResNet101, especially on very challenging tasks or large datasets [50, 49, 31]. However, the difference in performance may not always be significant [50, 49, 31]. Section 2.6 highlights some of the common performance metrics used to asses the performance of semantic segmentation algorithms.

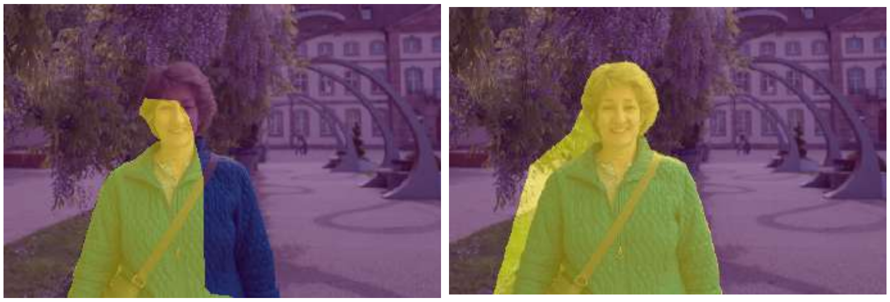
2.6 Metrics for Semantic Segmentation

Semantic segmentation algorithms are commonly assessed using performance metrics such as intersection over union (IoU), accuracy, recall, precision, and fscore. IoU is particularly useful for assessing how well a neural network or other segmentation algorithm performs in delineating object boundaries and classifying pixels [40, 28]. Before the equation for calculating the IoU is presented, intersection and union are explained in the context of semantic segmentation. Using the ground truth (i.e., the manually annotated and accurately labeled segmentation map or mask that serves as the reference) and predicted segmentation maps (i.e., visual representations of an image, where each pixel is assigned a label or class based on the object or category it belongs to) in Figures 16a and 16b, respectively, as an example, considering the pixels representing the person as the category of interest, the intersection is where the ground truth pixels overlap with the predicted segmentation map, as shown in Figure 16c [40, 28]. The union is the region enclosed by the combination of the ground-truth map and predicted map of the person, less the intersection pixels, as shown in Figure 16d [40, 28]. The IOU is the ratio of the intersection to the union, ranges between 0 and 1, and is calculated using Equation 1 [40, 28].



(a) Ground truth

(b) Predicted segmentation map



(c) Intersection

(d) Union

Figure 16: Demonstration of intersection over union [20]

$$IOU = \frac{Intersection}{Union} \tag{1}$$

The second metric considered is the pixel accuracy, also called the global accuracy, which is a calculation of the ratio of the number of correctly segmented pixels (True positives) in the predicted segmentation map to sum of the total number of correctly predicted pixels and incorrectly predicted pixels (False positives) in the ground truth [40, 28]. Accuracy is calculated as shown in Equation 2 [40, 28].

$$Accuracy = \frac{True\, positives\, for\, all\, categories}{total\, number\, of\, pixels\, in\, the\, image} \tag{2}$$

Precision, in the context of semantic segmentation, refers to a performance metric that evaluates the accuracy of the positive predictions made by a segmentation model [40, 28]. It specifically measures the ratio of true positive predictions to the total number of positive predictions made by the model. True positives are the correctly predicted positive instances, while false positives are instances that were predicted as positive but are actually negative. Precision is calculated as follows [40, 28]:

$$Precision = \frac{True\, positives}{True\, positives + False\, positives} \tag{3}$$

Recall, also known as sensitivity or true positive rate, focuses on the model's ability to capture all relevant positive instances for a particular category/class from the entire set of actual positive instances [40, 28]. It measures the proportion of true positive predictions out of all actual positive instances for a particular category. Recall and precision are calculated using the same equation; however, true positive and false positive represent the total true and false positives in the case of precision, whereas they represent the true and false positives for a particular category in the case of recall, as shown in Equation 4 [40, 28].

$$Recall_{category1} = \frac{(True\,positives)_{category1}}{(True\,positives)_{category1} + (False\,positives)_{category1}} \tag{4}$$

The fscore balances precision and recall, providing a single metric that reflects both the model's accuracy in classifying positive pixels and its ability to capture all instances of a class [40, 28]. fscore is calculated as shown in Equation 5 [40, 28].

$$fscore = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5}$$

The dice loss is metric that assesses the overlap between the ground truth and predicted segmentation mask. It has a value between 0 and 1 and is calculated as shown in Equation 6 [37], where false negatives represents the pixels in the ground truth not predicted by the segmentation model in the predicted segmentation map. Generally, a lower dice loss indicates that the segmentation model can delineate the boundaries of the considered category more accurately [37].

$$Diceloss = 1 - \frac{True\,positives}{True\,positives + False\,positives + False\,negatives} \tag{6}$$

The evaluation of computational burden involves two primary metrics: the algorithm's speed of completion and the amount of computational memory it requires [40, 28]. The execution time is measured as the total processing time, starting from the moment an individual image is introduced to the system or algorithm until the pixel-wise semantic segmentation results are obtained. The performance of this metric is heavily influenced by the hardware being used. Therefore, any execution time measurement for an algorithm should be accompanied by a detailed description of the hardware setup [40, 28]. For algorithms based on deep learning, the offline (training) and online (testing) operations may have significantly different time durations. Hence, the execution time refers specifically to the online operation, which denotes the test duration for a single image [40, 28].

2.7 Effect of Illumination

Illumination can significantly affect the performance of semantic segmentation networks. This is because different lighting conditions can change the appearance of objects, making it difficult for the network to distinguish between them [32]. Illumination can affect performance metrics of semantic segmentation networks because of its effect on color and brightness, shadows and highlights, reflections and refractions, and texture and pattern [32]. Changes in color and brightness can affect the accuracy of the network's color model, which is based on the assumption that objects have a specific color and intensity [32].

For example, if an object is illuminated by a bright light source, it may appear brighter than it actually is, leading to errors in the network's segmentation. Second, shadows and highlights can create ambiguity in the network's segmentation, as they can mask the edges and contours of objects. This can result in false positives or false negatives in the segmentation results [32]. Third, reflections and refractions can also affect the appearance of objects, making it difficult for the network to distinguish between them [32]. For example, if an object is reflective, it may appear to be part of the background, leading to errors in the network's segmentation [32]. Lastly, changes in texture and pattern can affect the network's ability to distinguish between objects. For example, if an object has a complex texture or pattern, it may be difficult for the network to segment it accurately [32]. Training a semantic segmentation model involves teaching a neural network to predict pixel-wise class labels (categories) for each pixel in an input image [9]. Sections 2.8 highlights some of the important considerations for training a semantic segmentation algorithm.

2.8 Considerations for Training a Semantic Segmentation Algorithm

Overfitting is a common challenge when training deep learning models, including semantic segmentation networks. It is a phenomenon that is characterized by a model learning to perform extremely well on the training data but failing to generalize to unseen or new data [24, 35]. In the context of semantic segmentation, overfitting can lead to the model producing highly accurate segmentations on the training set but performing poorly on real-world or previously unseen images [24, 35]. The effects of these factors are subsequently further explained.

If the training dataset is small, the model may memorize the training samples rather than learning generalizable features. Overly complex models with a large number of parameters have a higher risk of overfitting, especially when the dataset is limited. Therefore, proper data augmentation is essential to introduce variability and help the model generalize better [24, 35]. Without sufficient augmentation, the model might overfit to the specific data distribution in the training set. Small objects or rare classes in the training data may not be sufficiently represented, leading the model to struggle with accurate segmentation on these instances [24, 35]. If the dataset has imbalanced class distribution, where some classes have significantly more instances than others, the model might prioritize the dominant classes and perform poorly on minority classes. Selecting an appropriate dataset is a crucial step in training a semantic segmentation model and consists of ensuring that the dataset is sufficiently large and diverse, with sufficient labeled examples of the classes(categories) that the network is required to segment [24, 35]. Preparing the dataset involves properly annotating the images (i.e., labeling each pixel in an image with the corresponding class/category it belongs to) and dividing them into training, validation, and testing sets. Insufficient use of regularization techniques such as dropout, weight decay, or batch normalization can lead to overfitting [24, 35].

Optimization algorithms are used to update the parameters of a neural network during the training process in semantic segmentation tasks. These algorithms seek to minimize the chosen loss function and guide the model toward better performance. RMSprop is an adaptive learning rate optimization algorithm [24, 35]. It maintains a moving average of squared gradients for each parameter and adjusts the learning rate based on the magnitudes of these averages. This helps the algorithm adapt to different scales of gradients and can lead to faster convergence [24, 35]. Momentum is a technique that introduces a moving average of past gradients to the SGD optimization process. This helps in reducing oscillations and leads to faster convergence, especially in regions with high curvature. It allows the optimization process to have inertia and helps the algorithm overcome local minima [24, 35]. Adaptive Moment Estimation (Adam) is an adaptive optimization algorithm that combines ideas from both RMSprop and momentum. It maintains moving averages of both the first-order moments (the mean) and second-order moments (the uncentered variance) of the gradients [24, 35]. These moving averages are used to adjust the learning rate for each parameter individually. Adam often performs well with default settings and is widely used due to its robustness and efficiency.

In semantic segmentation, a loss function plays a crucial role in training a neural network to accurately segment objects within an image. The loss function quantifies the difference between the predicted segmentation and the ground truth segmentation, guiding the optimization process during training [24, 35]. Several loss functions are commonly used in semantic segmentation tasks. However, cross-entropy (Softmax Loss) loss is one of the most widely used loss functions for semantic segmentation. It measures the dissimilarity between the predicted class probabilities and the true class probabilities [24, 35]. In semantic segmentation, the output of the network is usually a per-pixel probability distribution over classes. The cross-entropy loss for a pixel can be calculated as the negative logarithm of the predicted probability of the correct class. This encourages the network to assign high probabilities to the correct class for each pixel [24, 35].

The learning rate is a critical hyperparameter in training neural networks. It determines the step size taken in the direction of the gradient during optimization. Choosing an appropriate learning rate is essential for achieving efficient and effective training [24, 35]. If the learning rate is too high, the optimization process might oscillate around the optimal solution or even diverge. Large updates to the model's parameters can cause the loss function to increase instead of decrease, leading to instability in training. In contrast, a very low learning rate can result in slow convergence, where the model takes tiny steps towards the optimal solution [24, 35]. Furthermore, training might be excessively time-consuming, and the model might get stuck in local minima [24, 35].

In the context of machine learning and deep learning, an ”epoch” refers to a complete pass through the entire training dataset during the training of a model [24, 35]. In the case of semantic segmentation, which is a computer vision task involving the classification of each pixel in an image into different classes, an epoch consists of feeding all the training images through the model, computing the loss (a measure of how well the model's predictions match the actual ground truth), and updating the model's parameters using an optimization algorithm like gradient descent [24, 35]. During an epoch, the model's parameters are updated iteratively based on the gradients of the loss with respect to those parameters [24, 35]. The goal is to minimize the loss, which in turn improves the model's ability to accurately segment objects in images. In practical terms, training a deep learning model, especially for complex tasks like semantic segmentation, typically requires multiple epochs. Training for too few epochs might result in an underfit model that has not learned the underlying patterns in the data, while training for too many epochs might lead to overfitting, where the model becomes too specialized to the training data and does not generalize well to new, unseen data [24, 35].

3 Significance of the study

This sections draws upon the concepts explained in Sections 2.1 to 2.7 to highlight the significance of the study. Section 2.5 highlights encoder-decoder architectures (i.e., ResNet, and DeeplabV3Plus). These architectures have various characteristics that enable them to offer improved. For example, ResNet152 and ResNet101 differ in terms of depth, model size, and capacity [50, 49, 31]. From the process of how the layers of a CNN function, as explained in Section 2.4, differences in the number of layers result in improved segmentation accuracy. In addition, UNet and DeeplabV3Plus have distinct features, which aim to improve performance, including skip connections and a symmetric architecture for UNet and atrous convolutions and atrous spatial pyramid pooling for DeepLabV3Plus. This study quantitatively assesses various architecture (i.e., resnet152DeepLabV3Plus, resnet101DeepLabV3Plus, and resnet101Unet) to provide a link between type of architecture and performance metrics, thus providing a reference for future semantic segmentation applications of encoder-decoder architectures.

Changes in illumination alter texture and color of objects and regions [27]. This is substantiated by the fact that certain colors can appear brighter than they actually are under illumination or darker without proper illumination. Thus, the numbers in the corresponding pixel matrix/matrices that represents the images of the scene will not be an accurate representation of actual colors of the objects in the image, as digital images are essential matrices, where each pixel is represented by a number representing the brightness and intensity, (Section 2.1). Section 2.4 explains how CNNs use convolution , which is the multiplication of the cell values at specific rows and columns within the image matrix with the values of corresponding cells in the filter matrix, to adjust the fighter matrix to recognize various edges and contours for semantic segmentation. Thus, if the image matrix is affected by changes in illumination, so will the performance of the semantic segmentation algorithm. This study assesses the performance of encoder-decoder architectures these varying illumination conditions to quantify the effect of the changes in illumination on the performance metrics. The objectives of the study are explained in Section 3.1.

3.1 Objectives

- To evaluate the performance metrics (i.e., intersection over union, accuracy, recall, precision, and fscore) of three segmentation networks using a dataset captured under different illumination conditions (sunny and cloudy conditions) to qualitatively asses the effect of illumination on the performance metrics.
- To evaluate the performance metrics (i.e., intersection over union, accuracy, recall, precision, and fscore) of three semantic segmentation networks using the same dataset to qualitatively explain the relationship between perception performance and network architecture (i.e., resnet152DeepLabV3Plus, resnet101DeepLabV3Plus, and resnet101Unet).

4 Methods and Apparatus

The dataset consists of two sets of driving scene images from the for Scientific and Industrial Research (CSIR), obtained under the cloudy and sunny conditions, as shown in Figure 17. The sunny and cloudy data consist of 4389 and 4409 images (frames), respectively .

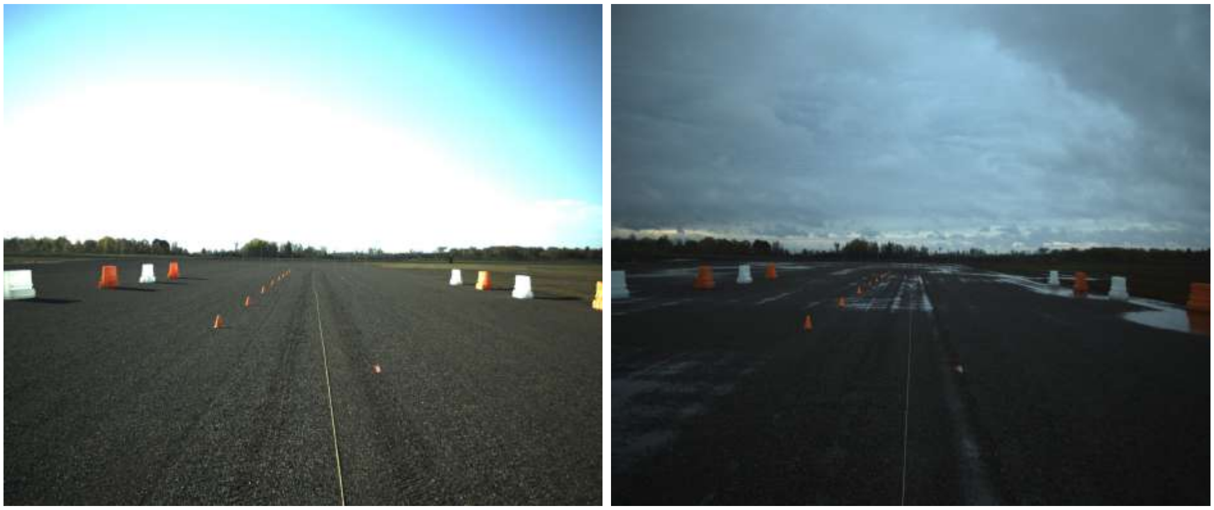
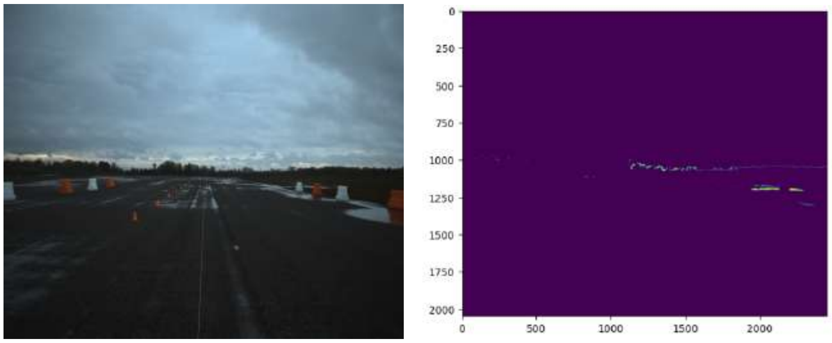
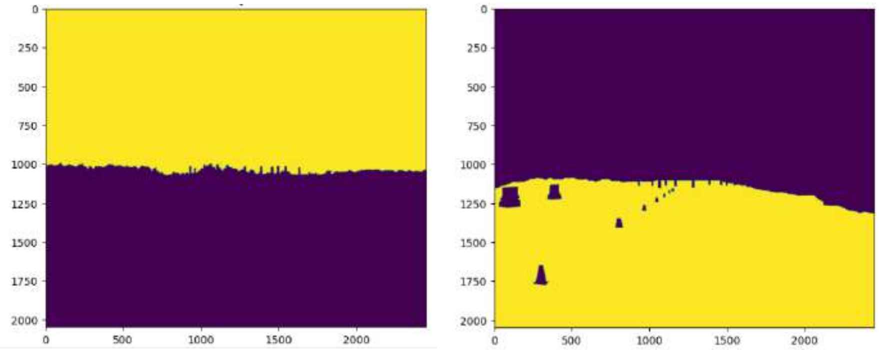

(a) Sunny image (b) Cloudy image

Figure 17: Driving scene images

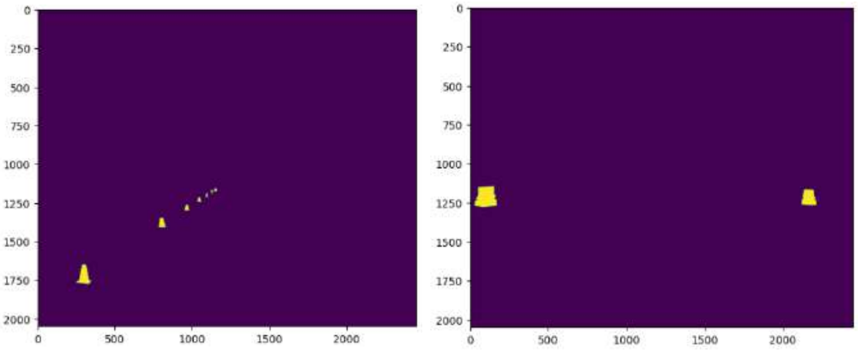
Pixel-level labeling, a computer vision task that involves classifying each pixel in an image into a specific category or class, is a critical aspect of semantic segmentation. Pixel-level labeling in the context of semantic segmentation refers to the process of manually or automatically assigning a class label to each individual pixel in an image based on the category that pixel belongs to. These images are called ground truth maps and are the basis for Calculating the evaluation metrics in Section 2.6. MATLAB Image Labeler was used for pixel level annotations of the images. Seven classes (categories) were considered, namely trees, cones, sky, road, white road barricade, orange road , and poles, as shown in Figure 18. In addition to the aforementioned categories, some of the pixels did not correspond to any category, as shown in Figure 18. This was a result of errors in manually labelling the images (i.e., some of the pixels were excluded by the annotating tools in MATLAB Image Labeler). Google Colab/Colaboratory is a cloud-based platform provided by Google that enables execution of Python code in a Jupyter Notebook environment. The platform is cloud based. Thus, it does not require installation of any software. In addition, it provides free access to graphics processing units and tensor processing units, which can significantly accelerate the execution of machine learning and data analysis tasks. The code for the investigation is executed using the Google Colab interface (Python 3.0, 12.7 GB RAM).



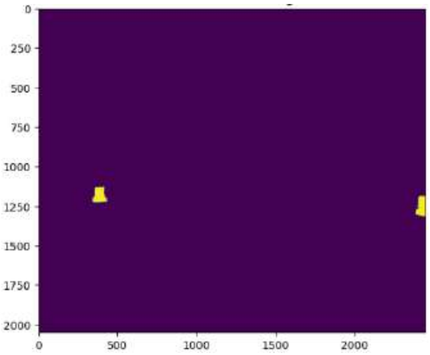
(a) Sunny image (b) Unlabelled pixels



(c) Sky pixels (d) Road pixels



(e) Cone pixels (f) White road barricade pixels



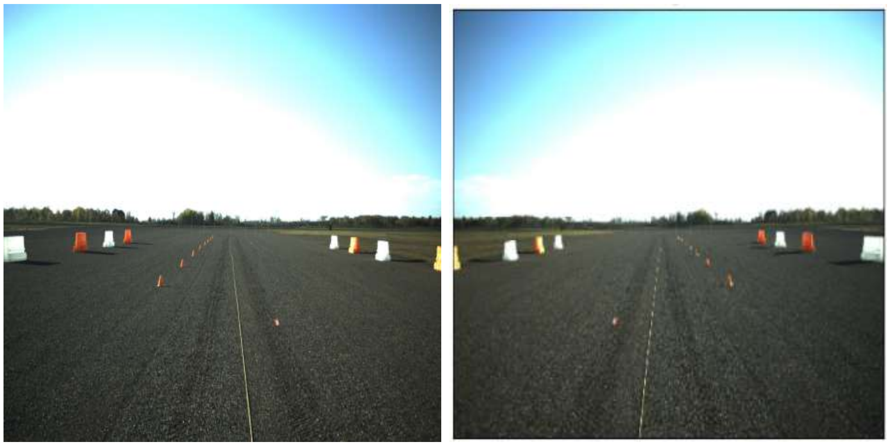
(g) Orange road barricade pixels

Figure 18: (a) Original image and pixels corresponding to (b) unlabelled (c) sky, (d) road, (e) cones, (f) white road barricade, and (g) orange road barricade (f). Labels for each category shaded in yellow

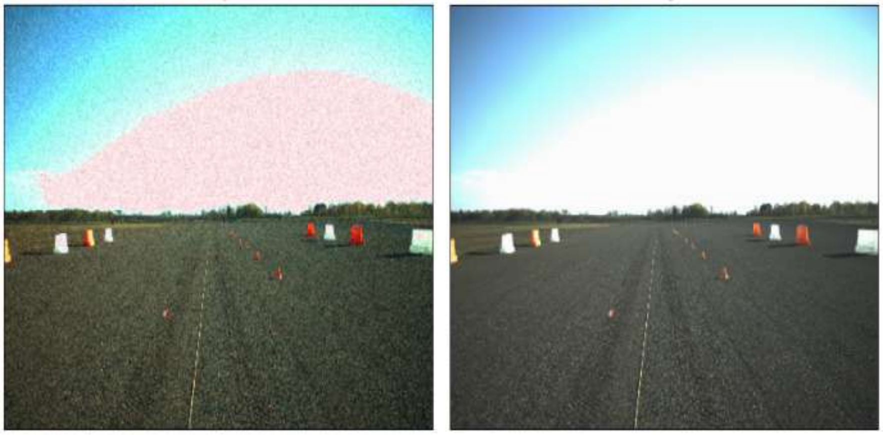
To mitigate overfitting (Section 2.8) in semantic segmentation, thereby mitigating the risk of the semantic segmentation model underperforming on unseen data, the following strategies were implemented: (1) a diverse dataset with enough labeled examples for each class was implemented, (2) data augmentation techniques were applied to increase the variability of the training data, (3) a regularization method was applied to prevent the model from becoming too complex, (4) transfer learning from pre-trained models on larger datasets was used to leverage learned features, (5) the training process was monitored and early stopping was implemented to prevent overfitting, and (6) the model was evaluated on an independent test set to evaluate its generalization performance.

(1) To diversify the dataset and ensure sufficient examples of each class, images corresponding to frames in certain intervals from both the cloudy and sunny datasets were used. The sunny and cloudy dataset was split into three separate datasets, as follows: training dataset (frames 2–16, 800–810, 820–830, 840–850, 1200–1210, 1220–1230, 1240–1250, 1400–1410, 1420–1430, 1440–1450, 1600–1610, 1620–1630, 1640–1650), validation dataset (frames 20, 80, 120, 860, 870, 880, 1260, 1270, 1280, 1460, 1470, and 1480), and test dataset (frames 30, 890, 900, 910, 920, 930, 1290, 1300, 1310, 1320, 1490, 1500, 1510, and 1520). (2) In addition, five image augmentation techniques were applied to supplement the training dataset: horizontal flip, random brightness, random contrast, hue saturation, and gaussian noise, as shown in Figure 19.

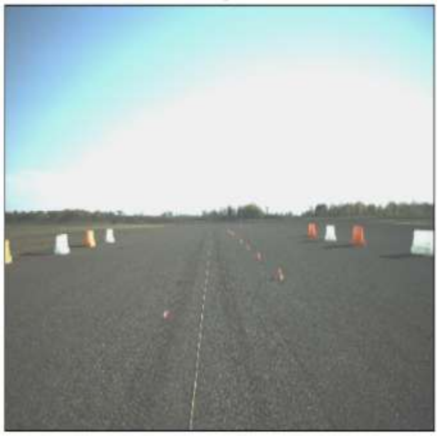
The horizontal flip technique flips the rows and columns associated with the pixel matrix about the y-axis. An example of the resulting image is shown in Figure 19a. Additionally, the random brightness and random contrast techniques randomly changes the brightness and contrast, respectively. The arguments for these functions include the limits for the brightness and contrast, respectively, which can have values between the range 0.1–1. They were both set to 0.4 in this study. Examples of the resulting images are presented in Figures 19b and 19c, respectively. Adjusting the hue of an image involves shifting the hue values of all its pixels along the color wheel. This can result in a change of the perceived color of the image, making it appear more tinted towards a particular color. Image saturation is a measure of the intensity or purity of the colors in an image. It describes how vivid and vibrant the colors are in comparison to a grayscale version of the same image. In more technical terms, saturation refers to the amount of gray in a color. Increasing the saturation factor enhances the vividness of the colors, while decreasing it reduces the saturation, potentially resulting in a black-and-white image if saturation is completely removed. The ”huesaturation” technique implements both of these techniques on the same image and takes an argument in the range 0.1–1. A value of 1 was used as the argument in this study, and an example of the resulting image is shown in Figure 19d.



(a) Horizontal flip (b) Random brightness



(c) Random contrast (d) Hue saturation



(e) Gaussian noise

Figure 19: Augmentation techniques used on the training dataset

(3) Cross-entropy (Softmax Loss) loss is one of the most widely used loss functions for semantic segmentation, as explained in Section 2.8. Thus, the Cross-entropy Loss was implemented in this study. (4) Transfer learning is a technique in machine learning that enables the leveraging of knowledge learned from one task to improve the performance on another related task. For example, if there is an existing model that can segment cars in images, some of features can aid to segment trucks in images as well. This eliminates the need of large training data to train a new model, thereby saving time. Transfer learning is especially useful for deep learning, where models often require a lot of data and computational power to train. A pre-trained model that has already learned some general patterns from a large dataset can be fine-tuned for a specific task with less data and resources. ImageNet is a large visual database designed for use in visual object recognition software research. ImageNet is based on the WordNet hierarchy of nouns and contains more than 14 million images that have been hand-annotated by the project. For this study, different pretrained encoders pre-trained using the ImageNet dataset were used with different decoders. The pre-trained encoders and corresponding decoders are listed in Table 1, where the third column of the table includes the names used to denote the three architectures used in this study.

Table 1: Models implemented in the study [17]

Encoder	**Decoder**	**Model name**
resnet152	DeepLabV3Plus	resnet152DeepLabV3Plus
resnet101	DeepLabV3Plus	resnet101DeepLabV3Plus
resnet101	UNet	resnet101UNet

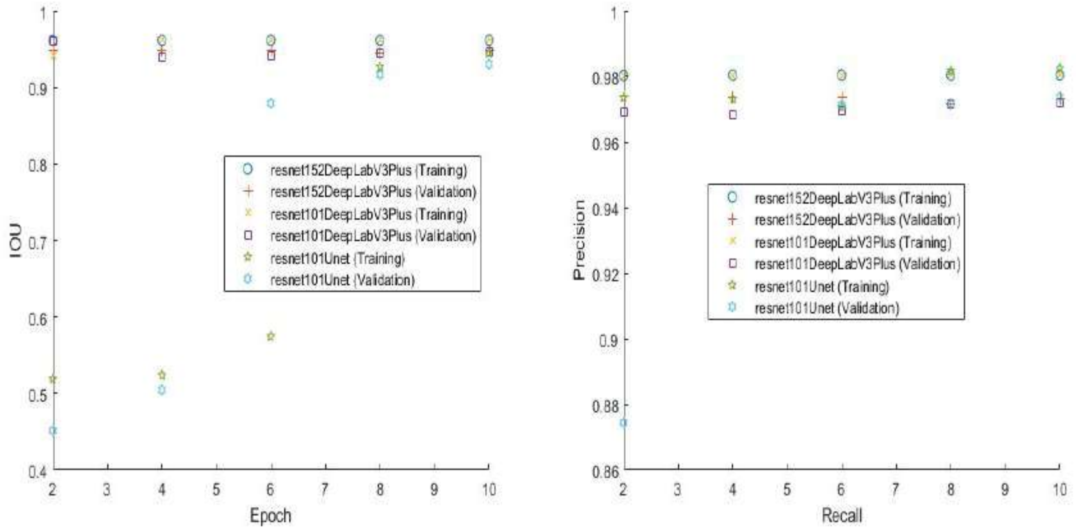
At each training epoch, the evaluation metrics, calculated using Equations 1 to 6, were evaluated for the training dataset (consisting of the cloundy and sunny image frames and augmented images) using a batch size of 6 and a learning rate of 0.001. The metrics were evaluated after every batch. This was done for all three models (i.e., resnet152DeepLabV3Plus, resnet101DeepLabV3Plus, and resnet101UNet). After each epoch, the metrics were evaluated for each model using the validation dataset (consisting of the cloundy and sunny image frames) the to ensure that the metrics obtained at every epoch during training are consistent with those for unseen image (i.e., validation images). In addition, the segmentation models were tested using the test dataset, which was split into sunny and cloudy images, to evaluate the performance metrics of each model on the sunny images and cloudy images of the test dataset. The segmentation maps predicted by each segmentation model were displayed to visualize and verify the metrics obtained during testing at epochs 2, 6, and 10.

5 Results and Discussion

5.1 Training and Validation

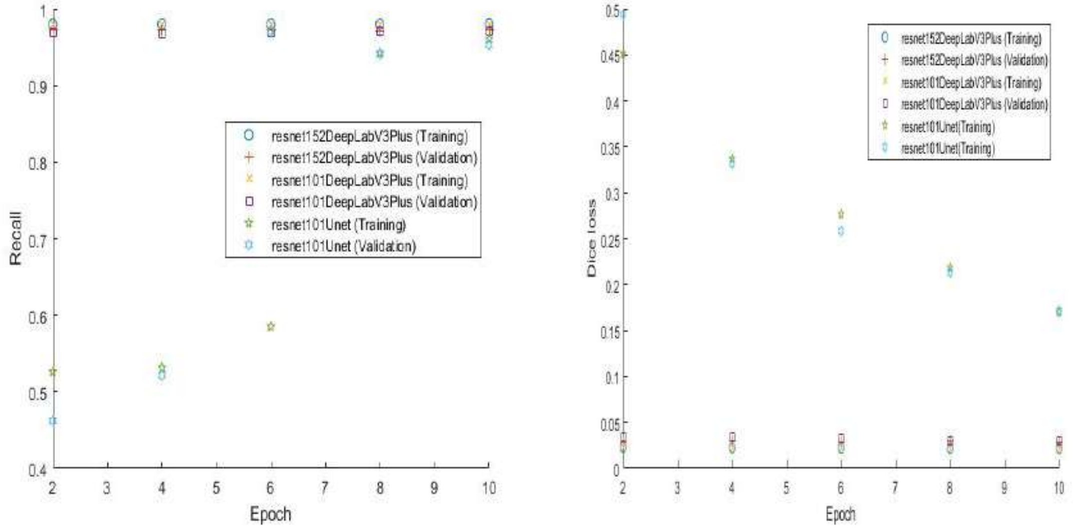
The IOU, precision, recall dice loss, and fscore obtained during the training and validation stages of the study are presented in Figures 20a to 20e, respectively, for the resnet152DeepLabV3Plus, resnet101DeepLabV3Plus, and resnet101Unet segmentation models. Epochs 2, 4, 6, 8, and 10 were considered to observe whether increasing the number of epochs during training resulted in any differences in the performance metrics. Two phenomena can be observed in Figures 20a to 20e. (1) The first phenomenon is that the IOU, precision, recall, dice loss, and fscore remain approximately the same with increasing number of epochs for the resnet152DeepLabV3Plus and resnet101DeepLabV3Plus segmentation models but vary for the resnet101Unet segmentation model (i.e., the IOU, precision, recall, and fscore increase with increasing number of epochs, whereas the dice loss decreases with increasing number of epochs. (2) The second phenomenon is that the difference between the IOU, precision, recall, dice loss, and fscore values obtained during the training and validation stages is different with increasing number of epochs. The implications of these phenomena are subsequently discussed. First, the high values (greater than 0.9 (90%)) of the for the IOU (Figure 20a), precision (Figure 20b), recall (Figure 20c), and fscore (Figure 20e) and the low values of the dice score (lower than 0.05 (5%)) for the resnet152DeepLabV3Plus and resnet101DeepLabV3Plus segmentation models indicate that these models outperformed the resnet101Unet segmentation model. This result is further assessed in Section 5.2 using visualizations of the predicted semantic segmentation maps for all three models at epochs 2, 6, and 10.

However, the approximately constant values of the IOU, precision, recall, dice loss, and fscore for the epochs considered indicate that the training hyperparameters (i.e., learning rate, batch size, optimization algorithm, loss function, data augmentation, regularization, data balancing, and image size and padding) are not effective for training the resnet152DeepLabV3Plus and resnet101DeepLabV3Plus segmentation models. This is substantiated by the fact that Choosing an appropriate learning rate is essential for achieving efficient and effective training. If the learning rate is too high, the optimization process might oscillate around the optimal solution or even diverge. Large updates to the model's parameters can cause the loss function to increase instead of decrease, leading to instability in training, as explained in Section 2.8. This is consistent with the observations observations in the metrics corresponding to the resnet152DeepLabV3Plus and resnet101DeepLabV3Plus segmentation models in Figure 20. Therefore, effective training would require a lower learning rate.



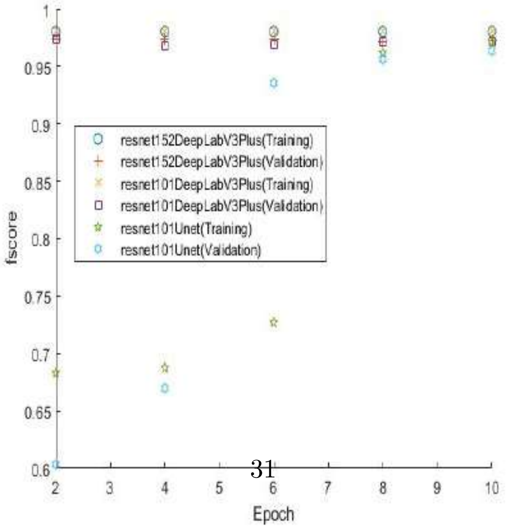
(a) IOU

(b) Precision



(c) Recall

(d) Dice loss



(e) fscore

Figure 20: Metrics monitored during testing and validation

The second implication is with regards to the difference between the in the metrics obtained in the training and validation stages for the resnet101Unet segmentation model. The largest difference in the performance metrics evaluated during training and validation is observed at epochs 2 and 6 for the resnet101Unet segmentation model (Figures 20a to 20e). Because different images were used in the training and validation datasets, differences in the performance metrics indicate that the performance of the resnet101Unet is not independent of the images used (i.e., the model does not perform consistently for on unseen data). The second implication is that this inconsistency is dependent on the number of epochs during training (i.e., epoch 6 presents the largest difference in evaluation metrics, followed by epoch 2) for the resnet101Unet segmentation model. However, the results for the performance metrics in the training are approximately similar to those obtained in the validation stage for epochs 4, 8, and 10 of the resnet101Unet segmentation model and thus deemed acceptable. However, for the resnet152DeepLabV3Plus and resnet101DeepLabV3Plus, the metrics evaluated during the training and validation stages did not exhibit any significant differences. Thus, the results for these segmentation models are deemed to be valid. The results obtained during the testing stage of the study are subsequently discussed, considering the limitations of the semantic segmentation models due to training hyperparameters.

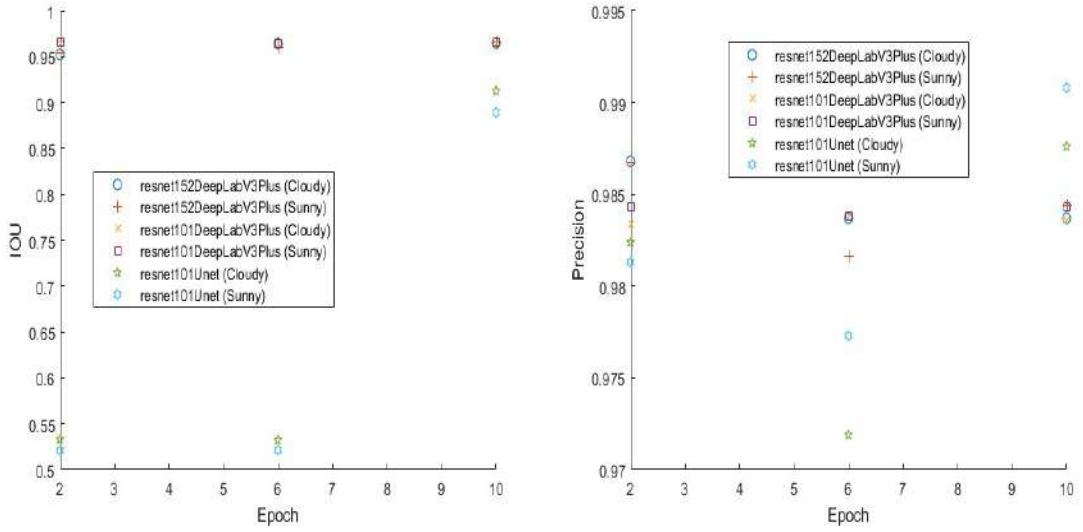
5.2 Testing

The execution time was assessed as the total processing time, starting from the moment an the individual image was introduced to the algorithm until the pixel-wise semantic segmentation results. This was done for all three semantic segmentation algorithms using images from the cloudy and sunny datasets, respectively, as shown in Table 2. The execution times for the resnet101DeepLabV3 and resnet101UNet segmentation models were generally in the range 5–6 s, whereas the execution time for the resnet152DeepLabV3Plus segmentation model was in the range 7–8 s. This implies that changing the encoder had a significant effect on the execution time, whereas changing the decoder had minimal effect on the execution time. ResNet101 consists of 101 layers, including convolutional and fully connected layers, whereas ResNet152 is deeper and consists of 152 layers. It includes more convolutional layers and building blocks. ResNet101 typically has fewer parameters compared to ResNet152 due to its smaller depth. Thus, ResNet152 has more parameters and thus requires more computational resources [50, 49, 31].

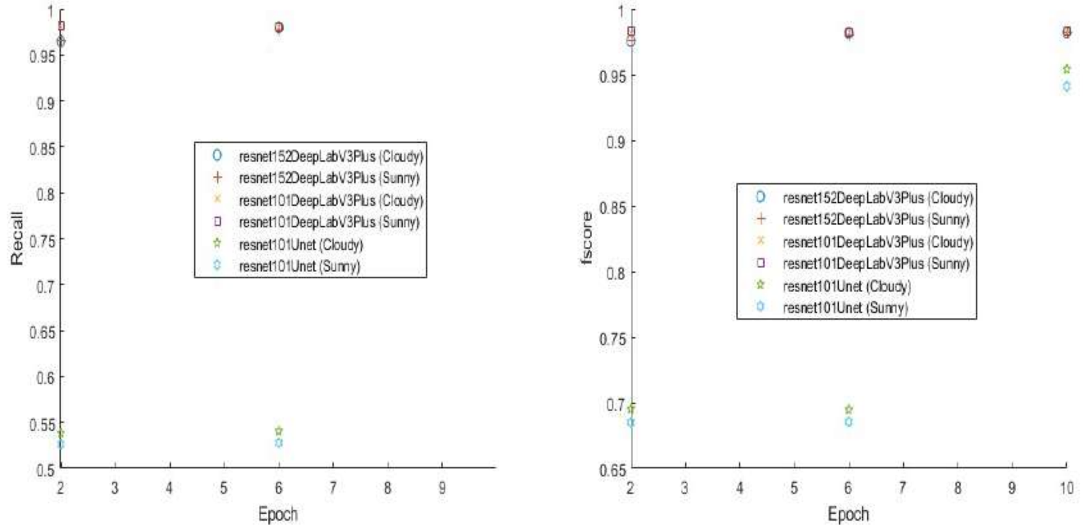
Table 2: Testing times for architectures considered: frame 0850 sunny and 0850 cloudy

Architecture	**Epoch 2 (Cloudy)**	**Epoch 6 (Cloudy)**	**Epoch 10 (Cloudy)**	**Epoch 2 (Sunny)**	**Epoch 6 (Sunny)**	**Epoch 10 (Sunny)**
resnet152DeepLabV3Plus	7 s	7 s	8 s	7 s	7 s	8 s
resnet101DeepLabV3	6 s	6 s	6 s	5 s	6 s	6 s
resnet101UNet	6 s	6 s	6 s	6 s	5 s	6 s

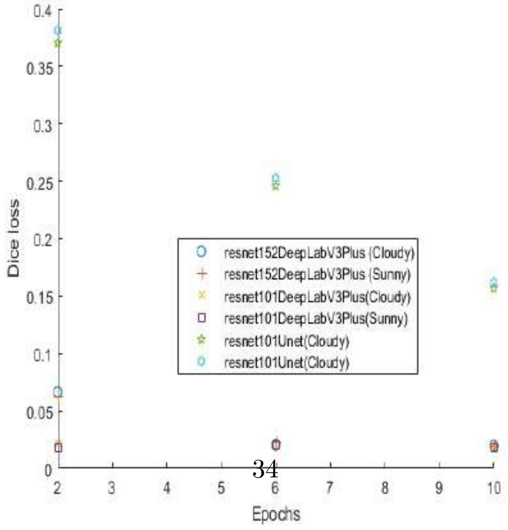
The IOU, precision, recall dice loss, and fscore values obtained during the training and validation stages of the study are presented in Figures 21a to 21e, respectively, for the resnet152DeepLabV3Plus, resnet101DeepLabV3Plus, and resnet101Unet segmentation models to provide a common reference for comparing the differences in the perfomance of the three models. These metrics were evaluated for both the cloudy and sunny test datasets to observe the effect of illumination on the performance of the models. This discussion considers that the training hyperparameters (learning rate) used in this study were not effective for the resnet152DeepLabV3Plus and resnet101DeepLabV3Plus architectures, as explained in Section 5.1. Therefore, the comparison in this section is only qualitative. A quantitative comparison would require tuning the training hyperparameters for the resnet152DeepLabV3Plus and resnet101DeepLabV3Plus architectures. Only the resnet101UNet models trained for 4, 8, and 10 epochs were deemed to be generalizable and valid in Section 5.1. Therefore, only these epochs are considered in this discussion. The findings are as follows. (1) The first metric considered is the dice loss, which is the metric that assesses the overlap between the ground truth and predicted segmentation mask. Generally, a lower dice loss indicates that the segmentation model can delineate the boundaries of the considered category more accurately [37], as explained in Section 2.6. (1) From Figure 21e, it can be observed that the dice loss values of the resnet152DeepLabV3Plus and resnet101DeepLabV3Plus are significantly lower than those for the resnet101Unet model at all epochs for both the sunny and cloudy datsets. (2) In addition, the IOU, precision, recall, and fscores values of the resnet152DeepLabV3Plus and resnet101DeepLabV3Plus architectures are generally significantly higher than those of the resnet101Unet architecture, as shown in Figures 21a to 21d. (3) Despite using different encoders, the performance metrics values of the resnet152DeepLabV3Plus and resnet101DeepLabV3Plus are approximately similar. Lastly, the performance metrics values of the resnet101Unet generally lower for the sunny dataset compared to those for the cloudy dataset, whereas the the performance metrics values of the resnet152DeepLabV3Plus and resnet101DeepLabV3Plus are approximately for the sunny dataset are approximately the same as those for the cloudy datasets. These findings are subsequently linked to the characteristics of the architectures and digital images.



(a) IOU (b) Precision



(c) Recall (d) fscore



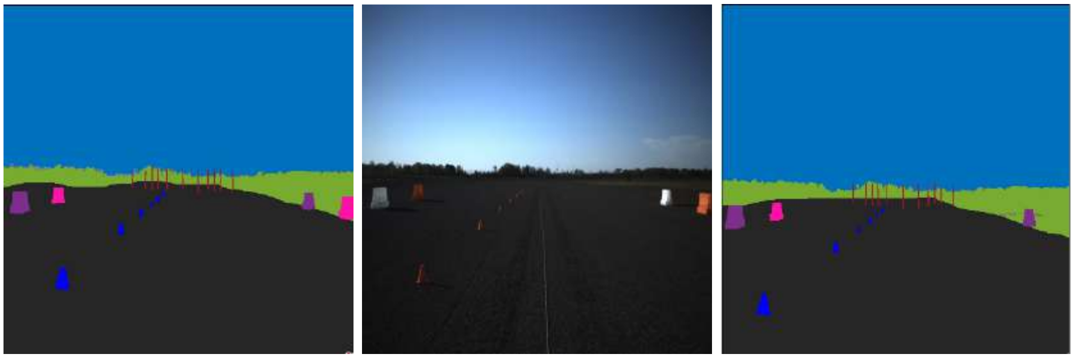
(e) Dice loss

Figure 21: Metrics monitored during the testing stage of the study

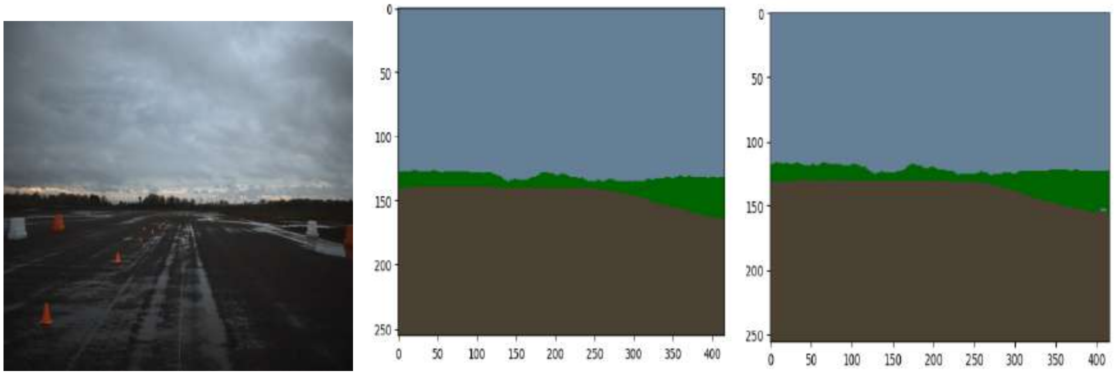
From the basic convolution process outlined in Section 2.4, it can be deduced that deep neural networks are heavily dependent on the image information (pixel matrix values) to perform the semantic segmentation tasks. For most semantic segmentation architectures, the feature representation used by the model does not capture sufficient boundary information or context information [26]. For example, some models, including the architectures considered in this study, use low-resolution feature maps or pooling operations that reduce the spatial resolution and lose fine-grained details [26]. This can make it difficult for the model to distinguish between similar or adjacent classes, or to handle complex scenes with multiple objects and occlusions. Thus, boundary artifacts are produced, which deteriorate the performance of the model [26]. In addition, Illumination can significantly affect the performance of semantic segmentation networks. This is because different lighting conditions can change the appearance of objects, as highlighted in Section 2.7, which magnifies the boundary artifact phenomenon. However, the difference in the performance metrics of the resnet152DeepLabV3Plus and resnet101DeepLabV3Plus architectures compared to that of the resnet101UNet architecture can be attributed to the different decoder architectures, that is DeepLabV3Plus and UNet. Both architectures have skip connections that connect corresponding layers in the encoder and decoder [23]. These connections allow the decoder to access high-resolution features from the encoder, facilitating precise localization of objects in the segmentation map. However, DeepLabv3Plus uses atrous spatial pyramid pooling (ASPP) to capture multi-scale information and better handle objects at different scales. The ASPP module serves the purpose of gathering context information at multiple scales [48]. The final predictions are generated through an up-sampling process. Within the ASPP network, in addition to the feature map derived from the backbone, four separate atrous convolutions are employed in parallel, each with distinct atrous rates, to effectively segment objects at various scales. To incorporate global context information, image-level features are integrated by applying global average pooling to the last feature map of the backbone [48]. ASPP helps to increase the field of view for extracting multi-scale features. It is used to consider objects at different scales and segment with much improved accuracy [48]. These differences in decoder architectures are attributed to the general superior performance of the resnet152DeepLabV3Plus and resnet101DeepLabV3Plus architectures compared that of the resnet101UNet architecture, even under increased illumination (sunny conditions).

The segmentation maps predicted by each segmentation model were displayed to visualize and verify the metrics obtained during testing at epochs 2, 6, and 10, as shown in Figures 22 to 24. As highlighted in Section 5.1, the learning rate used during training in this study was too high for the resnet152DeepLabV3Plus and resnet101DeepLabV3Plus architectures. This is evidenced by the absence in improvement of the of the predicted segmentation with increasing number of epochs for the resnet152DeepLabV3Plus and resnet101DeepLabV3Plus architectures, as shown in Figures 22 to 24 (images (e) to (h)). The resnet101UNet architecture exhibits improved every with in creasing number of epochs. However, not all the categories in the image were successfully predicted. From Figure 20, the performance metrics exhibit converging behavior during training for the number of epochs considered. Therefore, increasing the number of epochs is not expected to improve performance substantially. Alternatively, the number of images using during training can be increasing and image quality can be improved. Considerations for improving the image quality are subsequently discussed.

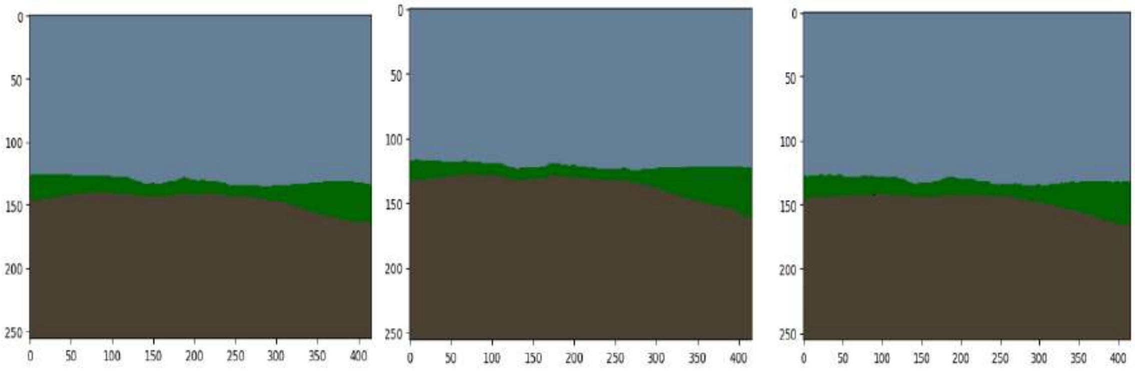
The resolution of an image refers to the number of pixels in the image, usually expressed as width x height (e.g., 1920 x 1080 pixels). Higher resolution images contain more detail and provide finer granularity for 8 processing algorithms. However, higher resolution images may require more computational resources and processing time [44]. Image resolution also affects image labelling, which was done manually in this study. Higher resolutions allow for more accurate and consistent ground truth labels, which improves the generalizability of the semantic segmentation model [44]. Additionally, Color depth, also known as bit depth, represents the number of bits used to represent each pixel's color information. It determines the number of distinct colors that can be displayed in an image. Common color depths include 8-bit (256 colors), 24-bit (16.7 million colors), and 48-bit (281 trillion colors). Higher color depth allows for more accurate color representation and smoother gradients [38]. The file format in which the digital image is stored can affect image processing. Different formats have various compression techniques and support different features. Lossy formats like JPEG can introduce compression artifacts, impacting the quality of subsequent processing. However, lossless formats like PNG retain more image detail but may have larger file sizes [4]. Lastly, image noise refers to random variations in pixel values that occur due to factors such as sensor limitations, low-light conditions, or transmission errors. Noise can affect the accuracy of image processing algorithms by obscuring details or introducing false information. Techniques like denoising filters or noise reduction algorithms are often employed to mitigate noise effects [4].



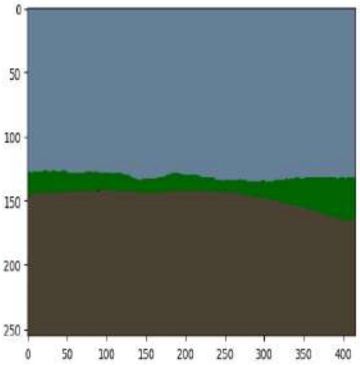
(a) Ground truth - sunny (b) frame0850 - sunny (c) Ground truth - cloudy



(d) frame0850 - cloudy (e) resnet101DeepLabV3Plus (Prediction) - cloudy (f) resnet101DeepLabV3Plus (Prediction) - sunny

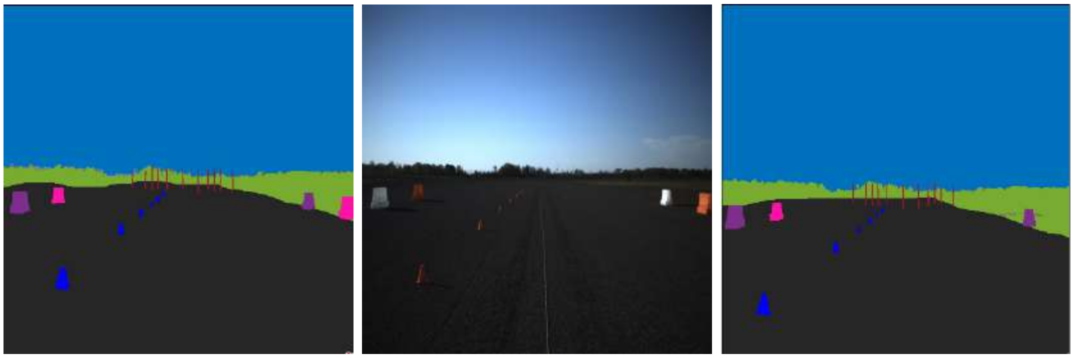


(g) resnet152DeepLabV3Plus (Prediction) - cloudy (h) resnet152DeepLabV3Plus (Prediction) - sunny (i) resnet101Unet (Prediction) - cloudy

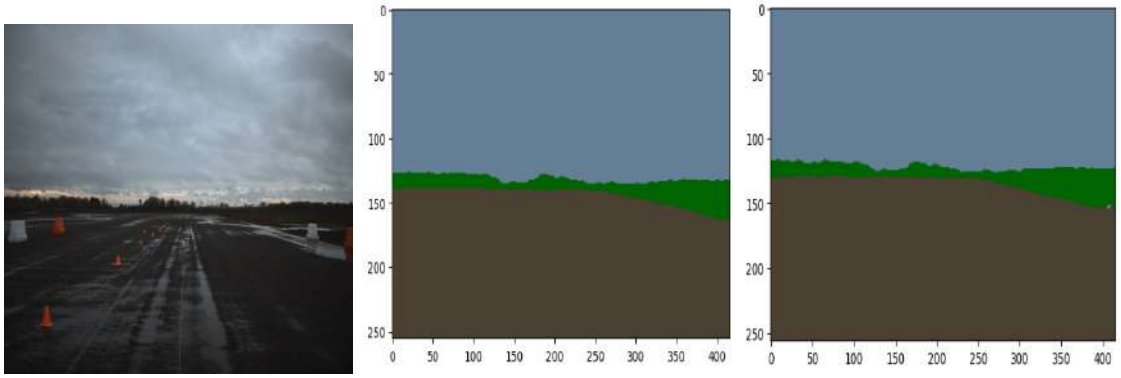


(j) resnet101Unet (Prediction) - sunny

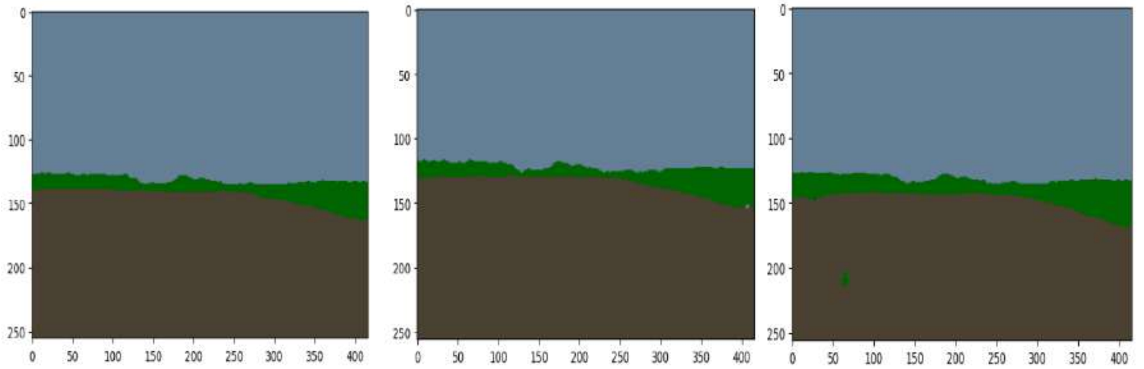
Figure 22: Predicted segmentation map after 2 epochs



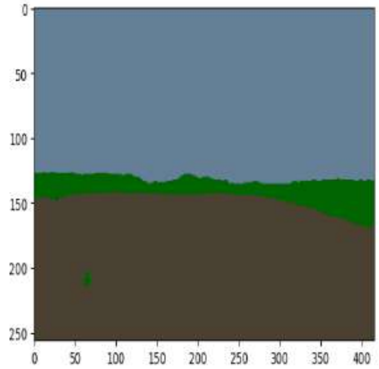
(a) Ground truth - sunny (b) frame0850 - sunny (c) Ground truth - cloudy



(d) frame0850 - cloudy (e) resnet101DeepLabV3Plus (Prediction) - cloudy (f) resnet101DeepLabV3Plus (Prediction) - sunny

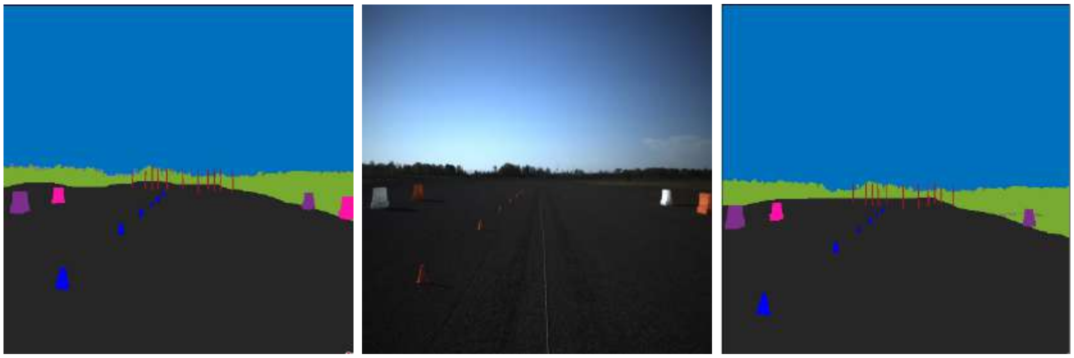


(g) resnet152DeepLabV3Plus (Prediction) - cloudy (h) resnet152DeepLabV3Plus (Prediction) - sunny (i) resnet101Unet (Prediction) - cloudy

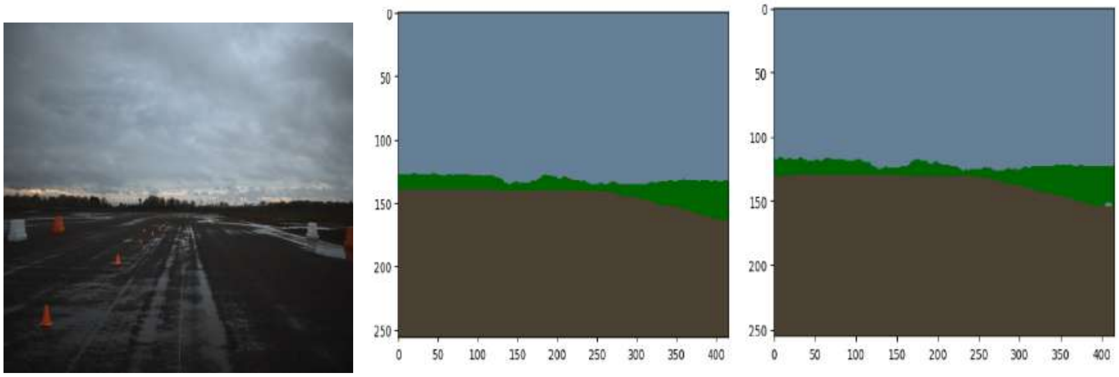


(j) resnet101Unet (Prediction) - sunny

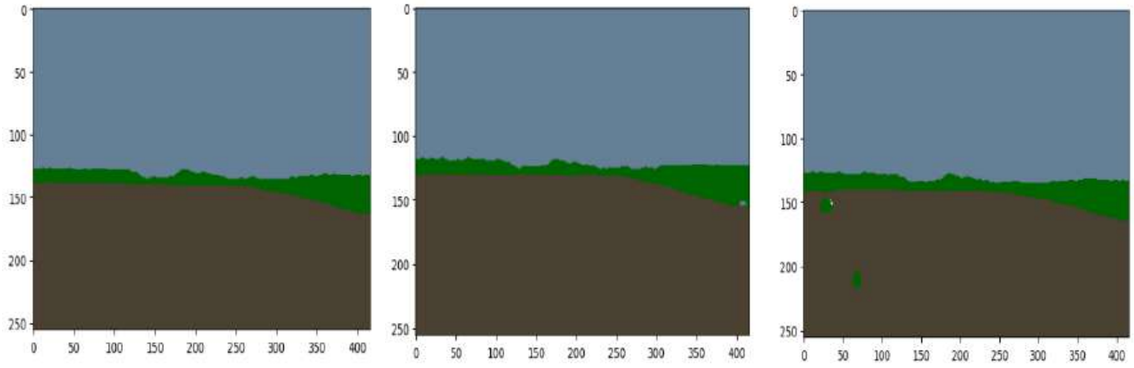
Figure 23: Predicted segmentation map after 6 epochs



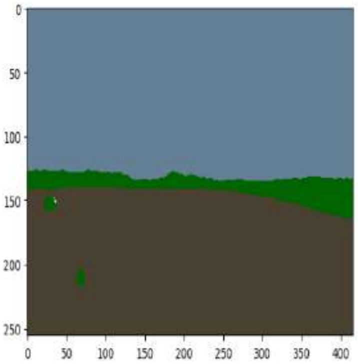
(a) Ground truth - sunny (b) frame0850 - sunny (c) Ground truth - cloudy



(d) frame0850 - cloudy (e) resnet101DeepLabV3Plus (Prediction) - cloudy (f) resnet101DeepLabV3Plus (Prediction) - sunny



(g) resnet152DeepLabV3Plus (Prediction) - cloudy (h) resnet152DeepLabV3Plus (Prediction) - sunny (i) resnet101Unet (Prediction) - cloudy



(j) resnet101Unet (Prediction) - sunny

Figure 24: Predicted segmentation map after 10 epochs

6 Conclusions and Recommendations

In this study, the performance metrics (i.e., intersection over union, accuracy, recall, precision, and fscore) of three segmentation networks were evaluated using a dataset captured under different illumination conditions (sunny and cloudy conditions) to qualitatively asses the effect of illumination on the performance metrics. In addition, the performance metrics (i.e., intersection over union, accuracy, recall, precision, and fscore) of three semantic segmentation networks were evaluated using the same dataset to qualitatively explain the relationship between perception performance and network architecture for the three semantic segmentation models (i.e., resnet152DeepLabV3Plus, resnet101DeepLabV3Plus, and resnet101Unet). The conclusions are as follows:

(1) The training hyperparameters, particularly the learning rate, was only effective for the resnet101Unet segmentation model but too high for the resnet152DeepLabV3Plus and resnet101DeepLabV3Plus semantic segmentation models. (2) The performance of the resnet101UNet segmentation model was not generalizable for all the training epochs considered (i.e., only the results for epochs 4, 8, and 10 were deemed acceptable), whereas the perfomance of the resnet152DeepLabV3Plus and resnet101DeepLabV3Plus models was generalizable for all epochs considered. (3) The execution time was longest for the resnet152DeepLabV3Plus semantic segmentation architecture (7–8 s) and lowest for the resnet101DeepLabV3Plus and resnet101UNet architectures (5–6 s). That is, only changing the encoder of the architecture affected the execution time. This is attributed to the greater number layers (152 layers) in ResnNet152 compared to those in ResNet101 (101 layers). The increased parameters in the former encoder results in an increased execution time. (4) resnet101UNet architecture exhibits decreased performance under sunny conditions compared to cloudy conditions, whereas resnet152DeepLabV3Plus and resnet101DeepLabV3Plus exhibit consistent performance. (5) resnet152DeepLabV3Plus and resnet101DeepLabV3Plus architectures generally exhibit superior (i.e. IOU, precision, recall, fscore, and dice loss) and consistent performance compared to the resnet101UNet architecture under sunny and cloudy conditions. This difference in performance is attributed to the superior DeepLabV3Plus architecture (i.e., atrous spatial pyramid pooling and atrous convolutions) in comparison to the UNet architecture.

To improve the predicted semantic segmentation maps and performace metrics obtained in this study, it is recommended to use a lower learning rate for the resnet152DeepLabV3Plus and resnet101DeepLabV3Plus, increase the number of images used during training, use higher resolution images, use images with higher color depths, use images with lossless image formats, and apply donoising techniques to the images.

References

[1] Towards AI. *Semantic segmentation: A complete guide.* 2023. URL: https://heartbeat.comet.ml/the-5-computer-vision-techniques-that-will-change-how-you-see-the-world-1ee19334354b.

[2] D. Mehta et al. "Simple and Efficient Architectures for Semantic Segmentation". In: *IEEE Xplore* (2022).

[3] S. Albawi, T. A. Mohammed, and S Al-Zawi. "Understanding of a convolutional neural network". In: *2017 international conference on engineering and technology (ICET).* Ieee. 2017, pp. 1–6.

[4] C Anghel. "Effect of image quality in computer vision for semantic segmentation of road images". B.S. thesis. University of Twente, 2022.

[5] V. Badrinarayanan, A. Kendall, and R Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation". In: *IEEE transactions on pattern analysis and machine intelligence* 39.12 (2017), pp. 2481–2495.

[6] N Buh. *Activation Functions in Neural Networks: With 15 examples.* 2023. URL: https://encord.com/blog/activation-functions-neural-networks/.

[7] H.S Chatterjee. "A Basic Introduction to Convolutional Neural Network". In: *Medium. com* (2019).

[8] L. Chen et al. "Semantic image segmentation with deep convolutional nets and fully connected crfs. arXiv 2014". In: *arXiv preprint arXiv:1412.7062* (2014).

[9] Saturn Cloud. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way.* 2023. URL: https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way//.

[10] Datagen. *Image Segmentation: The Basics and 5 Key Techniques.* 2023. URL: https://datagen.tech/guides/image-annotation/image-segmentation/.

[11] A. I. Georgevici and M Terblanche. "Neural networks and deep learning: a brief introduction". In: *Intensive Care Medicine* 45.5 (2019), pp. 712–714.

[12] D Gottlieb. *What is a sigmoid function in neural networks?* 2020. URL: https://www.quora.com/What-is-a-sigmoid-function-in-neural-networks.

[13] N Hakira. *Artificial Neural Networks and its Applications.* 2023. URL: https://www.geeksforgeeks.org/artificial-neural-networks-and-its-applications/.

[14] X. hang, N. Jia, and I Ivrissimtzis. "A study of the effect of the illumination model on the generation of synthetic training datasets". In: *Cornell University, New York, United States of America* (2006).

[15] S. Himanshi. *How do computers store images.* 2021. URL: https://www.analyticsvidhya.com/blog/2021/03/grayscale-and-rgb-format-for-storing-images/.

[16] Computer Hope. *Digital Camera.* 2021. URL: https://www.computerhope.com/jargon/d/digicame.htm.

[17] P Iakubovskii. *Segmentation Models: Available Encoders.* 2023. URL: https://smp.readthedocs.io/en/latest/encoders.html.

[18] Javapoint. *Computer vision techniques.* 2023. URL: https://www.javatpoint.com/computer-vision-techniques.

[19] J. Ji et al. "Encoder-decoder with cascaded CRFs for semantic segmentation". In: *IEEE Transactions on Circuits and Systems for Video Technology* 31.5 (2020), pp. 1926–1938.

[20] J. Jordan. *Evaluating image segmentation models.* 2018. URL: https://www.jeremyjordan.me/evaluating-image-segmentation-models/.

[21] R. Kundu. *Image Processing: Techniques, Types, Applications.* 2023. URL: https://www.v7labs.com/blog/image-processing-guide.

[22] J Le. *The 5 computer vision techniques that will change how you see the world.* 2023. URL: https://heartbeat.comet.ml/the-5-computer-vision-techniques-that-will-change-how-you-see-the-world-1ee19334354b.

[23] K.S. Lee et al. "U-Net skip-connection architectures for the automated counting of microplastics". In: *Neural Computing and Applications* 34.9 (2022), pp. 7283–7297.

[24] Z. Li, K. Kamnitsas, and B Glocker. "Analyzing overfitting under class imbalance in neural networks for image segmentation". In: *IEEE transactions on medical imaging* 40.3 (2020), pp. 1065–1077.

[25] C. Liu et al. "Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 2019, pp. 82–92.

[26] H. Ma, H. Yang, and D Huang. "Boundary guided context aggregation for semantic segmentation". In: *arXiv preprint arXiv:2110.14587* (2021).

[27] MathWorks. *Image processing and computer vision.* 2023. URL: https://www.mathworks.com/solutions/image-video-processing/semantic-segmentation.html#:~:text=Semantic%20segmentation%20is%20a%20deep,pavement%2C%20and%20other%20road%20features/.

[28] D. Müller, Iñ. Soto-Rey, and F Kramer. "Towards a guideline for evaluation metrics in medical image segmentation". In: *BMC Research Notes* 15.1 (2022), pp. 1–8.

[29] P Narasimman. *9 Types of Neural Networks: Applications, Pros, and Cons.* 2023. URL: https://www.knowledgehut.com/blog/data-science/types-of-neural-networks.

[30] S Pattanayak. *Pro Deep Learning with TensorFlow.* Apress, 2017.

[31] A.K Prabhakaran, J.J. Nair, and S Sarath. "Thermal facial expression recognition using modified resnet152". In: *Advances in Computing and Network Communications: Proceedings of CoCoNet 2020, Volume 2.* Springer, 2021, pp. 389–396.

[32] D. Pratiwi and I.H Kartowisastro. "Object segmentation under varying illumination effects". In: *New Trends in Intelligent Information and Database Systems.* Springer. 2015, pp. 13–21.

[33] C. R. Rafael and E. W. Woods. *Image Processing: An Introduction.* Prentice Hall, 2022.

[34] O. Ronneberger, P. Fischer, and T. Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18.* Springer. 2015, pp. 234–241.

[35] U. Sehar and M.L. Naseem. “How deep learning is empowering semantic segmentation: Traditional and deep learning techniques for semantic segmentation: A comparison”. In: *Multimedia Tools and Applications* 81.21 (2022), pp. 30519–30544.

[36] C. Solomon and T Breckon. *Fundamentals of Digital Image Processing: A practical approach with examples in Matlab.* John Wiley & Sons, 2011.

[37] C.H Sudre et al. “Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations”. In: *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: Third International Workshop, DLMIA 2017, and 7th International Workshop, ML-CDS 2017, Held in Conjunction with MICCAI 2017, Québec City, QC, Canada, September 14, Proceedings 3.* Springer. 2017, pp. 240–248.

[38] B Sushma and P Aparna. “Effect of Different Color Spaces on Deep Image Segmentation”. In: *2021 IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE).* IEEE. 2021, pp. 1–4.

[39] Z. Tian et al. “Decoders matter for semantic segmentation: Data-dependent decoding enables flexible feature aggregation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2019, pp. 3126–3135.

[40] E. Tiu. *Metrics to Evaluate your Semantic Segmentation Model.* 2019. URL: https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2.

[41] S.H Tsang. *Review: DilatedNet — Dilated Convolution (Semantic Segmentation).* 2018. URL: https://towardsdatascience.com/review-dilated-convolution-semantic-segmentation-9d5a5bd768f5.

[42] I. Ulku and E Akagündüz. “A survey on deep learning-based architectures for semantic segmentation on 2d images”. In: *Applied Artificial Intelligence* 36.1 (2022), p. 2032924.

[43] R. Unnikrishnan, C. Pantofaru, and M Hebert. “Toward objective evaluation of image segmentation algorithms”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* ”29”.”3” (2007), ”929–944”.

[44] J. Wang, B. Liu, and K Xu. “Semantic segmentation of high-resolution images”. In: *Science China Information Sciences* 60 (2017), pp. 1–6.

[45] Z. Wang et al. “Image Quality Assessment: From Error Visibility to Structural Similarity”. In: *IEEE Transactions on Image Processing* 12.1 (2004), pp. 5–8.

[46] D. R. Wilson and Tony R. M. “The Inefficiency of Batch Training for Large Training Sets”. In: *In Proceedings of the International Joint Conference on Neural Networks (IJCNN2000)* 2.1 (2000), pp. 113–117.

[47] Y. Xing, L. Zhong, and X Zhong. “An encoder-decoder network based FCN architecture for semantic segmentation”. In: *Wireless Communications and Mobile Computing* 2020 (2020).

[48] H. Zeng, S. Peng, and D Li. “Deeplabv3+ semantic segmentation model based on feature cross attention mechanism”. In: *Journal of Physics: Conference Series.* Vol. 1678. 1. IOP Publishing. 2020, p. 012106.

[49] L. Zhang et al. “An infrared and visible image fusion algorithm based on ResNet-152”. In: *Multimedia Tools and Applications* 81.7 (2022), pp. 9277–9287.

[50] Q Zhang. “A novel ResNet101 model based on dense dilated convolution for image classification”. In: *SN Applied Sciences* 4 (2022), pp. 1–13.

[51] Q. Zhou et al. “Banet: Boundary-assistant encoder-decoder network for semantic segmentation”. In: *IEEE Transactions on Intelligent Transportation Systems* 23.12 (2022), pp. 25259–25270.