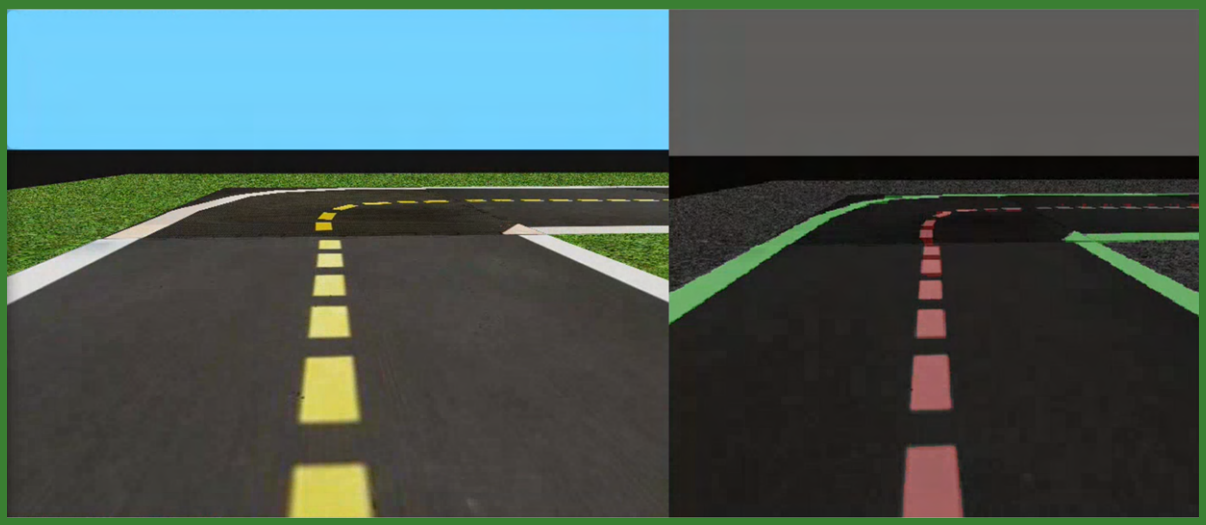
Monocular Navigation for a Duckiebot Using a High-Resolution Encoder-Decoder Architecture



Angelo R. Broere

Layout: typeset by the author using LaTeX.
Cover illustration: Duckietown simulator with image segmentation results, created by the author

Monocular Navigation for a Duckiebot Using a High-Resolution Encoder-Decoder Architecture

Angelo R. Broere
13168215

Bachelor Thesis
Credits: 18 EC

Bachelor *Kunstmatige Intelligentie*

University of Amsterdam
Faculty of Science
Science Park 900
1098 XH Amsterdam

Supervisor
Dr. A. Visser

Informatics Institute
Faculty of Science
University of Amsterdam
Science Park 900
1098 XH Amsterdam

June 30, 2023

Abstract

This thesis compares the performance of the high-resolution, light-weight LEDNet encoder-decoder architecture with a low-resolution vision transformer-based approach in the context of monocular robot navigation. The evaluation focuses on lane-following accuracy and obstacle avoidance. The LEDNet encoder-decoder architecture demonstrates superior performance in terms of navigation accuracy and stability. Its high frame rate and resolution allow for timely recognition of turns, enabling the robot to navigate corners and obstacles at higher speeds compared to the vision transformer. The LEDNet model also outperforms the vision transformer-based models in obstacle avoidance, accurately detecting and classifying obstacles and successfully navigating around them. Additionally, the LEDNet encoder-decoder architecture shows better visibility and avoidance of small obstacles compared to the vision transformer-based models. Although the evaluation is based on a simulated environment, the relative performance differences observed provide valuable insights for further research and potential real-world applications. The results of this study support the use of the LEDNet encoder-decoder architecture for enhancing monocular robot navigation, offering advantages in lane-following accuracy and obstacle avoidance.

Contents

Chapter 1

Introduction

Today, self-driving vehicles are equipped with a multitude of sensors and cameras for comprehensive perception of their surroundings. The seamless integration and functioning of components, such as sensors and cameras, are essential for autonomous vehicles to operate safely, and so the absence or malfunction of these aforementioned components can not only disrupt the vehicle's perception of the environment, but also undermine its ability to process data in real-time and make the right decisions, such as to stay in the lane.

Consequently, given the increasing popularity of self-driving cars and other autonomous vehicles, the need for reliable lane-following and obstacle-avoidance algorithms is paramount. Having a simple and reliable system as a fallback option would provide a secure alternative in cases where the main advanced system fails to function. Such system would be able to safely stay in a lane and avoid obstacles using only one camera in the process. This would be a reliable backup system for autonomous vehicles, like cars and robot container movers.

An example of a simple research robot to test out algorithms for autonomous driving are Duckiebots[1]. Duckiebots are small robots with a front-facing camera (see Fig. 1.1). Duckiebots can drive around in an environment called Duckietown. This environment is made out of tiles that show a part of a road or junction which can be laid out in any order, to create the desired research environment. Duckietown is also available in the form of a simulator.

[1] https://www.duckietown.org/

Figure 1.1: Duckiebot in Duckietown[1]

Saavedra-Ruiz, Morin, and Paull (2022) demonstrated that a Duckiebot is able to navigate through Duckietown using monocular vision and low-resolution image segmentation. The images were processed in real-time by a vision transformer (ViT). This ViT created image segmentations by labeling 8×8 pixel patches with the relevant class and with this segmentation mask, the algorithm calculated the correct steering direction to either stay in lane or avoid an obstacle. The authors suggest that in order to perform monocular robot navigation using high-resolution image segmentations, an encoder-decoder architectures may be applied. High-resolution segmentation models are better than low-resolution segmentation models at detecting obstacles and lane following (Saavedra-Ruiz et al., 2022).

There are many different encoder-decoder architectures available. Generally, encoder-decoders are computationally demanding, so there is need for a lightweight encoder-decoder that can create image segmentations in real-time (Holder & Shafique, 2022). There are real-time performing models available like LaneNet, but this model is only made to recognize road lines (Z. Wang, Ren, & Qiu, 2018). However, it is necessary for this lane-following and object-avoidance algorithm to employ a versatile model that can be trained to effectively recognize obstacles as well.

Consequently, this research uses a flexible and lightweight version of an encoder-decoder, namely LEDNet. LEDNet is able to process images in real-time with 71 frames per second (fps) using an NVIDIA GTX 1080Ti, so it is suitable for robot navigation (Y. Wang et al., 2019). LEDNet creates high-resolution image segmentations, which is needed to create a reliable, high-resolution, monocular robot navigation system. To explore the disparity in performance between ViT and LEDNet in monocular robot navigation, the following research question arises: *How does the performance of the LEDNet encoder-decoder architecture compare to the vision transformer-based approach in terms of monocular robot navigation, lane-following accuracy and obstacle avoidance?*

The anticipated outcome of this study is that the LEDNet-driven Duckiebot is able to navigate much more accurately and stably, for various reasons. First, LEDNet is able to process images with a much higher frame rate compared to the ViT. This enables the LEDNet-driven Duckiebot to turn corners faster than the ViT-driven one, because of the higher frame rate the LEDNet Duckiebot will notice on time when it has made the turn and when to straighten its wheels again. Second, LEDNet is able to create pixelwise image segmentations, whereas ViT only labels 8×8 patches of pixels. Therefore, LEDNet is not only able to notice details, but smaller obstacles will also be visible in the segmentations, while, on the other hand, the ViT needs the majority of the 8×8 image patches to be a certain class for the whole patch to obtain that class as labels.

The structure of this thesis is as follows. To begin, the necessary theoretical background will be discussed. This theoretical background consists of ViT and LEDNet. Furthermore, an explanation of the simulator environment along with the lane-following and obstacle-avoidance algorithm will be provided. Moreover, the research setup will be explained. Finally, the results will be presented and discussed, followed by a consideration of future research directions.

Chapter 2

Theoretical Background

This chapter will discuss necessary theoretical background. Here, vision transformers and encoder-decoder architectures will be explained. After the general overview of encoder-decoders, follows an explanation of LEDNet, the specific encoder-decoder for this study.

2.1 Vision Transformer

Vision transformer (ViT) is a new phenomenon in the world of computer vision. The ViT is based on the transformer which finds its origin in the field of natural language processing (NLP). The transformer was developed by Vaswani et al. (2017) to perform human language translation tasks. The transformer's attention mechanism distinguishes it from other language processing models. This attention mechanism is enables the transformer to learn which part of the sentence or text to focus on (Vaswani et al., 2017). Kolesnikov et al. (2021) transferred this technique to the world of computer vision. The attention technique is used on images by splitting up an image in patches of 8×8 or 16×16 pixels and dividing attention over these patches (Kolesnikov et al., 2021).

The ViT used in this study is adapted from Saavedra-Ruiz et al. (2022). A visual representation of this ViT can be seen in Figure 2.1. This ViT utilizes a self-supervised, pretrained mechanism called DINO (self-**di**stillation with **no** labels). DINO is a self-supervised learning method that uses a teacher network to predict the output of an image. The student network then learns from its own predictions without any labeled data (Caron et al., 2021). By using a pretrained weights of DINO, the model requires very little data to train.

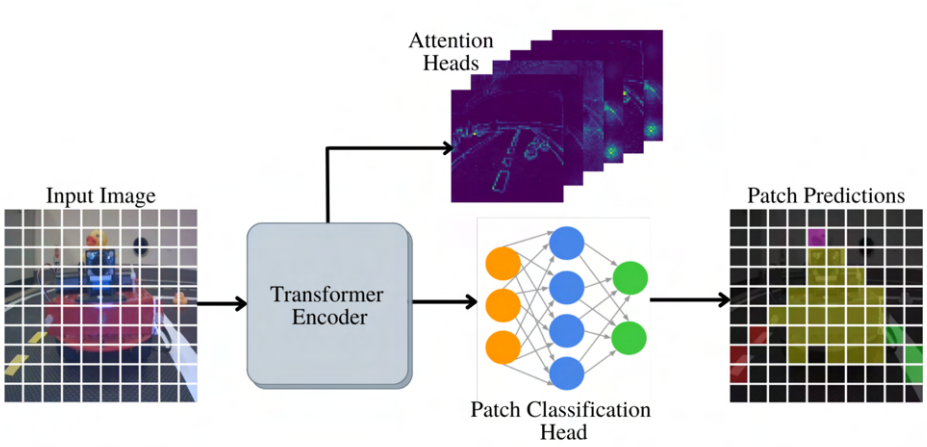
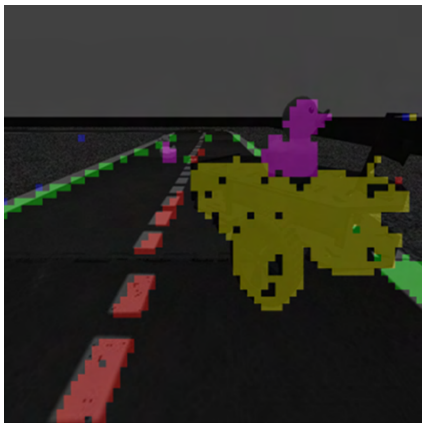


Figure 2.1: Vision Transformer. Image from Saavedra-Ruiz et al.(2022)

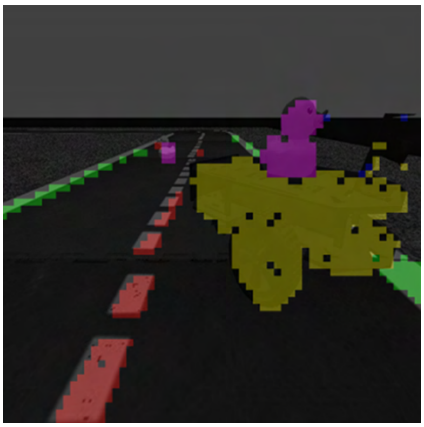
To further address the challenge of training with limited data, Saavedra-Ruiz et al. (2022) incorporated standard data augmentation techniques into the training sequence. These augmentation techniques maximize the usage of the available data. Data augmentation techniques used are random crops, flips, shifts, scales, rotations, color jittering, and Gaussian blur.

The ViT used in this study creates image segmentations by splitting up the input image in patches of 8×8 pixels. These patches are then passed on to the attention head, where it will be decided on which patch to focus on. The outputs from the attention heads are then fed into a fully connected network which is responsible for predicting the coarse segmentation mask. A visual representation of this model is shown in Figure 2.1. Each ViT can contain a different amount of blocks. A block consists of a self-attention layer and 2 fully-connected networks. More blocks can lead to higher performance, but it also requires more computational power. Example outputs of ViT with 1 block and ViT with 3 blocks can be seen in Figure 2.2.

(a) Input Image



(b) ViT 1 Block Output



(c) ViT 3 Blocks Output

Figure 2.2: ViT Image Segmentations of Duckietown

2.2 Encoder-Decoder Architecture

Encoder-decoder architectures were initially used in the field of NLP (Sutskever, Vinyals, & Le, 2014), but in recent years these architectures were found to be useful in computer vision tasks as well (Long, Shelhamer, & Darrell, 2015). One function of encoder-decoders in computer vision is for pixelwise image segmentation. An encoder-decoder architecture consists of two main parts: an encoder and a decoder. The encoder consists of a set of convolutional layers, which extract the information from the input image. The decoder consists of a set of transposed convolutional layers that upsample the feature maps that were produced by the encoder. By upsampling these feature maps, the decoder creates a pixelwise segmentation of the input image. An example of an encoder-decoder architecture can be seen in Figure 2.3.

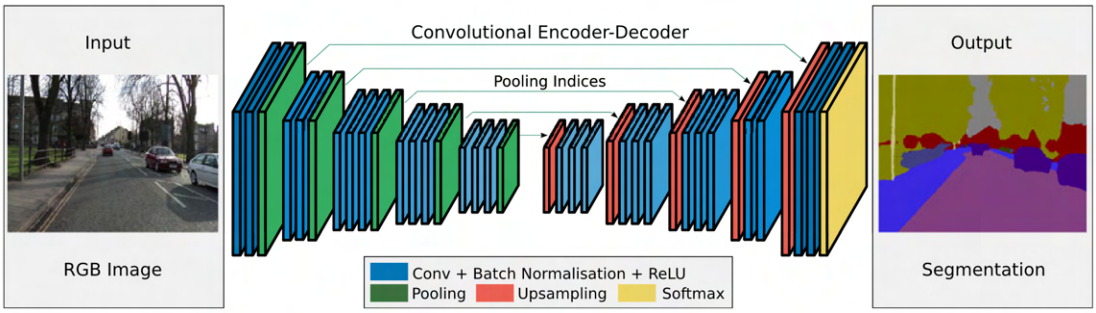


Figure 2.3: Encoder-Decoder Architecture (SegNet) for Pixelwise Segmentation (Badrinarayanan et al., 2017).

Training data for an encoder-decoder consists of pairs of input images and output images. In the case of image segmentation, these output images are matrices of the same size as the input images. Every instance of this matrix is an integer, and each integer refers to a class. Training an encoder-decoder is done by performing a process called backpropagation.

2.2.1 LEDNet

The encoder-decoder architecture selected for this study is LEDNet (Y. Wang et al., 2019). LEDNet stands for **L**ightweight **E**ncoder-**D**ecoder **Net**work. A visual representation of LEDNet can be seen in Figure 2.4. Encoder-decoders are known for being computationally demanding, because they contain a lot of convolutional layers. In order to reduce the computational burden of the encoder-decoder architecture while maintaining its high accuracy, Y. Wang et al. (2019) made modifications to the residual block (He, Zhang, Ren, & Sun, 2016). Pointwise convolutions demand the most computational power. At the beginning of the residual block the input channels are split evenly, and to avoid pointwise convolutions, a set of 1D convolutions are used (Y. Wang et al., 2019). Next, the outputs of the convolutions of both branches are concatenated. Before continuing all the channels are shuffled. Y. Wang et al. (2019) calls this residual block the split-shuffle-non-bottleneck (SS-nbt).

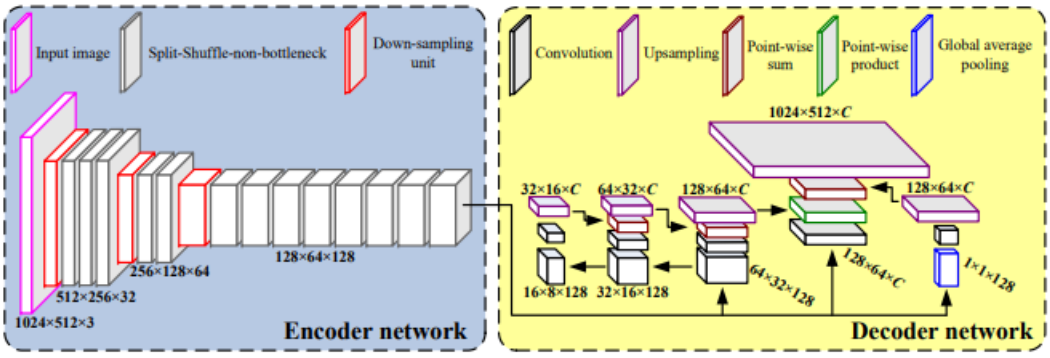
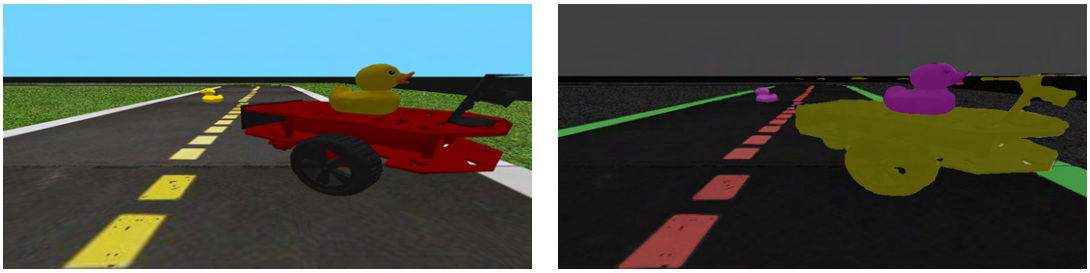


Figure 2.4: The LEDNet Architecture (Y. Wang et al., 2019).

The decoder part of LEDNet uses a design that is asymmetric and sequential. The decoder uses an attention pyramid network (APN) to upscale the feature maps created by the encoder to match the original input resolution (Hu, Shen, & Sun, 2018). This module combines features from different scales by using convolutions of different sizes (3×3, 5×5, and 7×7), which creates a multi-scale feature pyramid. This pyramid structure progressively combines information from different scales, allowing for more accurate inclusion of context from neighboring scales. To further improve performance, a global average pooling branch is added to incorporate global context before applying attention. Finally, an upsampling unit is used to match the resolution of the input image. By using the pyramid architecture, the decoder effectively captures context cues at multiple scales and produces attention maps at the pixel level for convolutional features, leading to the improved overall performance of LEDNet (Y. Wang et al., 2019).



(a) Input Image (b) LEDNet Output

Figure 2.5: LEDNet Image Segmentation of Duckietown

Chapter 3

Method

This chapter provides a detailed explanation of the approach employed in this study to achieve the research objectives. This chapter first discusses the datasets, data augmentations, and how LEDNet and the ViTs are trained using these datasets. Furthermore, it describes the algorithm used for lane-following and object-avoidance, and how LEDNet and the ViTs are incorporated into this algorithm. The chapter concludes by explaining how the performance of the two types of segmentation models is evaluated.

3.1 Dataset

The dataset utilized in this research study was sourced from Saavedra-Ruiz et al. (2022) and is openly accessible[1]. The dataset is split up in two parts; The first part contains data from the real world, which are images taken from the perspective of the real robot, while the second part contains images gathered from the Duckietown simulator. The real-world dataset consists of a total of 100 images, which are further partitioned into 70% for training, 15% for testing, and 15% for validation purposes. Additionally, the simulator dataset comprises a total of 2500 images. These images are distributed among the various subsets as follows: 70% for training, 20% for testing, and 10% for validation purposes. Because this research solely takes place in a simulated environment, only the simulator dataset is used.

[1] https://github.com/sachaMorin/dino

3.1.1 Data Augmentation

In contrast to ViTs, encoder-decoder architectures necessitate a considerably larger volume of data. Given the constraints of time associated with capturing additional images and manually annotating them, the adoption of data augmentation was chosen as a means to enlarge the existing dataset of 2500 images. By horizontally flipping all the images, the size of the dataset was doubled. In order to assess the influence of the data augmentation on the model accuracy, an evaluation method known as intersection over union (IoU) was used. IoU is the standard measurement technique to evaluate the efficacy of image segmentation models. The IoU is calculated by dividing the intersection area of the predicted segmentation and the ground truth by their union area (see Equation 3.1).

$$IoU = \frac{|A \cap B|}{|A \cup B|}. \tag{3.1}$$

This data augmentation increased the overall IoU score of the LEDNet encoder-decoder by almost 4% (see Table 3.1), with the largest improvement for the Duckiebot class. A comparison of the loss between the LEDNet model trained on the augmented dataset and the LEDNet model trained on the original dataset is presented in Appendix B.

Class	Precision (IoU)	
	Augmented Dataset	Original Dataset
background	**99.40**	99.18
yellow-line	**88.18**	85.17
white-line	**94.38**	92.56
duckiebot	**86.43**	79.16
sign	**89.40**	84.82
duck	**90.22**	84.77
Average	**91.13**	87.63

Table 3.1: LEDNet IoU comparison

As mentioned before, the ViT used in this study already incorporates data augmentations in its training sequence. Nonetheless, the augmented dataset was also tested on the ViT to see if there was an increase in performance. This was not the case and for the rest of this research, the standard dataset will be used for the ViT. It is worth mentioning that the ViT's consistent performance proves

its robustness, indicating its resistance to overfitting. See Appendix A for the IoU comparison between the ViT trained on the standard dataset and the augmented dataset.

3.2 Model Training

As observed in Section 3.1.1, the augmented dataset contributes significantly to the performance enhancement of LEDNet. Therefore, for the rest of the study, the LEDNet trained on the augmented dataset will be used. The decision had been made to train LEDNet using 200 epochs and a batch size of 5. The decision to train LEDNet involved using a batch size of 5, which is standard for LEDNet, along with 200 epochs (Y. Wang et al., 2019). The dataset is relatively simple, where the road and obstacles exhibit consistent colors, lighting, and shapes throughout, and therefore 200 epochs is enough. In addition, the loss graph starts to flatten around the 200-epoch mark, further proving that 200 epochs is sufficient (see Fig. 3.1).

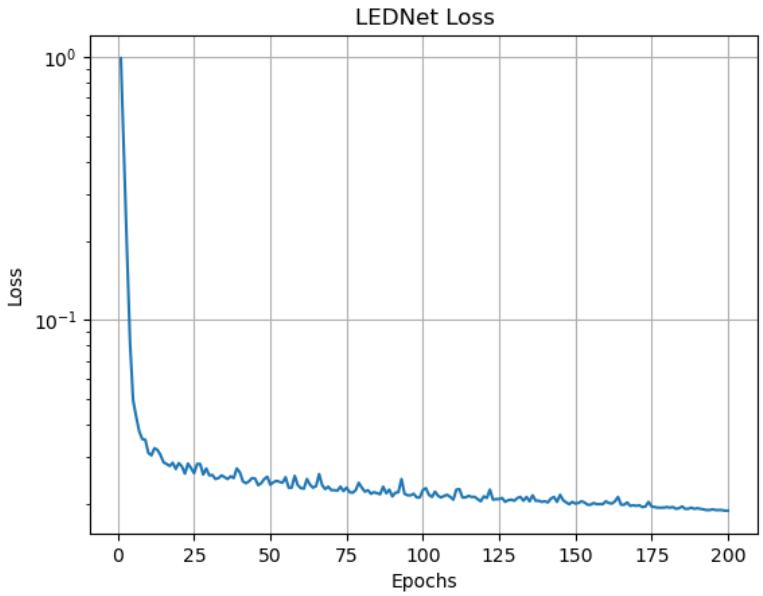


Figure 3.1: LEDNet Loss Graph

The ViT on the other hand is trained using the original dataset, since the augmented dataset does not lead to any significant performance increase (see Appendix A). The ViT is trained with the same configuration as Saavedra-Ruiz et al. (2022); 200 epochs and a batch size of 1.

3.3 Duckietown Simulator

This research takes place in the Duckietown Simulator[2]. This simulator is a Python package that uses OpenAI's Gym Python package. The Duckietown simulator enables the execution of code intended for the real Duckiebot within a simulated environment.

3.3.1 Maps

Within the Duckietown simulator, the map "loop_empty" (see Fig. 3.2) will be the designated environment for this study. This map, despite its simplicity, contains both left and right bends. Due to the limitations of the lane-following algorithm (see Section 3.4) in handling crossroads and junctions, these will be excluded from consideration.

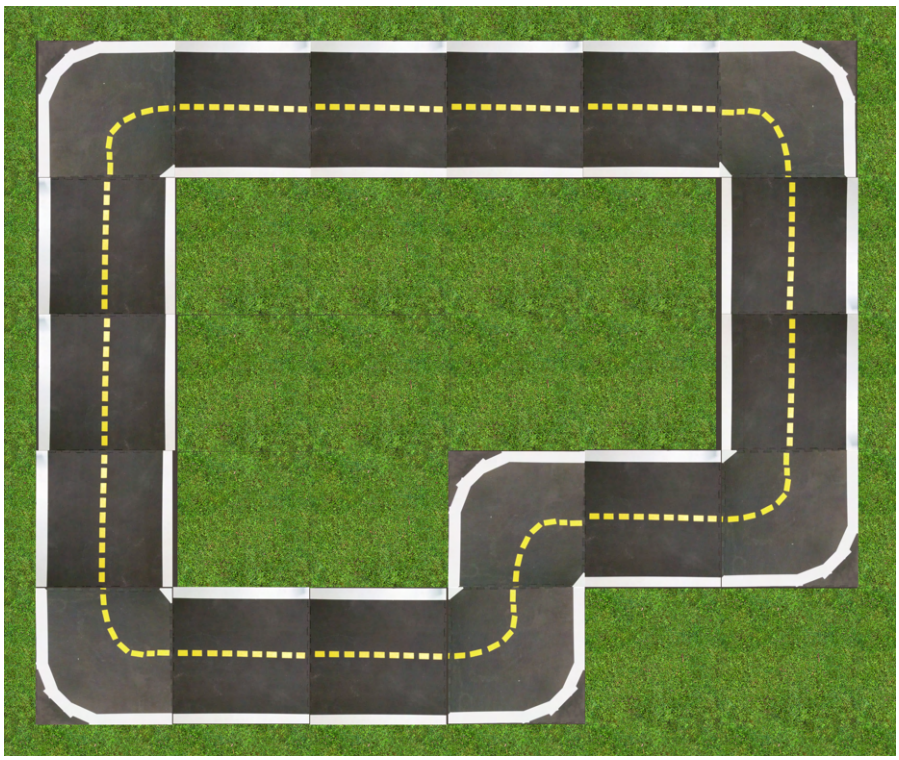


Figure 3.2: Simulated Duckietown Map: "loop_empty"

Another map utilized in this study shares the same configuration as the map depicted in Figure 3.2. However, this map has several obstacles. There are rubber

[2] https://github.com/duckietown/gym-duckietown

ducks and Duckiebots spread out over the map (see Fig. 3.3). Using this setup, the obstacle-avoidance performance of the LEDNet and ViT Duckiebots are tested.

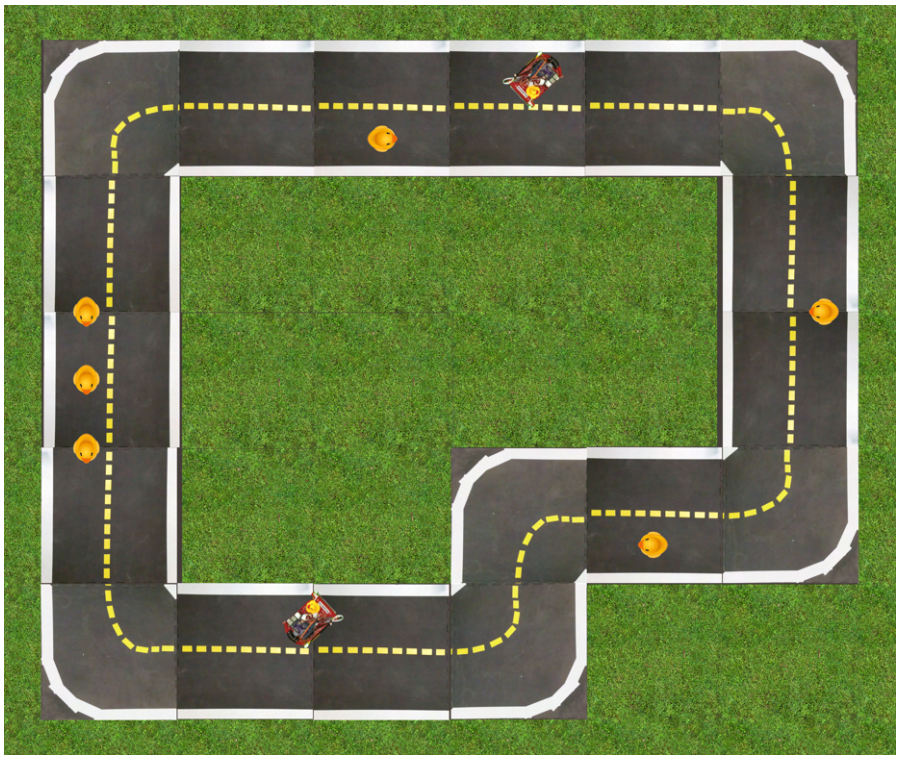


Figure 3.3: Simulated Duckietown Map: "loop_empty", with obstacles like: Duckiebots and Rubber Ducks.

3.4 Lane-Following and Obstacle-Avoidance Algorithm

This section explains the lane-following and obstacle-avoidance algorithm. This algorithm was sourced from Saavedra-Ruiz et al.(2022)[3]. A visual representation of this algorithm can be seen in Figure 3.4. This algorithm uses real-time generated image segmentations to compute the optimal steering direction for the Duckiebot, enabling it to maintain its position within the lane, evade obstacles or steer correctly around bends. However, it is important to note that this algorithm does not include a mechanism to recognize crossroads or junctions.

[3] https://github.com/MikeS96/object-detection/tree/daffy

The Duckietown simulator generates images from the perspective of the Duckiebot at a rate of 30 fps, with each image having dimensions of 640×480 pixels. In order to perform segmentations using various models, these images must undergo a reshaping process. Specifically, for the LEDNet model, the image needs to be adjusted to 1024×512 pixels, while for the ViT model, the image must be of shape 480×480 pixels. Once reshaping is done, the image is then fed into the segmentation model. The output of the model is the segmentation matrix $S_t \in \mathbb{Z}$; with similar height and width as the input image. For the rest of the algorithm, the size of the segmentation matrix S_t needs to be of shape 480×480, so only the segmentation matrix created by LEDNet needs to be reshaped.

Every integer value in S_t corresponds to a class, in this case: 0 corresponds to the background, 1 to the yellow line in the center, 2 is the white line, 3 is a Duckiebot, 4 is a sign and 5 is a (rubber)duck.

Next, the matrix M_t is created. This matrix has the same shape as S_t, but in M_t, all the labels corresponding to the white lines are replaced with 1 and all the labels corresponding to obstacles (Duckies and Duckiebots) are replaced with 2. Then, the steering direction is calculated as follows:

$$\phi_{t+1} = \phi_t - \gamma \sum_{i,j} (P \odot M_t)_{i,j}. \tag{3.2}$$

Here P is a matrix with -1 values in the left half of the matrix and 1 values in the other half. This equation balances the amount of obstacle and road marking pixels on the left and right sides of the screen. Consequently, when the Duckiebot encounters a right-turning bend, the road markings on the right side of the screen will progressively disappear from its view. As a result, the Duckiebot steers towards the right in order to restore the balance within the matrix. The higher weight of the obstacles, compared to road markings, in the matrix M_t cause the Duckiebot to steer around the obstacles to avoid crashing into them.

It is important to note that this lane-following and obstacle-avoidance algorithm maintains a static driving speed throughout, even when navigating bends. The algorithm focuses on optimizing the steering direction to keep the Duckiebot within the lane and avoid obstacles, while the speed remains constant. The decision to maintain a static driving speed is intentional to simplify the algorithm's implementation and reduce complexity.

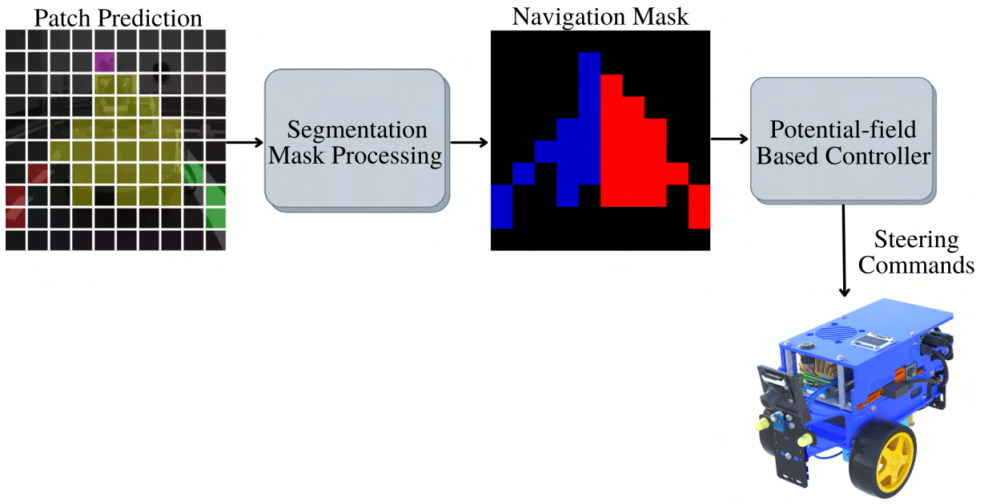


Figure 3.4: Lane-Following and Object-Avoidance Algorithm. Image from Saavedra-Ruiz et al. (2022)

3.5 Evaluation Methods

Before explaining the evaluation methods, it is important to note that all the experiments are conducted on a laptop with a *GeForce GTX 1050M* GPU, *Intel Core i7-8565U* CPU, and 16 GB of RAM.

There are several ways to assess the performance between the LEDNet-driven and ViT-driven Duckiebot. To start, the maximum speed with which the Duckiebot is able to successfully drive around the map without obstacles is measured. The Duckiebot's speed is visible in the simulator's interface. This method also measures the Duckiebot's maximum cornering speed, since the algorithm keeps the Duckiebot at a static speed, even in corners. A successful circuit around the map is defined as the Duckiebot remaining within the designated track and avoiding any 180-degree turns at bends. However, the latter occasionally occurs when the Duckiebot is driving too fast, resulting in the processing of the frame indicating the need to halt the turning motion being rendered too late. Consequently, the Duckiebot ends up reversing its course, resulting in a failed attempt. To ensure stability, the Duckiebot needs to complete four consecutive circuits around the course before the speed is increased. The speed is increased until the Duckiebot fails.

In addition, the obstacle-avoidance performance of both the LEDNet and ViT Duckiebot is compared. The same map will be used, but this time there are ob-

stacles like rubber ducks and stationary Duckiebots on the road that need to be avoided in order to finish the course (see Fig. 3.3). In order to assess and compare the performance, the ability of the Duckiebot to navigate the course four consecutive times without making contact with any obstacles or deviating from the designated road will be evaluated. The driving speed is fixed at 0.07 m/s.

Likewise, in order to verify the hypothesis that LEDNet can detect smaller obstacles more effectively due to its higher resolution, the image segmentations generated by LEDNet and ViT for images containing a small obstacle will be compared. This comparison will help determine if LEDNet is capable of identifying obstacles that may be missed by ViT.

Finally, to see if the visibility of small obstacles in the segmentations has any effect on the obstacle-avoidance, experiments will be conducted where the Duckiebot navigates around an obstacle. Again, to ensure stability, only after four consecutive successful attempts the size of the obstacle will be manually decreased. This repeats until an unsuccessful attempt occurs. The comparison between LEDNet and ViT Duckiebot will focus on identifying the smallest obstacle size that each system is capable of navigating around successfully.

Chapter 4

Results

In this chapter, the results from the evaluation methods (see Section 3.5) are shown. These results provide insight into the performance of the LEDNet-driven Duckiebots and the ViT-driven Duckiebots. This chapter starts by showing the maximum speed with which the different models can navigate the course (see Section 4.1). Furthermore, the obstacle-avoidance performance was tested (see Section 4.2). Additionally, the chapter explores the hypothesis regarding LEDNet's ability to detect small obstacles (see Section 4.3). Lastly, the performance of the models in navigating around obstacles of varying sizes is evaluated (see Section 4.4).

4.1 Maximum Speed

The first evaluation metric aimed to measure the maximum speeds at which the LEDNet and ViT models were able to successfully complete the map (see Fig. 3.2). The results are summarized in Table 4.1.

Model	**Maximum Speed** (m/s)
LEDNet	**0.15**
ViT 1 block	0.13
ViT 3 blocks	0.09

Table 4.1: Maximum Speeds Achieved by Different Models

The LEDNet model demonstrated the highest maximum speed of 0.15 m/s, followed by the ViT 1 block model with a maximum speed of 0.13 m/s. The ViT 3 block model achieved a slightly lower maximum speed of 0.09 m/s. These

results indicate that the LEDNet model outperformed the ViT models in terms of the maximum speed achieved on the map.

4.2 Obstacle-Avoidance

The second evaluation metric aimed to assess the obstacle-avoidance performance of the LEDNet and ViT models. In this evaluation, all three Duckiebots were required to navigate the map containing obstacles (see Fig. 3.3) in both clockwise and counterclockwise directions. The results are visible in Table 4.2, where the symbol ✓ signifies the successful completion of the course, while the symbol × indicates it failed.

Model	Counter-clockwise	Clockwise
LEDNet	✓	✓
ViT 1 block	✓	×
ViT 3 blocks	×	×

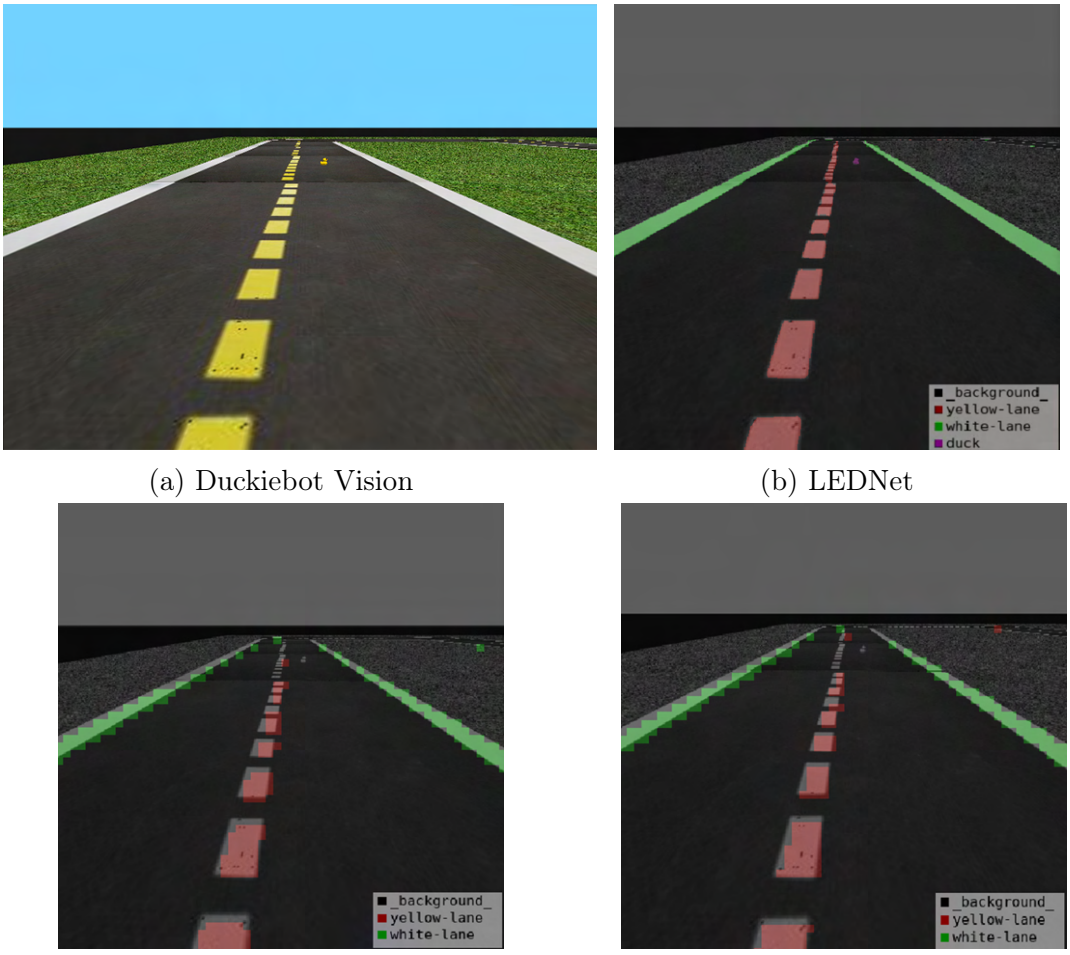
Table 4.2: Ability to Drive Around Obstacles on Map (see Fig 3.3)

The LEDNet-driven Duckiebot was able to drive around the course without touching any obstacles. The ViT with 1 block was only able to successfully complete the course driving counterclockwise. Finally, the ViT with 3 blocks was unable to safely complete the course without touching any obstacles.

The ViT with 1 block only failed to drive around the stationary Duckiebot positioned at the bottom part of the map (see Fig. 3.3) when it was on a clockwise attempt. The ViT with 3 blocks on the other hand showed unpredictable behaviour. It occasionally manages to circumvent certain obstacles, while other times failing to do so. Furthermore, it occasionally overcorrected navigating around an obstacle and drives off the road.

4.3 Small Obstacle Visibility

The third evaluation metric aimed to verify the hypothesis that LEDNet is able to detect more detail and spot smaller obstacles. The results are shown in Figure 4.1.



(a) Duckiebot Vision

(b) LEDNet

(c) ViT 1 Block

(d) ViT 3 Blocks

Figure 4.1: Image Segmentations Created by LEDNet, ViT 1 Block, ViT 3 Blocks

The Duckiebot's vision is shown in Figure 4.1a. A small rubber duck is visible on the right lane. The legend, as seen in Figures 4.1b, 4.1c and 4.1d, updates automatically based on what is visible in the image according to the image segmentation model. The presence of the class *duck* in the legend in Figure 4.1b indicates that the small rubber duck was visible to LEDNet. The absence of this class in the legend of the remaining Figures 4.1c and 4.1d indicate that the rubber duck was not visible to the ViTs.

Furthermore, it is noteworthy that the LEDNet model was able to accurately classify the road markings throughout the entire street. In contrast, the ViTs started to omit parts of the road markings that were further down the road.

4.4 Small Obstacle Avoidance

The last evaluation metric focused on the ability of the LEDNet and ViT models to detect and navigate around small obstacles of varying sizes. Table 4.3 summarizes the results.

Obstacle Size (cm)	**LEDNet**	**ViT 3 blocks**	**ViT 1 block**
8	✓	✓	✓
7	✓	✓	✓
5	✓	✓	✓
4	✓	×	✓
3	✓	×	✓
2	✓	×	×
1	×	×	×

Table 4.3: Ability of Models to Navigate Around Obstacles of Different Sizes

The results demonstrate that all the models were capable of successfully navigating around obstacles of sizes 8 cm, 7 cm, and 5 cm. However, the ViT 3 block model failed to navigate around the obstacles of size 4 cm and smaller. The LEDNet model exhibited the highest performance, successfully navigating around all obstacles of size 2 cm and larger. The smallest obstacle the ViT 1 block managed to drive around is 3 cm.

Chapter 5

Discussion

This chapter aims to provide an analysis of the results (see Chapter 4) obtained by evaluating the LEDNet-driven Duckiebot and the ViT-driven Duckiebots. These results contain the maximum speed with which every model is able to navigate the course, obstacle-avoidance performance, and small obstacle visibility and avoidance. Examining these results will yield proof for the hypothesis that the LEDNet-driven Duckiebot is able to navigate more accurately and stably in comparison to the ViT-driven Duckiebots.

5.1 Evaluation

All the results from Chapter 4 show that the high-resolution encoder-decoder architecture LEDNet-driven Duckiebots outperforms the ViT-driven Duckiebots. Notably, the LEDNet Duckiebot exhibits higher cornering speeds, attributed to its higher frame rate and high resolution. A higher frame rate leads to the earlier rendering and processing of the frame indicating the completion of the Duckiebot's turn. This allows the Duckiebot to timely recognize the end of the turn and transition to straight driving, enabling it to go through corners and around obstacles at increased speeds. The high resolution enables the LEDNet to accurately balance the left and right lane markings, keeping the Duckiebot in the lane.

Moreover, LEDNet's performance in terms of obstacle-avoidance is also higher compared to the ViTs. As shown by the results (see Section 4.2), the LEDNet Duckiebot was the only one able to navigate around the map with obstacles (see Fig. 3.3) both clockwise as well as counter-clockwise without touching any of the obstacles. This has to do with the higher resolution and higher accuracy of the LEDNet model. It was able to classify the entirety of the obstacles, causing the algorithm to accurately steer around the obstacle. The ViTs on the other hand,

sometimes failed to classify the entirety of the obstacles, especially the stationary Duckiebots, resulting in a collision with the unclassified part of the obstacle. On some occasions, the Duckiebot was able to classify the entire obstacle, but often this was only the case when the obstacle was too close to steer around. The mediocre performance of the ViT with 3 blocks in obstacle-avoidance is caused by its low frame rate, which results from the computational demands of the use of three transformer blocks.

Furthermore, the high resolution and accuracy of the LEDNet encoder-decoder also enabled the model to accurately detect small obstacles where the ViTs cannot (see Fig. 4.1). In contrast, the ViTs adopted an 8×8 pixel patch labeling approach. Therefore, if a patch does not contain enough pixels of the obstacle, it will not be labeled with the label of the obstacle. In fact, even if the obstacle is 4 times larger, its pixels could be distributed over 4 separate 8×8 pixel patches, and neither of those patches contain enough pixels of the obstacle to be labeled correctly. As a consequence, the obstacle was invisible to the ViT.

It is worth mentioning that LEDNet did not confuse the small rubber duck on the road for the yellow road markings. This occasionally happened with the ViTs when avoiding obstacle.

Evidently, the LEDNet proved to be the best driving around small objects (see Section 4.4). However, it must be said that the ViT with 1 block showed better results than anticipated. Even though the ViT was not able to identify the rubber duck from afar, it became visible when the Duckiebot was close enough. The rubber duck of size 2 cm was detected by the ViT too late to avoid a collision.

A limitation of this study is that it relied on a simulated environment. Within the Duckietown simulator all the colors, lighting, and objects are very consistent, and behaviour of obstacles is predictable. In contrast, the real world presents a greater degree of randomness and unpredictability in these aspects. However, as all three Duckiebots operated within this environment, the relative performance quality is expected to remain relatively consistent if the algorithm utilizing these segmentation models were to be applied in the real world.

Chapter 6

Conclusion

This chapter will aim to answer the research question of this thesis: *How does the performance of the LEDNet encoder-decoder architecture compare to the vision transformer-based approach in terms of monocular robot navigation, lane-following accuracy, and obstacle avoidance?*

In conclusion, LEDNet outperformed the ViTs in terms of monocular robot navigation, lane-following accuracy, and obstacle avoidance. The results clearly support this conclusion by showing that the LEDNet-driven Duckiebot is able to navigate its course safely with higher speeds compared to the ViTs. It is more accurate at recognizing bends as well as recognizing obstacles and driving around them. Furthermore, LEDNet showed outstanding performance in detecting small obstacles. It was also better at guiding the Duckiebot around these small obstacles without colliding. These outcomes collectively highlight the remarkable advantages offered by LEDNet in the context of monocular robot navigation, reinforcing its efficacy for enabling precise lane-following and efficient obstacle avoidance.

6.1 Future Work

This study can be improved by running the algorithm on a real-world Duckiebot and making it drive through real-world Duckietown, rather than relying solely on a simulator. This would allow for testing the performance of LEDNet and the ViTs in a more real-world environment. However, since the real-world dataset[1] is so small, this needs to be expanded first in order for LEDNet to work properly.

In the future, the lane-following and obstacle-avoidance algorithm could be further improved by adding a speed regulator for when the Duckiebot is going through bends or navigating around obstacles. For simplicity, the algorithm now

keeps the Duckiebot at a constant speed. However, it would be more realistic to slow down for bends and obstacles. For example, slowing down allows the algorithm to accurately respond to the possible obstacles around the bend that were not visible before going through the bend.

Moreover, the current lane-following algorithm is unable to handle junctions and crossings. This issue can be addressed by training LEDNet to recognize traffic signs, traffic lights, and lines that indicate junctions or crossings. Consequently, adapting the algorithm to appropriately handle these newly recognized classes within the segmentation masks enables the Duckiebot to navigate through these situations successfully.

References

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, *39*(12), 2481–2495.

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., & Joulin, A. (2021). Emerging properties in self-supervised vision transformers. In *Proceedings of the ieee/cvf international conference on computer vision* (pp. 9650–9660).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 770–778).

Holder, C. J., & Shafique, M. (2022). On efficient real-time semantic segmentation: a survey. *arXiv preprint arXiv:2206.08605*.

Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-excitation networks. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 7132–7141).

Kolesnikov, A., Dosovitskiy, A., Weissenborn, D., Heigold, G., Uszkoreit, J., Beyer, L., ... Zhai, X. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the 9th international conference on learning representations (iclr)*.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 3431–3440).

Saavedra-Ruiz, M., Morin, S., & Paull, L. (2022). Monocular robot navigation with self-supervised pretrained vision transformers. In *2022 19th conference on robots and vision (crv)* (pp. 197–204).

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, *27*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N.,

... Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Wang, Y., Zhou, Q., Liu, J., Xiong, J., Gao, G., Wu, X., & Latecki, L. J. (2019). Lednet: A lightweight encoder-decoder network for real-time semantic segmentation. In *2019 ieee international conference on image processing (icip)* (pp. 1860–1864).

Wang, Z., Ren, W., & Qiu, Q. (2018). Lanenet: Real-time lane detection networks for autonomous driving. *arXiv preprint arXiv:1807.01726*.

Appendix A

Vision Transformer Trained on Augmented Dataset

Class	Precision (%)	
	Augmented Dataset	Original Dataset
background	97.11	**97.17**
yellow-line	**46.97**	42.20
white-line	62.43	**64.18**
duckiebot	10.04	**10.65**
sign	27.74	**31.63**
duck	**31.12**	30.68
Average	64.73	**65.00**

Table A.1: ViT 3 Blocks

Class	Precision (%)	
	Augmented Dataset	Original Dataset
background	**97.02**	97.01
yellow-line	**46.90**	41.98
white-line	62.19	**63.86**
duckiebot	**8.16**	8.09
sign	13.67	9.51
duck	**28.51**	26.28
Average	**64.06**	63.85

Table A.2: ViT 1 Block

Appendix B

LEDNet Loss Graph Comparison

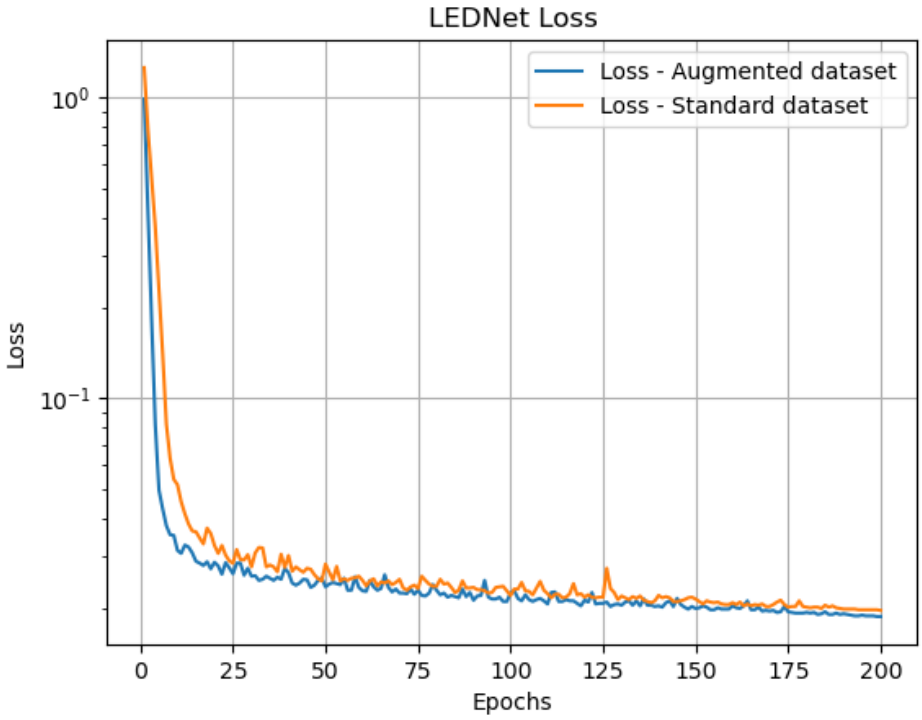


Figure B.1: LEDNet Loss Graph