

ROAD FEATURES CLASSIFICATION
USING SEMANTIC IMAGE
SEGMENTATION

SIVARAJ GHANESH A/L THANASAGAR

1151103260

2020/2021

MULTIMEDIA UNIVERSITY

SEPTEMBER 2020

**ROAD FEATURES CLASSIFICATION
USING SEMANTIC IMAGE
SEGMENTATION**

by

SIVARAJ GHANESH A/L THANASAGAR

Session 2020/2021

The project report is prepared for
Faculty of Engineering and Technology
Multimedia University
in partial fulfilment for
Bachelor of Engineering (Hons) MECHANICAL

FACULTY OF ENGINEERING AND TECHNOLOGY
MULTIMEDIA UNIVERSITY

September 2020

© 2020 Universiti Telekom Sdn. Bhd. ALL RIGHTS RESERVED.

Copyright of this report belongs to Universiti Telekom Sdn. Bhd as qualified by Regulation 7.2(c) of the Multimedia University Intellectual Property and Commercialisation Policy. No part of this publication may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Universiti Telekom Sdn. Bhd. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

DECLARATION

I hereby declare that this work has been done by myself and no portion of the work contained in this report has been submitted in support of any application for any other degree or qualification of this or any other university or institute of learning.

I also declare that pursuant to the provisions of the Copyright Act 1987, I have not engaged in any unauthorised act of copying or reproducing or attempt to copy / reproduce or cause to copy / reproduce or permit the copying / reproducing or the sharing and / or downloading of any copyrighted material or an attempt to do so whether by use of the University's facilities or outside networks / facilities whether in hard copy or soft copy format, of any material protected under the provisions of sections 3 and 7 of the Act whether for payment or otherwise save as specifically provided for therein. This shall include but not be limited to any lecture notes, course packs, thesis, text books, exam questions, any works of authorship fixed in any tangible medium of expression whether provided by the University or otherwise.

I hereby further declare that in the event of any infringement of the provisions of the Act whether knowingly or unknowingly the University shall not be liable for the same in any manner whatsoever and undertake to indemnify and keep indemnified the University against all such claims and actions.

Signature: SIVARAJ GHANESH

Name: SIVARAJ GHANESH A/L THANASAGAR

Student ID: 1151103260

Date: 28-9-2020

ACKNOWLEDGEMENT

First of all, I would like to thank my supervisor Dr Em Poh Ping for his constant guidance and encouragement throughout this project. This project will certainly not be able to be completed without him.

Secondly, I would like to thank my moderator, Mr. Alif Zulfakar Bin Pokaad for his advice during the presentation in order to improve this project.

Thirdly, I would like to thank Multimedia University (MMU) for providing the necessary academic resources to conduct my research for this project.

Lastly, I would like to thank my parents for their financial assistance that provides me with the opportunity to study in this university.

ABSTRACT

This project is aimed to contribute to the advancement of Machine Learning in autonomous vehicle technology implementations. Thus, based on the Literature Review phase, the problem statements encountered by the authors were put into three main categories, namely Low Accuracy, Underutilization of datasets, and High Computation time. Hence, three objectives were set in order to guide this project towards solving these three problem statement themes within the scope of the researcher. The first objective is to identify a suitable road footage dataset. As such, the first objective has been completed with the selection of the Microsoft Common Objects In Context (COCO) dataset. The second objective is to improve on a pixel-level neural network classification model using semantic segmentation. This project aimed at exploiting the fine-tuning of hyperparameter configurations of a Mask R-CNN framework. Due to the lack of computation power, the initial control set that is used as the control in this project is dialled down. Thus, any improvements proposed were based on this control set. The final objective is to benchmark the proposed method with state-of-the-art methods. The mean Average Precision achieved with the modified hyperparameter configuration is 36.95 % while that of the original Mask R-CNN framework is 38.2 %.

TABLE OF CONTENTS

COPYRIGHT	ii
DECLARATION.....	iii
ACKNOWLEDGEMENT	iv
ABSTRACT	v
TABLE OF CONTENTS.....	vi
LIST OF FIGURES	ix
LIST OF TABLES	xi
CHAPTER 1: INTRODUCTION.....	1
1.1 Background	1
1.2 Project Motivation	1
1.3 Problem Statement	2
1.4 Project Gap	3
1.5 Project Objectives.....	3
1.6 Project Scope	3
1.7 Project Contributions.....	5
1.8 Project Gantt Chart	5
1.9 Project Flow Chart.....	9
1.10 Report Organization.....	11
CHAPTER 2: LITERATURE REVIEW.....	12
2.1 Overview	12
2.2 Semantic Segmentation	13

2.2.1	Convolutional Neural Network.....	15
2.2.2	Generative Adversarial Network.....	17
2.2.3	Auto Encoder	18
2.3	Instance Segmentation.....	19
2.3.1	You Only Look Once	19
2.3.2	Faster R-CNN.....	20
2.4	Datasets	21
2.4.1	CityScapes Dataset.....	21
2.4.2	CamVid Dataset	22
2.4.3	Microsoft COCO Dataset.....	22
2.5	Summary	23
CHAPTER 3: METHODOLOGY.....		33
3.1	Overview	33
3.1.1	ResNet Backbone Network.....	33
3.1.2	Feature Pyramid Network	34
3.1.3	Region Proposal Network	35
3.1.4	ROIAlign.....	36
3.1.5	Segmentation Masks	37
3.2	Implementation.....	38
3.2.1	Training.....	39
3.2.2	Validation.....	39
3.3	Configurations	41
3.4	Transfer Learning	42
3.5	Microsoft COCO Dataset	42

CHAPTER 4: RESULTS AND DISCUSSION	44
4.1 Hyperparameter fine-tuning	44
4.1.1 Detection_Min_Confidence	46
4.1.2 Iterations.....	46
4.1.3 Learning Rate	47
4.1.4 Detection_Non-Max Supression_Threshold.....	47
4.1.5 Rpn_Anchor_Scales.....	48
4.1.6 ROI_Positive_Ratio	48
4.1.7 Rpn_Train_Anchors_Per_Image and Train_Rois_Per_Image	48
4.2 Benchmarking	53
CHAPTER 5: CONCLUSION.....	56
5.1 Conclusion.....	56
5.1.1 Achieved Objectives	56
5.1.2 Project Limitations.....	56
5.1.3 Recommendations	57
REFERENCES.....	58
WEBSITE REFERENCES	60

LIST OF FIGURES

Figure 1.1:An image of a street before and after semantic segmentation and classification	1
Figure 1.2: Problem statement topics addressed in [2-12], [14-15], [17-22], [26]	2
Figure 1.3: FYP Part 1 Flow Chart	10
Figure 1.4: FYP Part 2 Flow Chart	10
Figure 2.1: Mind map of image segmentation	12
Figure 2.2: Image of handwritten digits before and after segmentation	14
Figure 2.3: Road scene image passing through semantic segmentation and classification processes	14
Figure 2.4: Specific geometries used for segmenting digits	15
Figure 2.5: Three types of layers found in a Neural Network	17
Figure 2.6: A vertical edge kernel applied to a stairway image.....	17
Figure 2.7: Main components of a Generative Adversarial Network	18
Figure 2.8: Auto Encoder network.....	18
Figure 2.9: Instance segmentation demonstrated on.....	19
Figure 2.10: You Only Look Once framework	20
Figure 2.11: Faster Region-based Convolutional.....	20
Figure 2.12: Distribution of datasets used in [2-12], [14-15], [17-22], [26].....	21
Figure 2.13: Images from three datasets in [2-12], [14-15], [17-22], [26]	23
Figure 3.1: Overall Mask R-CNN framework	33

Figure 3.2: ResNet-50 extracting low-level	34
Figure 3.3: An illustration of Feature Pyramid Network as.....	35
Figure 3.4: Simplified illustration to visualize anchors	36
Figure 3.5: Visualization of bilinear interpolation.....	37
Figure 3.6: 28×28 pixels mask generated by	38
Figure 3.7: Implementation process flow.....	39
Figure 3.8: An original image and its annotated version from the Microsoft COCO Train 2017 dataset	43
Figure 4.1: Loss vs Iterations Graph for Configuration Set A-Set H	49

LIST OF TABLES

Table 1.1: Scope of this project	4
Table 1.2: FYP Part 1 Gantt Chart.....	7
Table 1.3: FYP Part 2 Gantt Chart.....	8
Table 2.1: Summary of Significance of Findings in Papers Reviewed.....	25
Table 3.1: Summary of the environment used to conduct this project.....	41
Table 3.2: Initial Hyperparameter Configurations (Set A)	42
Table 4.1: Hyperparameter configurations comparison.....	45
Table 4.2: Training Losses Comparison between Configurations Set A, Set B, Set C and Set D	50
Table 4.3: Validation Losses Comparison between Configurations Set A, Set B, Set C and Set D	50
Table 4.4: Differences between Train and Validation (Val).....	51
Table 4.5: Accuracy differences between Set A, Set C, Set D and Set H.....	52
Table 4.6: The framework used in this project is benchmarked with other state-of-the- art frameworks	53
Table 4.7: Performance Comparison of Images with Benchmarked Mask R-CNN Framework	54

CHAPTER 1: INTRODUCTION

1.1 Background

Image segmentation is a subset of computer vision. It identifies various objects and puts these objects into the classes that they belong in. This is done for all the pixels that are in an image.



(a) A street in a neighbourhood

(b) Semantically segmented
image of the street

Figure 1.1:An image of a street before and after semantic segmentation and classification [1]

Figure 1.1(a) is a normal image of a road scene while in Figure 1.1(b), objects (road features) have been classified using semantic segmentation. Each type of object (class) is colour coded. Cars are purple, the road is pink, the sidewalk-blue, buildings are red, etc.

1.2 Project Motivation

Image understanding is essential for progress in autonomous vehicle implementation. There are 5 levels of automation which are extensively covered in [2]. In order for machines to reach level 3 automation from the current level 2 [2], image

understanding has to be mastered. Semantic segmentation is the first step in image understanding [1]. Thus, the motivation of this project is to contribute to Image Understanding.

1.3 Problem Statement

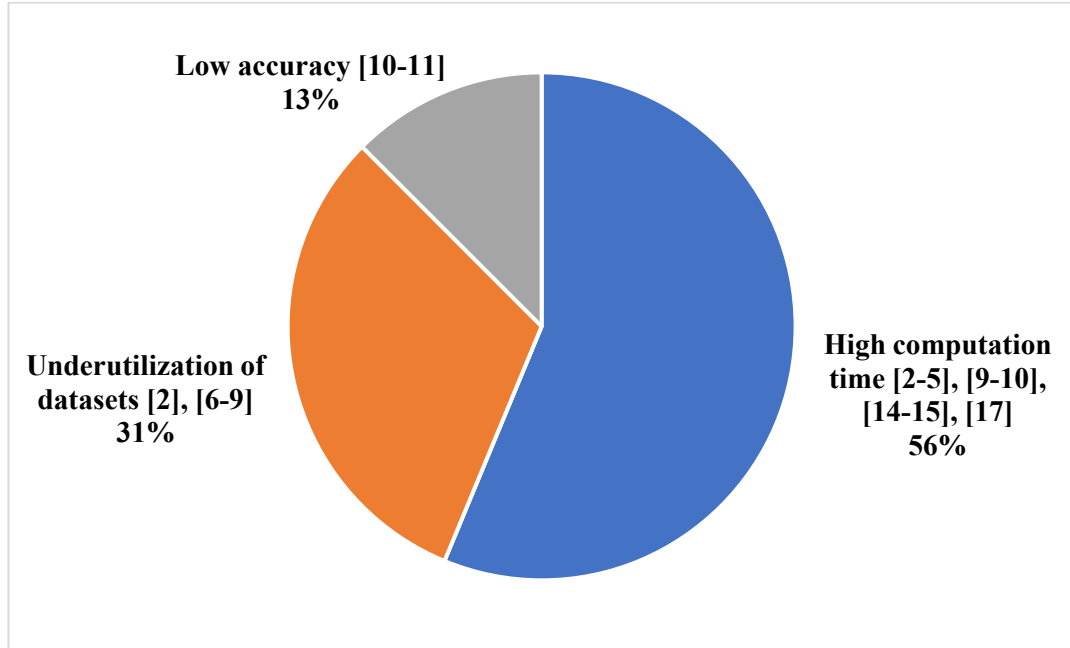


Figure 1.2: Problem statement topics addressed in [2-12], [14-15], [17-22], [26]

In Figure 1.2, a big percentage of the research work in this field focuses on improving the computation time of the Machine Learning (ML) models. Thus, this project will not be focusing on this aspect. Additionally, problems related to the underutilization of datasets have been paid the second most attention to by researchers. For example, in the underutilization of datasets theme, researchers mentioned problems such as the underrepresentation of a few classes that are specific to their use cases. It is worth noting that in these scenarios, transfer learning was not utilized. Instead, researchers trained their framework from scratch. On the other hand, researchers tended to pair problems related to the large number of parameters with either problems related to high computation time or problems related to high costs. Also, researchers tended to experiment on the ML models in specific scenarios which

rendered the publicly available datasets to lack certain road features that are prevalent to their application [1] and not others. A larger number of parameters contributes higher complexities in the model. Thus, the complexity may be a contributing factor for long computations.

1.4 Project Gap

With reference to the problem statement, the research gap identified is problems related to the complexity of ML models. Therefore, the aim of this project would be to simplify an existing ML model to be relatively simple and fast while maintaining a respectable accuracy when benchmarked with other state-of-the-art methods.

1.5 Project Objectives

Three objectives were set for this project. The first objective is to identify a suitable road footage dataset. As such, the first objective has been completed with the selection of the Microsoft Common Objects In Context (COCO) dataset. The reason this dataset was chosen is stated in the Summary section. The second objective is to improve on a pixel-level neural network classification model using semantic segmentation. The final objective is to benchmark the proposed method with state-of-the-art methods. It is worth noting that these objectives are set to bridge the gap between the problem statement and scope of this project. Additionally, the technical abilities of the researcher for this project is also taken into account when setting the objectives.

1.6 Project Scope

The entire field of semantic segmentation cannot be covered in this project as it is too vast. Therefore, the information explored and reviewed in this project will be constrained to that available in the academic papers cited in the Reference section.

The amount of time available for this project is spread across 28 weeks. Thus, it does not permit the development of a machine learning model from scratch as it is too time consuming. Instead, a pre-trained model based on the concept of transfer learning will be employed. Due to the unavailability of funding, the scope of this project in a few aspects will be narrowed down as summarised in the Table 1.1.

Table 1.1: Scope of this project

Scope	Description
Dataset	Publicly available datasets
Programming Language, frameworks, libraries and IDE	Python and other open sourced frameworks, libraries and IDE
Training type	Local Machine (offline)
GPU	GPU that was shipped with the machine at the time of purchase: NVIDIA Geforce 930MX
CPU	CPU that was shipped with the machine at the time of purchase: Intel Core i5- 7200U @ 2.50 GHz
RAM	Upgraded to 8.00 GB

Due to limitations in computing resources, the size of the datasets used to train the model in this project were limited to roughly 200 000 images. This may contribute to lower accuracies achieved by the model [3], [4]. Additionally, the iterations and epochs and batch size were reduced in order to fit to the researcher's hardware configurations. Artificial Intelligence and its subsets, ML and Deep Learning (DL) are inspired by theories in cognitive science and neuroscience. Thus, this project too, in the context of adding modifications to current models are closely related and limited by current theories in neuroscience and cognitive science and other related fields [4].

1.7 Project Contributions

Road scene applications require real-time or near real-time computation speeds in order to be practical for use in driving scenes [5]. This, obviously is the case as split second decisions can be the difference between routine passage and the occurrence of an accident. Through the literature review task, three main themes of the research papers referenced in this project report were uncovered as problem statements. Hence, the aim of this project was to contribute to the solution proposals for these problem statements. In summary, the problem statements were the underutilization of datasets, high computation time and low accuracy. The publicly available Microsoft COCO dataset was utilized in this project. The selection of this dataset allowed to solve for dataset underutilization and high computation time as transfer learning could be utilized. The experimentation phase involved fine-tuning the hyperparameters in the Mask R-CNN framework. Initially, the configurations were set to enable the researcher's hardware to cope with the demands of the framework. Thus, this initial set of configurations were labelled as *Set A* and was used as the base set up which was used to compare the performance metrics of succeeding sets. The hyperparameter configurations of sets which show an improvement in testing accuracy performance are adopted for the next sets in an attempt to build on the performance. The improvement in accuracy obtained among the eight sets used in the experimentation phase is a proof of concept that the hyperparameter fine-tuning can be utilized as a cost-effective way to solve for low accuracy problems.

1.8 Project Gantt Chart

In Table 1.2, the task of researching the topic and compiling the references for the project took one week as expected. The literature review activity took longer than expected due to reviews of the work and amendments. The understanding of the concepts required further reading, contributing to a longer time taken for this activity. On the other hand, the setting up of IDE and Python libraries took a much longer

duration than planned. This is because acquiring the libraries that matched the correct dependencies was more challenging than anticipated. In the midst of experimenting with the model in [6], a substitute model had to be found because further efforts to match the exact dependencies with the requirements of the author was not feasible due to time constraints. The activity of familiarising with ML frameworks and models took lesser time than anticipated as the previous activities provided better understanding and experience. The FYP 1 report writing activity was started and ended according to the schedule. The last two weeks leading to the FYP1 Presentation week will be spent on reviewing and preparing for the presentation, as scheduled.

On the other hand, as shown in Table 1.3, the second part of this project consists of three main experimentation activities which will begin with the task of acquiring and understanding the hidden layers of a pre-trained model. This is an important pre-requisite knowledge to perform modification and analysis with existing ML models. Eventually, the model will be trained and tested on a chosen dataset. If satisfying results are obtained using the output data, the new model would be compared with other state-of-the-art models that are available. On the other hand, the writing of the second part of the project would be carried out simultaneously beginning week two of FYP Part 2. Finally, the thesis will be reviewed and necessary modifications will be made and finalized one week before the submission date.

Table 1.2: FYP Part 1 Gantt Chart

Year		2019							2020						
Activity		Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14
Research of topic	Planned														
	Actual														
Literature Review	Planned														
	Actual														
Set up of IDE and libraries	Planned														
	Actual														
Familiarisation with Machine Learning framework and models	Planned														
	Actual														
FYP Part 1 report writing	Planned														
	Actual														
FYP Part 1 report review and presentation slides preparation	Planned														
	Actual														
FYP Part 1 Presentation	Planned														
	Actual														

Legend		Planned Duration
		Actual Duration

Table 1.3: FYP Part 2 Gantt Chart

Year		2020													
Activity		Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14
Acquiring amd understanding hidden layers of the pre-trained model	Planned														
	Actual														
Performing modification on acquired model and analyzing output data	Planned														
	Actual														
Comparative analysis of own model to state-of-the art models	Planned														
	Actual														
FYP Part 2 thesis writing	Planned														
	Actual														
FYP Part 2 thesis review and presentation slides preparation	Planned														
	Actual														
FYP Part 2 Submission	Planned														
	Actual														
FYP Part 2 Presentation	Planned														
	Actual														

Legend		Planned Duration
		Actual Duration

1.9 Project Flow Chart

In Figure 1.3, the research topic is the beginning activity of FYP Part1. It is important to identify the research gap in this field. Once the research gap has been identified, the problem statement can be identified. Literature review enables one to know in more depth of the frameworks, ML models and Python libraries that may be used in this project. Further familiarization with the ML models and frameworks can commence once the right libraries and packages are installed in Anaconda. FYP part 1 report writing will be conducted during the phase of setting up the work environment. The FYP part 1 report will be reviewed and amendments will be done accordingly. Once satisfied with the outcome, the presentation slides will be prepared with reference to the finished FYP part 1 report. Finally, FYP part 1 will end with a presentation.

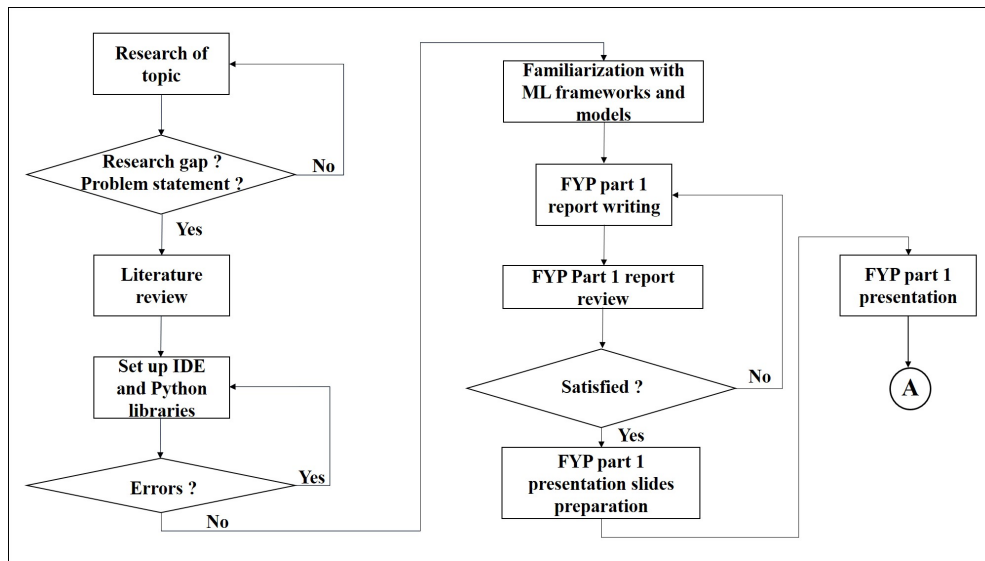


Figure 1.3: FYP Part 1 Flow Chart

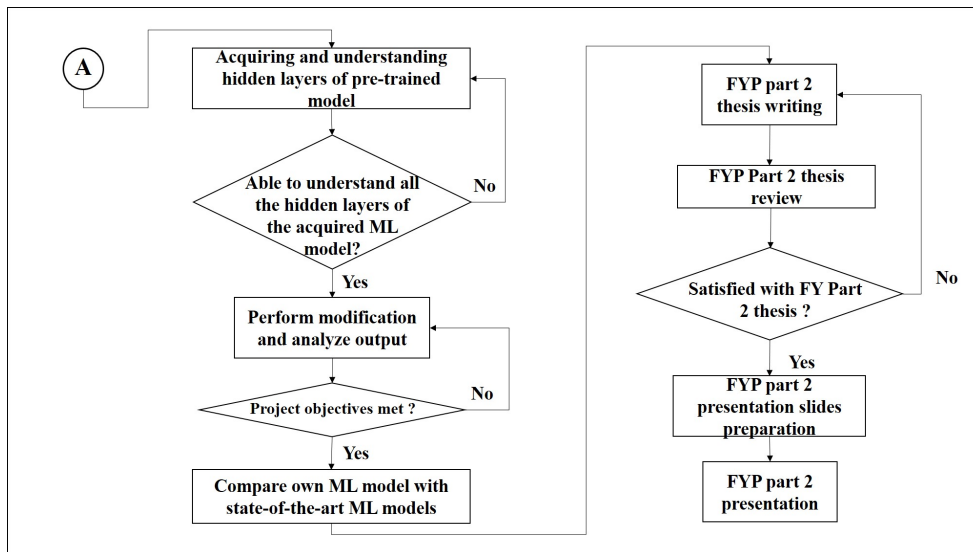


Figure 1.4: FYP Part 2 Flow Chart

1.10 Report Organization

In Project Background sub-section, an in-depth explanation about the basic concept of semantic segmentation is provided. Next, road scene features that makeup the classes are discussed in the Road Features sub-section. Then, a discussion on three image classification frameworks as per the scope of this project is provided. Next, in the Datasets sub-section, three popular datasets are introduced. Finally, in the Ground Truths sub-section, an explanation about ground truth is provided.

CHAPTER 2: LITERATURE REVIEW

2.1 Overview

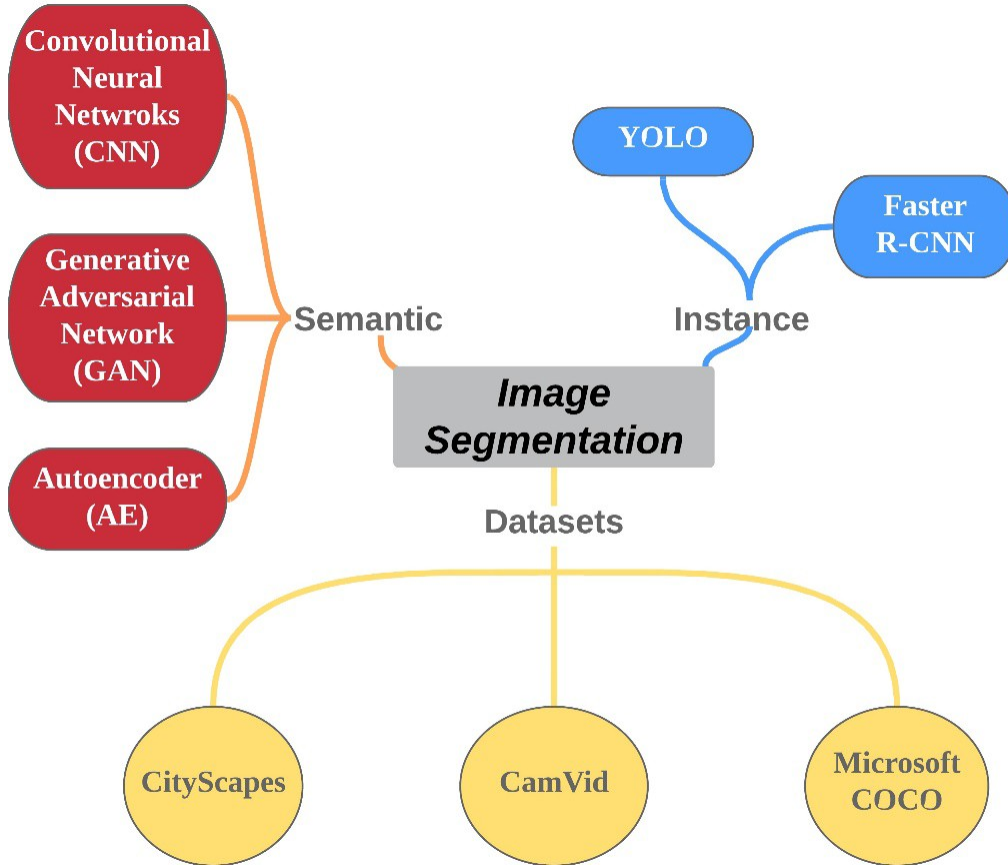


Figure 2.1: Mind map of image segmentation

As shown in Figure 2.1, image segmentation can be split into two main methods, namely, semantic and instance. This project focuses on semantic segmentation. Therefore, only aspects of this method will be discussed in depth. As shown in Figure 2.1, this section begins with a discussion on semantic segmentation. Hence, three types of Machine Learning (ML) frameworks of semantic segmentation are discussed. Various ML frameworks have been introduced in this field with the goal of increasing accuracy. The timeline of the debut and advancement of these frameworks have been covered extensively in [4] and [11]. Therefore, the history and background of these frameworks will not be discussed. Instead, the concepts and working principles of three of these frameworks, namely Convolutional Neural Networks (CNN) [5], [8], [13], [15], Generative Adversarial Networks (GAN) [10] and Autoencoders (AE) [13] will be reviewed. Then, the concept of instance segmentation is introduced briefly followed by a brief introduction to the You Only Look Once

(YOLO) and Faster R-CNN frameworks. Then, three different public datasets are discussed. Finally, this chapter is concluded in the summary section.

2.2 Semantic Segmentation

The concept of segmentation is to identify individual objects of interest that in an image. To illustrate this, Figure 2.2 shall be observed. Segmenting the image in Figure 2.2(a) would result in having six individual digits(objects) as shown in Figure 2.2(b). Similarly, the road scene image in Figure 2.3(a) results in the segmented image illustrated in Figure 2.3(b). In order to segment objects individually, each object must have its own characteristics that distinguishes it from the background. This set of characteristics are known as semantics and typically fall into one of two categories, namely specific geometry and colour [1], [3], [12], [13], [4]–[11].

Upon segmenting the objects in the image, each object can be classified. In each situation, the classes of objects would be different. With reference to Figure 2.2, Figure 2.2(a-b) would require classes of digits while Figure 2.3(a-c) would require classes of road features such as listed in [8]. In the application of the concepts mentioned above, the image shown in Figure 2.3(a) would be output as the image shown in Figure 2.3(c). Figure 2.4 illustrates three different types of specific geometries that are used as semantics in segmenting digits. As such, whenever a pattern of these geometries appear in a given vicinity, that collection of pixels would be segmented from the background [14].

504192

(a) Image of handwritten digits

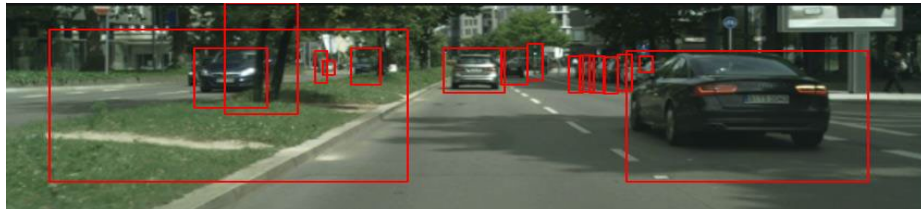


(b) Segmented handwritten digits

Figure 2.2: Image of handwritten digits before and after segmentation [14]



(a) Road scene image



(b) Semantically segmented road scene image



(c) Semantically segmented and classified road scene image

Figure 2.3: Road scene image passing through semantic segmentation and classification processes

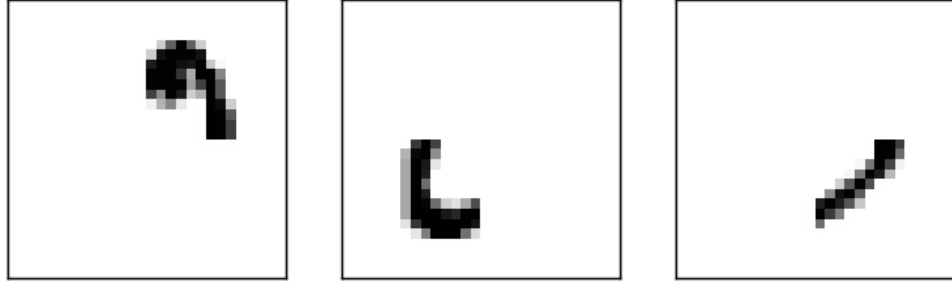


Figure 2.4: Specific geometries used for segmenting digits [14]

2.2.1 Convolutional Neural Network

The underlying concept of a neural network (NN) [1], [3], [12]–[16], [4]–[11] is neurons that are designed based on the concept of sigmoid neurons [14], [16]. The coloured circles in Figure 2.5 are neurons and those neurons that serve as inputs are collectively known as the input layer while those that provide the output, are collectively known as the output layer. The remaining neurons are collectively known as the hidden layers of the network. In a NN, the hidden layers that are closer to the input would detect primitive patterns in an image such as edges and boundaries while the hidden layers that are closer to the output layer pick up much more sophisticated patterns such as faces and entire shapes. An analogy for hidden layers would be to think of departments in a company. Take for example, a finance department. This department would generally handle processes that involve a company's finances. Within that department, there would be smaller units to handle sub-processes such claims, budgeting, payments, rent, etc. Each hidden layer is analogous to a department while each neuron within that layer is analogous to the smaller units within the department.

The type of neurons typically used are known as sigmoid neurons. These neurons take in one or more inputs and produce an output. In equation 2.1, x is the inputs, x_1, x_2, x_3, \dots , w is the weights, w_1, w_2, w_3, \dots , and b is the bias. Weights are merely the degree of importance one would want to give to a particular data. Each hidden layer has such sigmoid neurons with each having its own weight and bias. The next layer takes the output from the preceding as its input. Biases are threshold values

$$\sigma(z) = \frac{1}{1 + \exp(-\sum_j w_j x_j - b)} \quad (2.1)$$

Typically, images in a dataset have a resolution of 1024 x 2048 [8]. That is more than two million input parameters. Thus, it would be good practice if these images can be shrunk in size but still preserve the important information. This inspired the advent of Convolutions in CNNs. Convolutions are merely matrices whose values represent weights. These matrices are also known as kernels. These kernels are multiplied to the input pixels across the input image to obtain a smaller image that still preserves the information. The idea behind this is the assumption that if a particular pixel is part of a semantic feature, it is very likely that its neighbouring pixels are also part of the edge. Upon running the kernel across and vertically for the entire image, a convoluted image is produced. The convoluted image would function as a feature map for a particular feature. Figure 2.6 illustrates this concept in application. It is worth noting in Figure 2.6(b), the image is merely a feature map of edges and is still of the same size as the original image. This happens when padding is introduced into original image. When a kernel is applied to an image, the pixels at the corners of the image do not get emphasized as much as those at the centre of a particular group of pixels. Thus, a solution is to ensure that even corner pixels are the centre of a group of pixels by adding zeros to the edges of an image. This is known as zero-padding. Pooling is another way in which images are shrunk while also preserving important data. The most common way this is done is by taking a group of pixels and merely selecting the highest value among that group of pixels. The input image is taken, convoluted and pooled. Then, the pixel values of the convoluted and pooled images are fed into the higher layers of the network. Obviously, now there would be a much lesser number of neurons required as there are lesser pixels and thus, fewer input parameters. More importantly, the convolutions would have enabled the NN to learn the features that exist in the input images as the weights are tweaked to produce predictions with lesser and lesser errors.

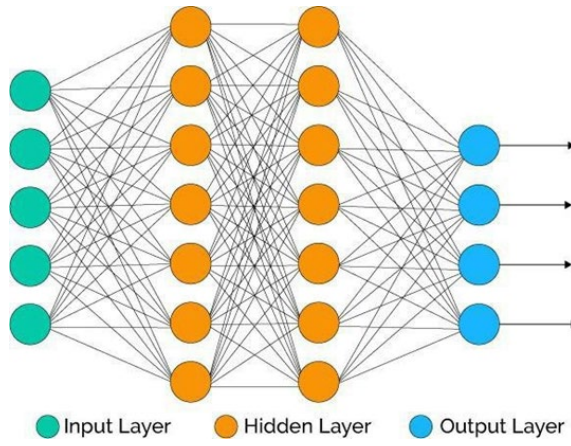
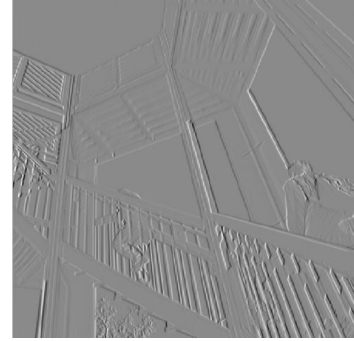


Figure 2.5: Three types of layers found in a Neural Network [14]



(a) Stairway image before convolution



(b) Stairway image after convolution

Figure 2.6: A vertical edge kernel applied to a stairway image [1]

2.2.2 Generative Adversarial Network

Generative Adversarial Networks (GANs) can be broken down into two neural networks. One is a Generator Network (GN) and the other is a Discriminator Network (DN). Figure 2.7 illustrates the main components of a GAN. As seen in the figure, random noise (random images) is fed into the GN. Using this random noise, the GN, which in itself is a convolutional network, creates a fake image (denoted in figure by $X = G(z)$). Then, the fake images and real images are fed into the DN. The DN looks at the real images and fake images separately and decides if the input image is real or fake. A GN on its own will merely produce random noise. The DN sees the features

that contribute to a real image. Thus, the discriminator virtually guides the generator to produce images that will be much closer to the real images.

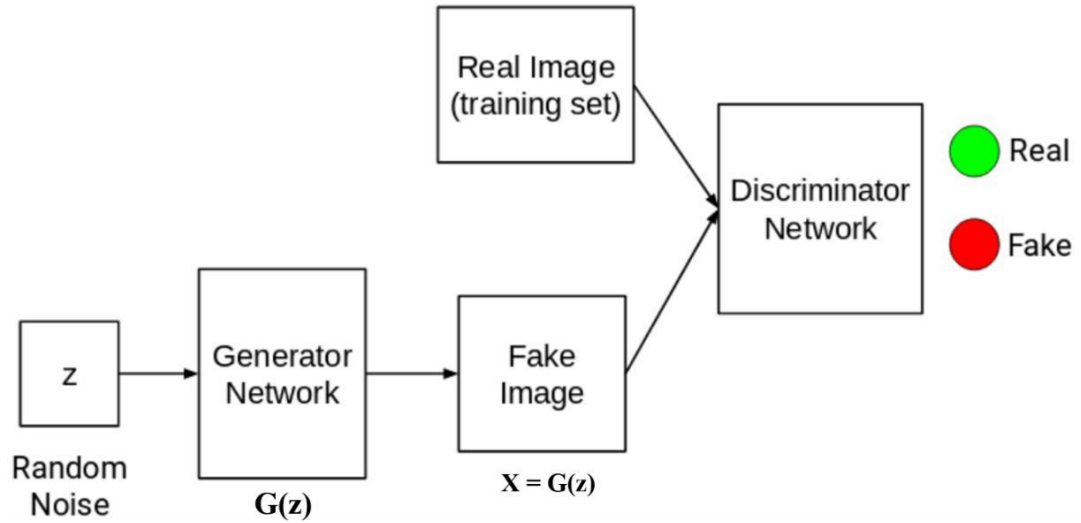


Figure 2.7: Main components of a Generative Adversarial Network

2.2.3 Auto Encoder

In Figure 2.8, the illustrated AE network takes inputs, denoted as X in the figure and produces a feature map, denoted as Z in the figure. Then, the decoder network reconstructs images based on the feature map, denoted as \hat{X} in the figure. Then, the reconstructed image is compared with the labelled image and the model is trained by minimizing the reconstruction error using the L2 Loss Function [1], [4], [10].

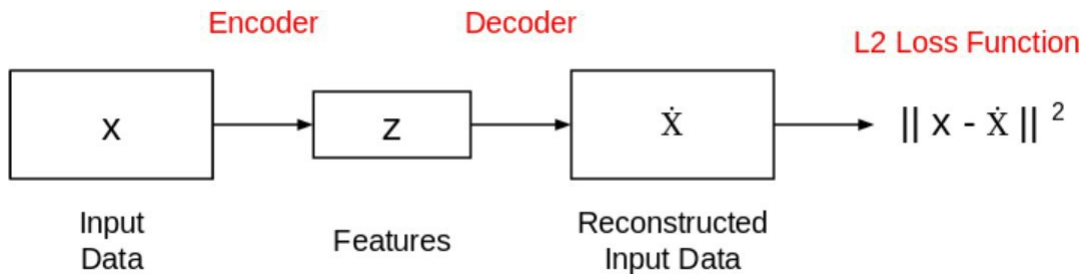


Figure 2.8: Auto Encoder network

2.3 Instance Segmentation

In instance segmentation, the objects in an image are localized and identified as separate individual objects. As illustrated in Figure 2.9, this is done by identifying the boundaries of objects in an image and individual objects are differentiated with different colours even if they belong to the same class. In this section, two frameworks that are often used in instance segmentation are discussed, namely You Only Look Once (YOLO) and the other, Faster Region-based Convolutional Neural Network (Faster R-CNN).

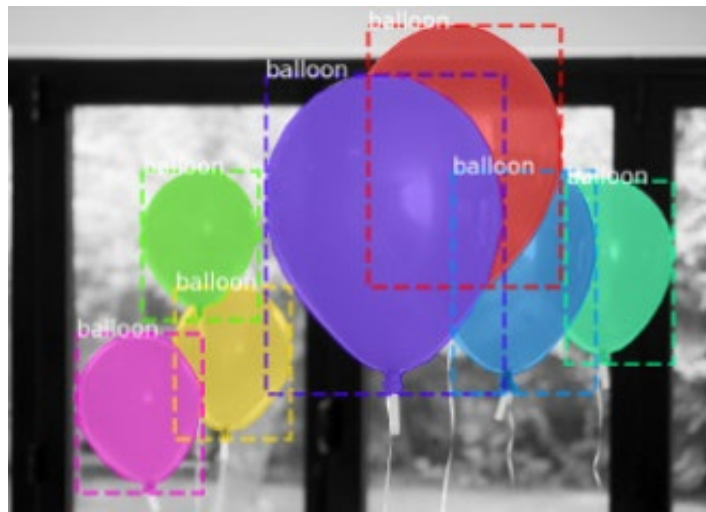


Figure 2.9: Instance segmentation demonstrated on an image of balloons [1]

2.3.1 You Only Look Once

You Only Look Once (YOLO) [2] essentially, is a framework that is inspired by the R-CNN framework. The framework takes in an input image and breaks it down into grids. As illustrated in Figure 2.10, the input image which has been broken down into grids are fed into a CNN layer to perform localization and classification on each grid. Hence, bounding boxes along with the corresponding probabilities of classes for the objects in the grids are predicted. In Figure 2.10, the max pool layer is similar to convolutions performed in CNNs. However, pooling lightens the load on the network

by reducing its spatial size. These steps are repeated until a final output image with bounding box predictions are obtained

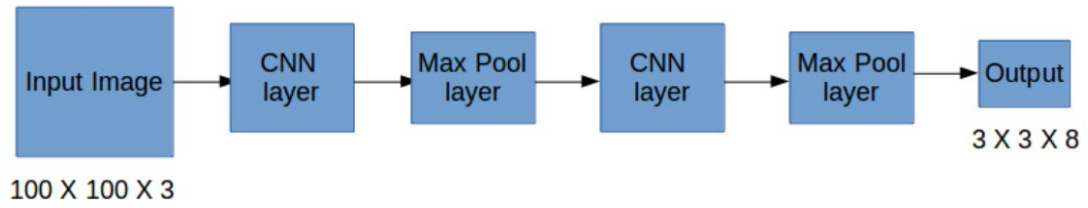


Figure 2.10: You Only Look Once framework [2]

2.3.2 Faster R-CNN

The Faster R-CNN [3], [4] framework is a predecessor of the Mas R-CNN framework that is used in this project. As shown in Figure 2.11, the convolutional layers of the CNN take in an image as the input where the features are extracted to form feature maps. These feature maps are in turn passed into the Region Proposal Network (RPN) [3] and ROI pooling layer as shown in Figure 2.11. The RPN produces proposals termed as Regions of Interest (ROI). The ROI pooling layer selects ROIs that has the highest probability of containing an object. Finally, the classifier identifies the objects in these ROIs and finalises the bounding boxes for these object classes in the final output image.

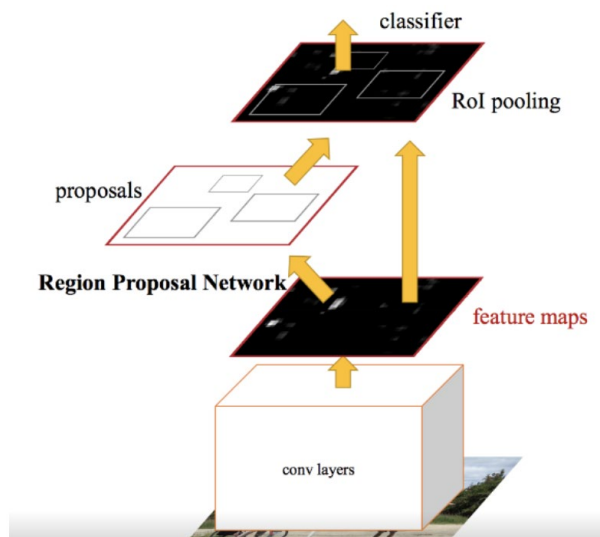


Figure 2.11: Faster Region-based Convolutional Neural Network framework [2], [3], [4]

2.4 Datasets

In this section, two commonly used road scene datasets as shown in Figure 2.12, namely Cityscapes [8], [9] and CamVid [1], [5], [13], will be reviewed. The Microsoft COCO dataset will also be discussed as it is the dataset that was utilized in this project

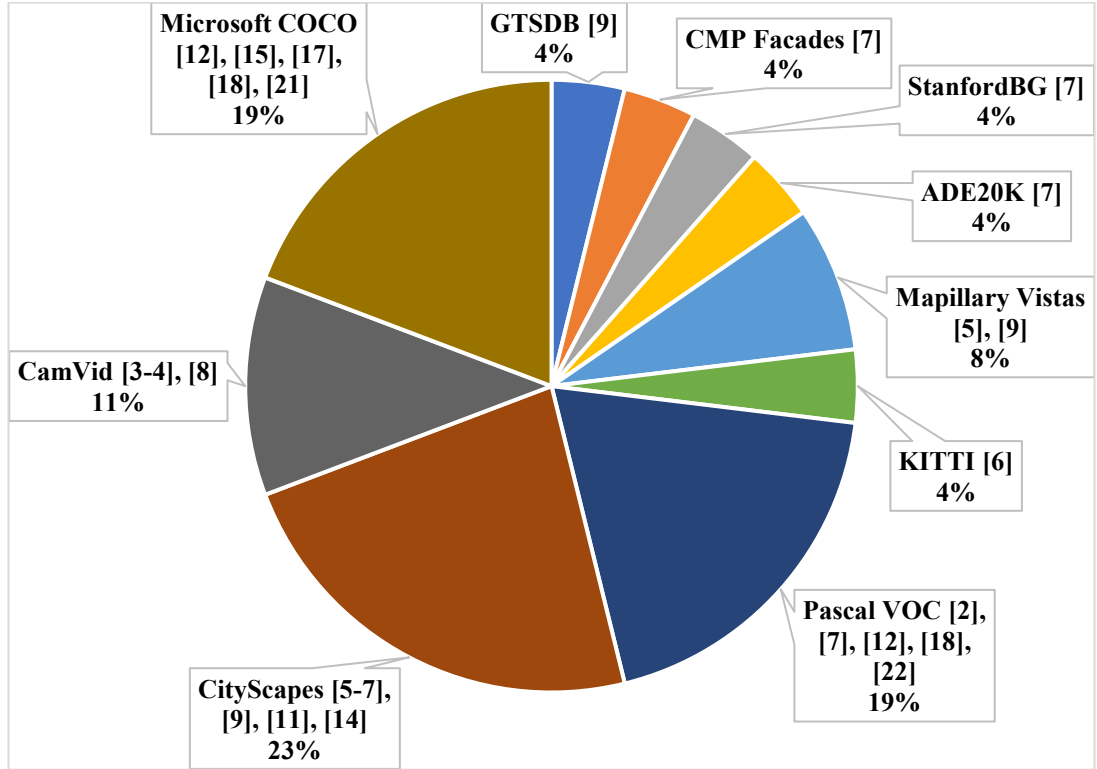


Figure 2.12: Distribution of datasets used in [2-12], [14-15], [17-22], [26]

2.4.1 CityScapes Dataset

As shown in Figure 2.13(a), Cityscapes is a dataset that comprises urban road footage. All 30 classes in this dataset comprise dense pixel annotations of instances. It has about 5000 fine annotated images for data captured in 50 cities.

2.4.2 CamVid Dataset

CamVid is another road footage dataset, as shown in Figure 2.13(b). It is captured with a 960×720 resolution dashboard camera. The images in this dataset consist of 32 classes that were hand-annotated. This dataset is also broken into 367 images for training, 100 images for validation, and 233 images for testing. Additionally, in this dataset, less important classes such as skies, roads and buildings make up large areas of individual images in the dataset while more important classes such as vehicles, pedestrians and signal signs account for lesser pixels.

2.4.3 Microsoft COCO Dataset

This dataset contains 81 classes including a background class. There are 118000 images for training, 5000 images for validation and 41000 images for testing in this dataset. Figure 2.13(c) shows a few images taken from this dataset. However, the images in this dataset are general images. For this project, the dataset was obtained from [1].



(a) Images found in Cityscapes dataset [5-7], [9], [11], [14]



(b) Images found in CamVid dataset [3-4], [8]



(c) Images found in Microsoft COCO dataset [12], [15], [17], [18], [21]

Figure 2.13: Images from three datasets in [2-12], [14-15], [17-22], [26]

2.5 Summary

In summary, the first objective of this project has been fulfilled with the selection of the Microsoft COCO dataset for this project. This dataset was chosen because it is the dataset used in the Mask R-CNN framework that is used for this project. Choosing a popular dataset would enable fairer comparisons to be made with the other models in this field as the different models would essentially be trained and

tested on similar input data. However, this would require further adaptations in the implementation of the Python code used to build this framework. Furthermore, using a fair benchmarking method further justifies the usage of the Microsoft COCO dataset. The research papers that implemented ML frameworks are summarized in Table 2.1. As shown in Table 2.1, an underlying framework based upon CNNs are implemented in [1], [2], [3], [4], [5], [6], [7], [8]. This can be attributed to the effectiveness of CNNs in the application of image segmentation [9]. The convolutional nature of CNNs make it ideal to be used as backbone frameworks that extract low-level features in the early stages of the overall framework and feed this information in the form of feature maps to the next layers in the framework. On the other hand, as shown in Table 2.1, [2], [3], [4], [5], [6], [7], have utilized Nvidia GPUs in conducting the experiments. This can be attributed to the availability of documentation and libraries that enable reductions in training time required by GPUs by utilizing the parallel computing capabilities of GPUs [1].

Table 2.1: Summary of Significance of Findings in Papers Reviewed

No.	Title	Authors	Publication Year	Type Of Paper	Significance of Findings	
1	Methods and datasets on semantic segmentation: A review[5]	Yu, Hongshan	2018	Journal	Accuracy:	86.9% mIoU (highest accuracy method chosen. Other techniques available)
		Yang, Zhengeng			Technique(s) used:	DeepLabv3
		Tan, Lei			Dataset(s):	PASCAL VOC 2012
		Wang, Yaonan			Limitations:	Computation time
		Sun, Wei			Hardware:	NVIDIA Tesla K40
		Sun, Mingui				
		Tang, Yandong				
2	Fast road scene segmentation using deep learning and scene-based models[6]	John Vijay	2016	Conference	Accuracy:	100%
		Kidono, Kiyosumi			Technique(s) used:	Deconvnet
		Guo, Chunzhao			Dataset(s):	Custom dataset (acquired. Source not mentioned)
		Tehrani, Hossein			Limitations:	limited object classes in dataset, computation speed too low to perform real-time computation
		Mita, Seiichi			Hardware:	NVIDIA Geforce GTX960
		Ishimaru, Kasuhiza				
3	Three-Skips CNN for road scene semantic segmentation [7]		2017	Conference	Accuracy:	mean IoU of 53.1%
		Tang, Jing			Technique(s) used:	VGG
		Wang, Xin			Dataset(s):	CamVid
					Limitations:	Computation time,memory in road scene applications
					Hardware:	NVIDIA Titan X

No.	Title	Authors	Publication Year	Type Of Paper	Significance of Findings	
4	Real-time Semantic Segmentation for Road Scene[8]	Zhang, Xuetao	2019	Conference	Accuracy:	mean IoU of 55.6 %
		Chen, Zhenxue			Technique(s) used:	ResNet
		Lu, Dan			Dataset(s):	CamVid
		Li, Xianming			Limitations:	Computation time
					Hardware:	NVIDIA Titan XP
5	Multistage shallow pyramid parsing for road scene understanding based on semantic segmentation [9]		2019	Conference	Accuracy:	78.61 % IoU
					Technique(s) used:	PSPNet
		Nurhadiyatna, Adi			Dataset(s):	Cityscapes
		Loncaric, Sven			Limitations:	Computation time
					Hardware:	NVIDIA Titan X
6	Road segmentation for all-day outdoor robot navigation[1]	Zhang, Yuxiao	2018	Journal	Accuracy:	93.7% IoU, 93.5 % IoU
		Chen, Haiqiang			Technique(s) used:	Supervised Generative Segmentation Network (SGSN)
		He, Yiran			Dataset(s):	Cityscapes,KITTI
		Ye, Mao			Limitations:	inaccuracy of existing datasets for all-day outdoor robots
		Cai, Xi			Hardware:	NVIDIA GTX 1080 Ti
		Zhang, Dan				
7	A novel framework for semantic segmentation with generative	Zhu, Xiaobin	2019	Journal	Accuracy:	79.1 %, 83.4 %, 36.9 %, 73.9 %, 53.5%
		Zhang, Xinming			Technique(s) used:	Atrous Spatial Pyramid Pooling (ASSP)
		Zhang, Xiao Yu			Dataset(s):	Cityscapes, Pascal VOC 2012,ADE20K,StandfordBG,CMP Facades
		Xue, Ziyu			Limitations:	insufficient GPU memory

No.	Title	Authors	Publication Year	Type Of Paper	Significance of Findings	
	adversarial network[10]	Wang, Lei			Hardware:	Not stated
8	A road segmentation method based on the deep auto-encoder with supervised learning[11]	Song, Xiaona	2018	Journal	Accuracy:	95.40%
		Rui, Ting			Technique(s) used:	Autoencoder (AE)
		Zhang, Sai			Dataset(s):	CamVid
		Fei, Jianchao			Limitations:	Computation time
		Wang, Xinqing			Hardware:	Not stated
9	Training of Convolutional Networks on Multiple Heterogeneous Datasets for Street Scene Semantic Segmentation [12]		2019	Conference	Accuracy:	mean Pixel Accuracy (mPA) of 49.0%
		Meletis, Panagiotis			Technique(s) used:	CNN
		Dubbelman, Gijs			Dataset(s):	Cityscapes, Mapillary Vistas, GTSDb
					Limitations:	limited GPU memory
					Hardware:	NVIDIA Titan X
10	Image understanding: Semantic Segmentation of Graphics and Text using Faster-RCNN[13]	Latha, H N	2018	Conference	Accuracy:	
		Rudresh, Sadhan			Technique(s) used:	Faster R-CNN
		Sampreeth, D			Dataset(s):	pre-trained Alexnet, own data collected
		Otageri,			Limitations:	stock Optical Character Recognition (OCR) are not trained for any specific font or language. Hence, this leads to false recognition of some images
		Sangamesh M			Hardware:	Not stated
		Hedge, Saurabh S				

No.	Title	Authors	Publication Year	Type Of Paper	Significance of Findings	
11	Fully Convolutional Adaptation Networks for Semantic Segmentation [14]	Zhang, Yiheng	2018	Conference	Accuracy:	mIoU of 47.5%
		Qiu, Zhaofan			Technique(s) used:	Fully Convolutional Adaptation Networks (FCAN)
		Yao, Ting			Dataset(s):	custom dataset (synthetic rendered images, CityScapes, BSDS)
		Liu, Dong			Limitations:	dataset underutilization (generalization error on real images due to domain shift)
		Mei, Tao			Hardware:	Not stated
12	Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks [3]	Ren, Shaoqing	2017	Conference	Accuracy:	mAP of 59.9%
		He, Kaiming			Technique(s) used:	Region Proposal Network (RPN), Faster R-CNN
		Girshick, Ross			Dataset(s):	PASCAL VOC 2007, PASCAL VOC 2012 , MS COCO
		Sun, Jian			Limitations:	Not stated
					Hardware:	Not stated
14	Attribute-aware Semantic Segmentation of Road Scenes for Understanding Pedestrian Orientations [15]	Sulistiyo, M. D.	2018	Journal		
		Kawanishi, Y.			Accuracy:	50.3 % mean IoU
		Deguchi, D.			Technique(s) used:	SegNet
		Hirayama, T.			Dataset(s):	CityScapes, modified CityScapes
		Ide, I.			Limitations:	Computation time
		Zheng, J. Y.			Hardware:	NVIDIA Titan X
		Mutase, H.				
15	Deep residual learning for	He, Kaiming	2016	Journal		
		Zhang, Xiangyu			Accuracy:	mAP @ [0.5:0.95] : 37.4

No.	Title	Authors	Publication Year	Type Of Paper	Significance of Findings	
	image recognition [16]	Ren, Shaoqing			Technique(s) used:	Faster R-CNN with ResNet-101
		Sun, Jian			Dataset(s):	Microsoft COCO
					Limitations:	Computation time
					Hardware:	-
16	A Review on Deep Learning Techniques Applied to Semantic Segmentation [17]		2017	Journal		
		Garcia-Garcia, Alberto			Accuracy:	IoU of 91:60 % (method with highest accuracy chosen. Other methods are available)
		Orts-Escolano, Sergio			Technique(s) used:	Directed Acyclic Graph RNNs (DAG-RNNs)
		Oprea, Sergiu			Dataset(s):	CamVid
		Villena-Martinez, Victor			Limitations:	Computation time
		Garcia-Rodriguez, Jose			Hardware:	Not stated
17	Weighted feature pyramid networks for object detection [4]	Li, Xiaohan	2019	Conference		
		Lai, Taotao			Accuracy:	AP@[0.5]: 58.5%
		Wang, Shuaiyu			Technique(s) used:	Faster R-CNN with FPN and ResNet-101
		Chen, Quan			Dataset(s):	Microsoft COCO
		Yang, Changcai			Limitations:	Computation time
		Chen, Riqing			Hardware:	NVIDIA M40
18	Fast R-CNN [18]		2015	Conference		
					Accuracy:	mAP of 70.0%

No.	Title	Authors	Publication Year	Type Of Paper	Significance of Findings	
					Technique(s) used:	Fast R-CNN
		Girshick, Ross			Dataset(s):	Pascal VOC 2007
					Limitations:	Computation time
					Hardware:	Nvidia K40
19	PixelNet: Representation of the pixels, by the pixels, and for the pixels [19]		-	-		
		Aayush Bansal			Accuracy:	IoU of 67.4%
		Xinlei Chen			Technique(s) used:	Fully Convolutional Networks (FCN)
		Bryan Russell			Dataset(s):	NYUDv2, BSDS, PASCAL-Context
		Abhinav Gupta			Limitations:	Computation time
		Deva Ramanan			Hardware:	NVIDIA TITAN X
20	Cross-Domain Traffic Scene Understanding : A Dense Correspondence-Based Transfer Learning Approach [20]	Ling, Haibin	2018	Journal		
		Di, Shuai			Accuracy:	highest per-class Cumulative Matching Score of 80.8%
		Zhang, Honggang			Technique(s) used:	CNN, Correspondence-based transfer learning (DCTL)
		Li, Chun Guang			Dataset(s):	custom dataset
		Mei, Xue			Limitations:	Underutilized dataset (datasets do not provide varying weather condition images for similar locations)
		Prokhorov, Danil			Hardware:	NVIDIA GeForce GTX TITAN X GPU
21	Mask R-CNN [21]		2020	Journal		
		He, Kaiming			Accuracy:	AP of 37.1%(CityScapes dataset accuracy also available)

No.	Title	Authors	Publication Year	Type Of Paper	Significance of Findings	
		Gkioxari, Georgia			Technique(s) used:	Mask R-CNN
		Dollár, Piotr			Dataset(s):	Microsoft COCO, CityScapes
		Girshick, Ross			Limitations:	Computation time, Dataset underutilization
					Hardware:	Nvidia Tesla P100
22	Region-Based Convolutional Networks for Accurate Object Detection and Segmentation [22]		2016	Journal		
		Girshick, Ross			Accuracy:	mAP of 66.0 %
		Donahue, Jeff			Technique(s) used:	R-CNN with OxfordNet (O-Net)
		Darrell, Trevor			Dataset(s):	Pascal VOC 2007
		Malik, Jitendra			Limitations:	Underutilization of dataset (limited data)
					Hardware:	NVIDIA Titan Black
23	A two-stage real-time YOLOv2-based road marking detector with lightweight spatial transformation-invariant classification [2]		2020	Journal		
		Ye, Xing Yu			Accuracy:	86.30%
		Hong, Dza Shiang			Technique(s) used:	YOLOv2
		Chen, Hung Hao			Dataset(s):	Custom dataset
		Hsiao, Pei Yung			Limitations:	Not stated
		Fu, Li Chen			Hardware:	NVIDIA GeForce GTX 1070
24	DAG-Recurrent Neural Networks For		2016	Conference		
		Bing Shuai			Accuracy:	IoU of 91:60 %
		Zhen Zuo			Technique(s) used:	Directed Acyclic Graph RNNs (DAG-RNNs)

No.	Title	Authors	Publication Year	Type Of Paper	Significance of Findings	
	Scene Labeling [23]	Gang Wang			Dataset(s):	CamVid
		Bing Wang			Limitations:	Underutilization of dataset (equipping local features with a broader view of contextual awareness)
					Hardware:	Not stated

CHAPTER 3: METHODOLOGY

3.1 Overview

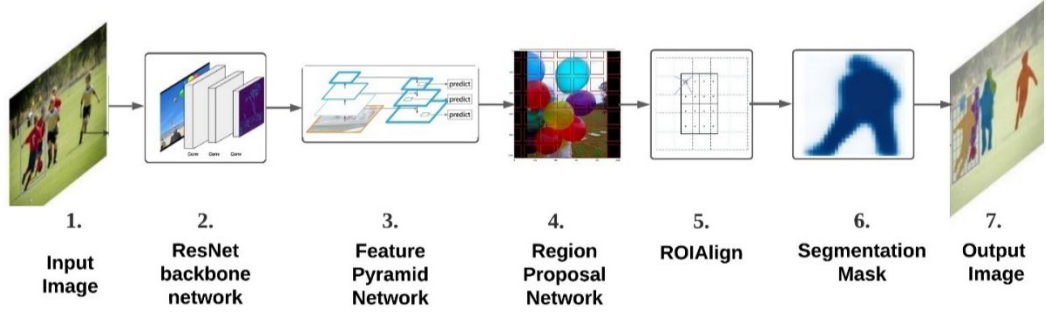


Figure 3.1: Overall Mask R-CNN framework [2], [3]

This section begins by introducing the overall framework of Mask R-CNN illustrated in Figure 3.1. Hence, selected components of the framework are discussed further in the sub-sections that follow. ResNet-50 backbone framework. The next stage is the Feature Pyramid Network (FPN). The stage that follows the FPN is the Region Proposal Network (RPN). This is followed by the ROIAlign stage which leads to the final stage, the Segmentation Mask stage. The output image is produced after this stage. As shown in Figure 3.1, the flow of the framework starts with the input image that are 800×1024 pixels in size.

3.1.1 ResNet Backbone Network

Typically known as a backbone network, the ResNet-50 is a CNN that has the role of extracting low level features such as edges, corners and lines from images in the early layers of the network as illustrated in Figure 3.2. The image undergoes convolutions as it passes through the backbone network, being converted from 1024×1024 pixels as the input image to 32×32 pixels feature map that is then passed on to the later stages of the network. In this Mask R-CNN framework however, the backbone network is extended with a Feature Pyramid Network (FPN) [4] as an improvement to the ResNet backbone network used in Faster R-CNN [2].

The feature maps are passed into the Region Proposal Network (RPN). The RPN makes predictions to pick the anchors with the highest foreground scores and pass these on to the next stage as ROI proposals. ROIAlign, as opposed to ROI Pooling is performed on the outputs of the RPN.

In Segmentation Mask stage, the positive regions outputted by the ROI classifier are used to generate masks for each object. ROIs are categorised as positive only when they have an IoU of at least 0.5 and negative otherwise

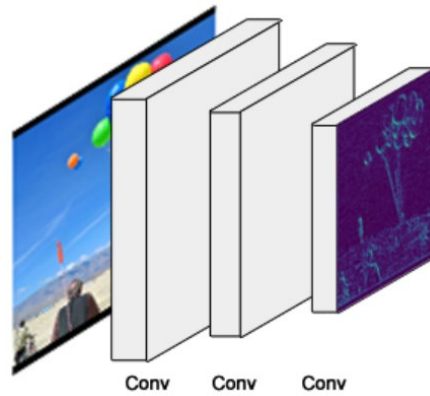


Figure 3.2: ResNet-50 extracting low-level features from input images [1]

3.1.2 Feature Pyramid Network

This FPN [4] component is an improvement made to this framework from the predecessor Faster R-CNN framework [21]. The FPN plays a role in enhancing the single feature extraction pyramid of the ResNet-50 backbone. As illustrated in Figure 3.3, the FPN consists of not one but two feature extraction pyramids. In Figure 3.3, the second feature extraction pyramid on the right takes extracted features from the first feature extraction pyramid on the left and passes those features to the lower layers of the network. This enables ROI feature extraction according to the scale of the image and predictions to be made at every level

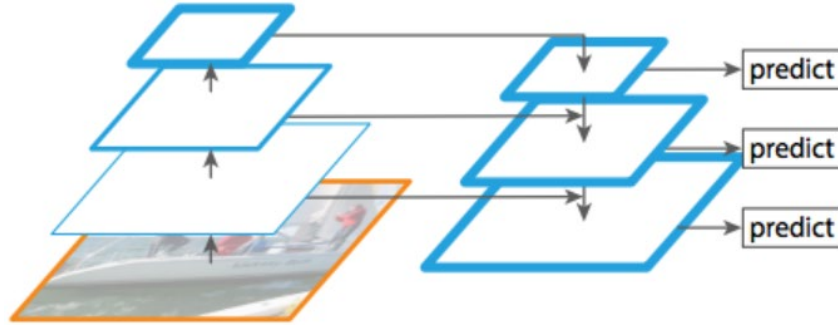


Figure 3.3: An illustration of Feature Pyramid Network as an additional pyramid that passes high level features [2]

3.1.3 Region Proposal Network

This component is an existing component of the predecessor Faster R-CNN framework. However, it is worth discussing the role of this component as it sheds light on the limitation of the Faster R-CNN framework to be used for mask segmentation [2], [21].

The RPN is a neural network that takes the feature maps produced by the backbone network and locates regions that contain objects as visualized in Figure 3.4. These regions are typically known as anchors. In the same figure, for simplicity just 49 anchors were used on an input image. However, in application, a feature map may contain a few thousand overlapping anchors that vary in size and aspect ratios in order to cover the feature map maximally [2]. Two outputs are generated by the RPN for each anchor. These are the anchor class and bounding box refinement. The anchor class signifies if there is an object in the anchor by classifying it with a Foreground or Background class while the bounding box refinement makes adjustments to fit the anchor boxes to fit the object more accurately. The limitation of this component to be used in segmentation arises as the anchors and bounding boxes generated in this component vary in size. Thus, the ROI Pool component is used to overcome this problem in the Faster R-CNN framework.

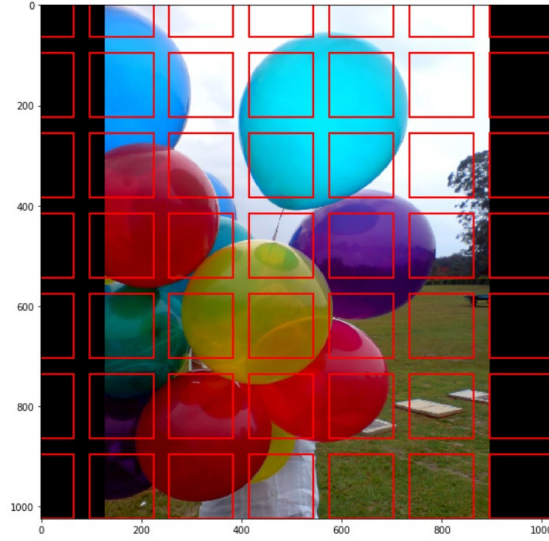


Figure 3.4: Simplified illustration to visualize anchors produced by the Region Proposal Network [2]

3.1.4 ROIAlign

The ROIAlign [21] component is a mere improvement that replaces the ROI Pool used in the Faster R-CNN framework [18]. ROIAlign is a standard process that extracts small feature maps from each ROI proposal from the RPN. The older ROI Pool quantizes a floating-number ROI. This quantization induces misalignments between the extracted features and the ROI proposals. Thus, here, the ROIAlign layer is used to replace the quantization of ROI Pool with a more robust technique to properly align the extracted features with the ROI proposals. This is done by breaking the feature map into grids as shown by the dashed lines in Figure 3.5. Then, a ROI box as shown by the solid lines in Figure 3.5 is generated. Additionally, as seen in Figure 3.5, four sampling points are computed using bilinear interpolation of nearby grid corners of the feature map. This creates standard-sized and better aligned feature maps extracted from the ROI proposals of the RPN [21].

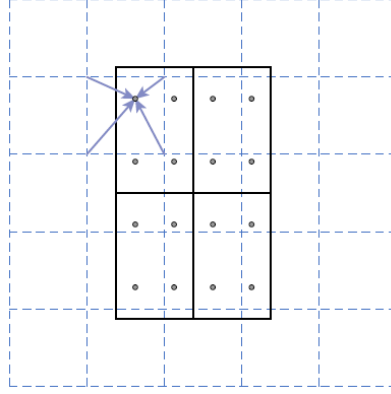


Figure 3.5: Visualization of bilinear interpolation performed by the ROIAlign component [2]

3.1.5 Segmentation Masks

This is a component that the Faster R-CNN lacks and thus is regarded as an extension to the faster R-CNN framework. In this component, the ROI classifier takes in the output of ROIAlign as its input and produces two outputs. Similar to the RPN, these two outputs are the anchor class and bounding box refinement. A distinction here is that unlike in the RPN, this module is capable of classifying the anchors to specific classes such as car, truck, van, people, ..., etc. For each class, the IoU is computed. ROIs that contain classes with IoU values of 0.5 and above are categorized as positive IoU ROIs and negative otherwise. Masks are only generated for positive ROIs. The masks generated are low resolution masks that are 28×28 pixels as shown in Figure 3.6. These masks are represented using float numbers in order to hold more details than binary masks. However, these are soft masks. During inferencing, the masks are scaled to the size of bounding boxes and are then represented as the final binary masks [2]. Each mask encodes the spatial layout of objects in input images [21]. Convolutions are naturally suited for pixel-to-pixel correspondence [19]. Thus, the extraction of spatial structure of the masks can be dealt with using convolutions. This pixel-to-pixel characteristic requires the feature maps of ROIs to be well aligned to accurately preserve the explicit per-pixel spatial connections. Thus, the ROIAlign layer plays an important role in mask prediction. On the other hand, the bounding box refinement in

this component makes further adjustments to the bounding boxes' location and size in order to fit the objects more accurately, much like that done in the RPN.

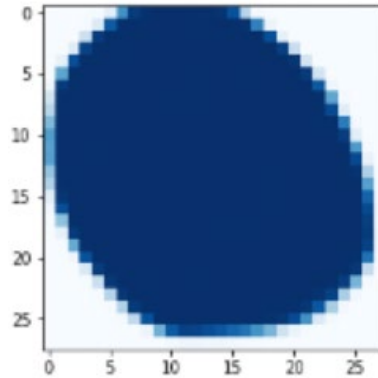


Figure 3.6: 28×28 pixels mask generated by the Segmentation Mask component [2]

3.2 Implementation

The implementation process flow for this project is shown in Figure 3.7. The process starts by implementing a Mask R-CNN framework [3] with initial configurations as shown in Table 3.2. As shown in Figure 3.7, this initial framework is loaded with the training dataset to train selected layers and update the weights of the initial framework. Then, the framework is loaded with the validation dataset to measure performance. This will be utilized as the controlled variable in the experimentation.

Then, the proposed configurations to the hyperparameter are implemented into the framework and the process is restarted. Upon validation, if there is no improvement to the performance of the framework as compared to the control framework, the proposed configurations to the hyperparameter are tweaked in the framework tuning process. These processes are iterated until an improved framework is obtained. Hence, the improved framework is then loaded with the test dataset. If there is high variance between the losses of the test dataset and validation dataset, framework tuning would have to be carried out again and the overall process restarted.

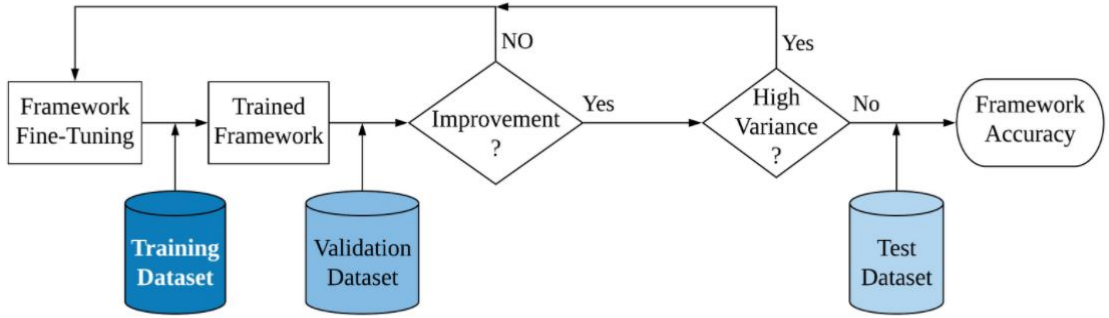


Figure 3.7: Implementation process flow

3.2.1 Training

The training itself consists of three stages. In stage one, the heads layers are trained with a learning rate of 0.002 and an epoch of 20. In stage two, the heads layers of the backbone ResNet-50 layers four and above are trained to fine tune the weights. This is similar to [1, 2]. Then, the heads layers of the ResNet-50 layers three and above are trained. Due to the utilization of transfer learning, the initial layers of each network within the framework do not need to be trained.

3.2.2 Validation

The Mask R-CNN framework uses mean Average Precision (mAP) [21], [1] to measure the accuracy of its predictions. This metric takes the AP of each class and averages them. The AP of each class in turn is computed by taking the area under a Precision-Recall curve. To calculate precision, the Intersection-over-Union (IoU) is computed. Typically, IoU is also referred to as True Positive. Equations 3.1, 3.2 and 3.3 are used in the computation of IoU, Precision and Recall respectively.

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}} \quad (3.1)$$

In Equation 3.1, the area of overlap is taken as the intersection of each class in each input image with its annotation. The range of this value is between 0 and 1 where 0 indicates a false prediction and 1 indicates an exact match.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (3.2)$$

$$\begin{aligned} &\text{Where} \\ &TP = IoU \geq 0.7 \\ &FP = \text{False Positive} \end{aligned}$$

In Equation 3.2, if a truck is predicted as a car, for example, this would be categorised as a False Positive. Here, the assumption made is that the annotation of a particular object in the ROI is a car. Thus, the car is a positive class while the truck is a negative class. Hence, the prediction that a truck is a car is a false prediction of a positive class.

$$\text{Recall} = \frac{TP}{TP + FN} \quad (3.3)$$

$$\begin{aligned} &\text{Where} \\ &TP = IoU \geq 0.7 \\ &FN = \text{False Negative} \end{aligned}$$

In Equation 3.3, if a car is predicted as a truck, for example, this would be categorised as a False Negative. Here, the assumption made is that the annotation of a particular object in the ROI is a car. Thus, the car is a positive class while the truck is a negative class. Hence, the prediction that a car is a truck is a false prediction of a negative class.

3.3 Configurations

Different configurations play a vital role in GPU usage intensity. Thus, the researcher's machine specifications and capabilities were taken into account when setting the configurations whilst aiming to achieve the objectives of this project. The environment and Python libraries used to run and experiment on the Mask R-CNN framework is summarised in Table 3.1. As shown in Table 3.1, a Docker container that supported all the libraries needed were utilized for this project. The utilization of Docker containers is time-saving as the libraries' dependency configurations are taken care of.

On the other hand, this Mask R-CNN implementation itself has configuration settings. In total, there are 38 configurations in this implementation of this Mask R-CNN framework [3]. Typically, configurations are also referred to as Hyperparameters. These configurations were mostly adopted from [1, 2] while a few that are discussed in this section were altered to the values displayed in Table 3.2. A more complete list is provided in section 4.1

Table 3.1: Summary of the environment used to conduct this project

Name of Python libraries within Docker container	Version of Python libraries within Docker container
Ubuntu	Ubuntu 16.04 LTS
Python	Python 3.5.2
Tensorflow	Tensorflow 1.6.0
Keras	Keras 2.1.5
PyTorch	PyTorch 0.3.1
OpenCV	OpenCV 3.4.1
Numpy	Numpy
Scipy	Scipy
Scikit Learn	Scikit Learn
Scikit Image	Scikit Image
Integrated Development Environment (IDE)	Jupyter Notebook

Table 3.2: Initial Hyperparameter Configurations (Set A)

No.	Initial Configuration (<i>Set A</i>)	Value Used
1.	GPU_COUNT	1
2.	TRAIN_ROIS_PER_IMAGE	128
3.	DETECTION_MIN_CONFIDENCE	0.7
4.	IMAGE_MIN_DIM IMAGE_MAX_DIM	800 1024
5.	ITERATIONS	100

3.4 Transfer Learning

The backbone network of the Mask R-CNN enables low-level features to be learned in the earlier layers (layers prior to layer 3) of the overall framework. Thus, a pre-trained model can be utilised to preserve the knowledge base of the low-level features [13]. The Mask R-CNN framework implemented in this project was trained on the Microsoft COCO Dataset. The domain similarity between the images in this dataset and the road driving scene enables the weights in the earlier layers to be preserved while the last few layers of each network within the framework can be trained to fine tune the framework, conserve computation load and shorten training time [2]. Typically, these last few layers that lead to the final output in each network within the framework are referred to as the network heads.

3.5 Microsoft COCO Dataset

This dataset is split into 3 parts, namely the training dataset, the validation dataset and test dataset. The Microsoft COCO 2017 [1] Train Images dataset was used as the training dataset. This dataset contains 118000 annotated images as shown in Figure 3.8 (b). The Microsoft COCO 2017 Val Images dataset was used as the validation dataset. This dataset contains 5000 annotated images. Lastly, the Microsoft COCO 2017 Test Images dataset was used as the test dataset. The purpose of the validation dataset is to give approximations of generalization errors at each

configuration setting. Thus, the model does not learn off of the validation dataset as opposed to the training dataset.



(a) Original image of a bus
and a person walking

(b) Fine yellow lines
showing the annotations of the
image

Figure 3.8: An original image and its annotated version from the Microsoft COCO Train 2017 dataset [1]

CHAPTER 4: RESULTS AND DISCUSSION

4.1 Hyperparameter fine-tuning

This project focuses on implementing hyperparameter configuration fine-tuning to improve on the existing framework. Thus, a few hyperparameters were tested by changing their respective values. To accomplish this, eight different sets were used, each to represent one or two changes in a particular hyperparameter configuration. Table 4.1 lists the hyperparameter configurations used in *Set A* - *Set H*. In Table 4.1, the 'Hyperparameter Configurations' column lists the hyperparameters that were tested in this project. For each 'Set' column, the hyperparameter configuration that was tested is represented by a grey cell. The default value of that cell can be checked by referring to the corresponding cell in *Set A* column. Thus, *Set A* is used as the base model to be compared with in this project. However, should a hyperparameter configuration prove to improve the performance of the framework, its value is preserved for the succeeding sets in an effort to build on the improvement with other combination of hyperparameter configurations. The values of the hyperparameters that were altered in each set are highlighted in grey in Table 4.1. Sub-section 4.1.1. discusses the alteration of the DETECTION_MIN_CONFIDENCE hyperparameter configuration

Table 4.1: Hyperparameter configurations comparison

Hyperparameters Configurations	<i>Set A</i>	<i>Set B</i>	<i>Set C</i>	<i>Set D</i>	<i>Set E</i>	<i>Set F</i>	<i>Set G</i>	<i>Set H</i>
ITERATIONS	100	100	500	500	500	500	500	500
VALIDATION_STEPS	50	50	1000	1000	1000	1000	1000	1000
RPN_ANCHOR_SCALES	(32, 64, 128, 256, 512)	(32, 64, 128, 256, 512)	(32, 64, 128, 256, 512)	(32, 64, 128, 256, 512)	(32, 64, 128, 256, 512)	(8, 16, 32, 64, 128)	(32, 64, 128, 256, 512)	(32, 64, 128, 256, 512)
RPN_TRAIN_ANCHORS_PER_IMAGE	256	256	256	256	256	256	256	512
TRAIN_ROIS_PER_IMAGE	128	128	128	128	128	128	128	512
ROI_POSITIVE_RATIO	0.33	0.33	0.33	0.33	0.33	0.33	0.66	0.33
DETECTION_MIN_CONFIDENCE	0.7	0.9	0.9	0.9	0.9	0.9	0.9	0.9
DETECTION_NMS_THRESHOLD	0.3	0.3	0.3	0.3	0.6	0.3	0.3	0.3
LEARNING_RATE	0.002	0.002	0.002	0.0002	0.0002	0.0002	0.0002	0.0002

4.1.1 Detection_Min_Confidence

The DETECTION_MIN_CONFIDENCE hyperparameter sets the minimum confidence level required for the model to make a prediction. As shown in Table 4.2, the overall loss for training using *Set B* is higher than *Set A* by only 0.13%. Thus, there is no bias error in the model. However, as shown in Table 4.3, the difference in overall validation losses between *Set A* and *B* is much higher at 30%. This prompts an inspection for variance error in the model between the training and validation for each configuration set as shown in Table 4.4. From Table 4.4, it is evident that *Set B* has variance error due to the high overall loss difference of 36% between its training and validation processes. This may be due to the model overfitting to the training dataset at a very high DETECTION_MIN_CONFIDENCE value. Thus, the iterations hyperparameter shall be looked at in the next sub-section.

4.1.2 Iterations

Theoretically, the hyperparameter STEPS_PER_EPOCH determines the number of iterations to set for training. However, this hyperparameter is also controlled by the batch size, which is controlled by the hyperparameters GPU_COUNT and IMAGES_PER_GPU. Thus, essentially, when GPU_COUNT and IMAGES_PER_GPU are held constant, varying the STEPS_PER_EPOCH and EPOCH hyperparameters essentially sets the effective iterations for training. Iterations determine how many samples from the dataset is used in each epoch for training the framework. The researcher's hardware utilizes a single GPU. Thus, the hyperparameter GPU_COUNT is constant with a value of 1 and to enable the GPU to cope with the taxing computations required, IMAGES_PER_GPU is held constant as well, with a value of 1 while varying STEPS_PER_EPOCH, EPOCHS and VALIDATION STEPS. Hence, as shown in Table 4.1, for *Set C*, the iterations were increased to 500 and the VALIDATION STEPS were increased to 1000 for training. Figure 4.1 shows a plot of the training loss against iterations for *Set A-Set H*. For *Set A* and *Set B*, the loss values fluctuate mostly in the 1.2-1.6 range while for *Set C*, the

loss value started off high, then decreased and eventually plateaued at the 1.2-1.4 range. Additionally, as shown in Table 4.2 and Table 4.3, the overall loss of *Set C* compared to *Set A* reduced by 1.58 % for training and 30.43 % for validation. Since *Set C* has shown some improvement by converging to a lower loss range, *Set C* was selected for the testing phase. However, as shown in Table 4.5, the test dataset accuracy of *Set C* dropped by 3.9 % compared to *Set A*. Due to not having an improvement in test dataset accuracy, further fine-tuning of the hyperparameter is required. This prompts the researcher to look at the learning rate hyperparameter, discussed in the next sub-section.

4.1.3 Learning Rate

For *Set D*, a learning rate of 0.0002 was used. A lower learning rate would allow the framework to potentially choose a more optimum solution in the gradient descent. As shown in Figure 4.1, reducing the learning rate improved the performance of the framework as *Set D* achieved the lowest loss values among the four sets used. Additionally, as shown in Table 4.2 and Table 4.3, *Set D* obtained overall training and validation loss values that are lower by 36.31% and 29.68% compared to *Set A*. Furthermore, as shown in Table 4.4, the difference between *Set D*'s training and validation losses is just 12.95 %, lower than the 22.53% achieved by *Set C*. Hence, *Set D* was selected for the testing phase. As shown in Table 4.5, *Set D* achieved an improved performance on accuracy with an accuracy of 36.95%, representing a 4.01% improvement to *Set A*. To further experiment on other hyperparameters that may contribute to an improvement in performances, *Set E*, - *Set H* were introduced and discussed upon in the following sub-sections.

4.1.4 Detection_Non-Max Supression_Threshold

The DETECTION_NMS_THRESHOLD hyperparameter sets the minimum confidence value of a detection required to keep the ROI proposals for training. Thus, if a particular object of interest contains more than one ROI proposal, the ones that are above the threshold value are compared and the one with the highest confidence value

is chosen. *Set E* utilizes a higher value for this hyperparameter. As shown in Figure 4.1, *Set E* performs better than *Set A* but still not matching the performance of *Set D*.

4.1.5 Rpn_Anchor_Scales

This hyperparameter sets the predefined size of anchors used by the RPN. This is done by controlling the ratio of the sides of the anchor boxes. *Set F* utilizes a smaller set of scales. As shown in Figure 4.1, the performance of the framework has deteriorated significantly and show no signs of converging losses. Thus, this hyperparameter clearly does not contribute to an improvement in performance of the framework.

4.1.6 ROI_Positive_Ratio

A ROI is considered positive if its IOU value is above 0.5. Thus, this hyperparameter sets the number of positive ROIs to be used to determine loss values. Therefore, a lower loss performance upon increasing the ROI_POSITIVE_RATIO may not necessarily indicate a better performing model as only the number of positive losses used to compute the loss values increase and are reflected in the average. As shown in Figure 4.1, *Set G* which utilizes a higher ROI_POSITIVE_RATIO does not indicate any improvement in performance. On the other hand, this set does not show a deterioration in performance either.

4.1.7 Rpn_Train_Anchors_Per_Image and Train_Rois_Per_Image

These two hyperparameters were chosen to be tuned in tandem because they are inter-relatable. The RPN_TRAIN_ANCHORS_PER_IMAGE sets the number of anchor boxes to be used to detect ROIs in images. Thus, increasing the number of anchors naturally calls for increasing the TRAIN_ROIS_PER_IMAGE hyperparameter as well. As shown in Table 4.1, *Set H* implements an increase in both these hyperparameters to 512. As shown in Figure 4.1, the loss performance of *Set H* is better than the other sets, except for *Set D* where almost similar performance is recorded.

Additionally, as shown in Table 4.4, the difference between the training and validation in Set H is just 4.88, where the negative value is an indication that the validation set performed better than the training set. Thus, this set was selected for Testing. As shown in Table 4.5, the accuracy performance of *Set H* improved by 2.93 % compared to *Set A* but did not surpass that of *Set D*. Thus, in the eight sets experimented on, *Set D* achieved the highest accuracy as shown in Table 4.5.

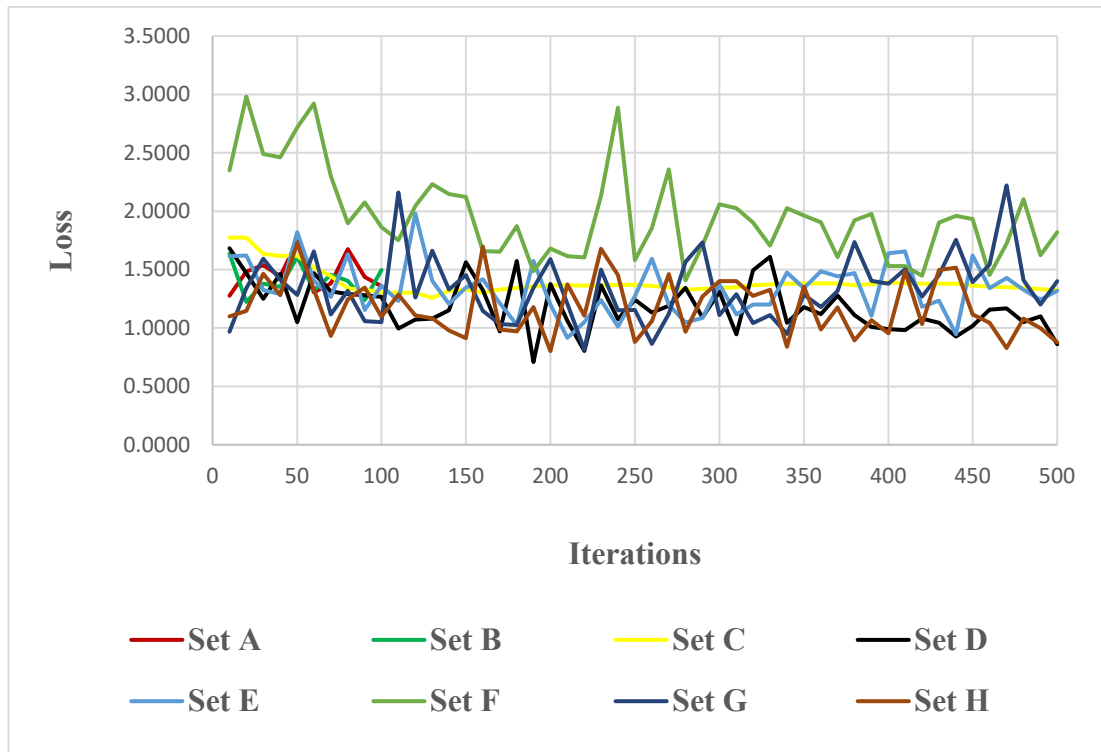


Figure 4.1: Loss vs Iterations Graph for Configuration *Set A-Set H*

Table 4.2: Training Losses Comparison between Configurations *Set A-Set H*

Process	Loss Type	Configurations Sets								Difference						
		A	B	C	D	E	F	G	H	A-B	A-C	A-D	A-E	A-F	A-G	A-H
Train	Overall loss	1.41	1.41	1.40	1.05	1.32	1.84	1.34	1.17	0.13%	-1.58%	-36.31%	-9.14%	42.91%	-7.43%	-24.36%

Table 4.3: Validation Losses Comparison between Configurations *Set A- Set H*

Process	Loss Type	Configurations Sets								Difference						
		A	B	C	D	E	F	G	H	A-B	A-C	A-D	A-E	A-F	A-G	A-H
Validati-on	Overall validation loss	1.48	1.78	1.17	1.18	0.59	3.08	2.06	1.12	30.13%	-30.43%	-29.68%	-88.03%	160.85%	58.23%	-35.57%

Table 4.4: Differences between Train and Validation (Val)**Losses for Each Set of Configurations**

Loss Type		Overall loss
Set A	Train	1.41
	Val	1.48
Difference		6.32%
Set B	Train	1.41
	Val	1.78
Difference		36.33%
Set C	Train	1.40
	Val	1.17
Difference		-22.53%
Set D	Train	1.05
	Val	1.18
Difference		12.95%
Set E	Train	1.32
	Val	0.59
Difference		-72.56%
Set F	Train	1.84
	Val	3.08
Difference		124.27%
Set G	Train	1.34
	Val	2.06
Difference		71.98%
Set H	Train	1.17
	Val	1.12
Difference		-4.88%

Table 4.5: Accuracy differences between *Set A*, *Set C*, *Set D* and *Set H*

Process	Accuracy	Configurations Sets				Difference		
		A	C	D	H	A-C	A-D	A-H
Testing	mean Average Precision (mAP)	32.94%	29.04%	36.95%	35.87%	-3.90%	4.01%	2.93%

4.2 Benchmarking

As shown in Table 4.6, the Mask R-CNN framework that is used in this project is benchmarked with 4 other frameworks, namely a YOLO-v2 [2], a Mask R-CNN [21] framework, a Faster R-CNN with ZFNet [3] and a Faster R-CNN with ResNet-101 [16]. The (mAP) is the metric used to benchmark the framework used in this project with that of authors. The mAP values for items 3, 4 and 5 in Table 4.5 are computed using IoU thresholds [1] of 0.5 to 0.95. Thus, these are indicated as mAP @ [0.5:0.95]. On the other hand, for item 2 in Table 4.5, the mAP is computed using IoU threshold of 0.5 and is indicated as mAP @ [0.5]. The accuracy of the framework used in this project is obtained by evaluating on the Microsoft COCO Val2017 dataset and is highlighted in grey in Table 4.6.





In Table 4.7, the performance of the Mask R-CNN framework that is implemented in this project is compared visually with the Mask R-CNN in [21]. This is done by taking three similar Malaysian road footage test images and running the images through the frameworks and comparing the predictions on the output images. As shown in Table 4.7, visually, the differences between the two frameworks are barely noticeable.



Table 4.6: The framework used in this project is benchmarked with other state-of-the-art frameworks

No.	Authors	Framework	Dataset	GPU	Accuracy (mAP)
1	Ye, Xing Yu Chen, Hung Hao Hsiao, Pei Yung Hong, Dza Shiang Fu, Li Chen	YOLOv2	Custom dataset, publicly available	Nvidia GeForce GTX 1070	97.6 % mAP

No.	Authors	Framework	Dataset	GPU	Accuracy (mAP)
2	Ren, Shaoqing He, Kaiming Girshick, Ross Sun, Jian	Faster R-CNN with ZFNet	PASCAL VOC 2007	NVIDIA Tesla K40	59.9% mAP @ [0.5]
3	He, Kaiming Gkioxari, Georgia Dollár, Piotr Girshick, Ross	Mask R-CNN	Microsoft COCO	Nvidia Tesla M40	38.2% mAP @ [0.5:0.95]
4	He, Kaiming Zhang, Xiangyu Ren, Shaoqing Sun, Jian	Faster R-CNN with ResNet-101	Microsoft COCO	-	37.4% mAP @ [0.5:0.95]
5	This Project	Mask R-CNN	Microsoft COCO	Nvidia GeForce 930MX	36.95% mAP @ [0.5:0.95]

Table 4.7: Performance Comparison of Images with Benchmarked Mask R-CNN Framework

No .	Benchmark Framework Test Image	This Project Test Image
1.		
2.		

No .	Benchmark Framework Test Image	This Project Test Image
3.		

CHAPTER 5: CONCLUSION

5.1 Conclusion

5.1.1 Achieved Objectives

The three objectives of this project were to identify a suitable road footage dataset, improve on a pixel-level neural network classification model using semantic segmentation and benchmark the proposed method with state-of-the-art methods. Through the selection of the publicly available Microsoft COCO dataset, the first objective of this project was achieved. Secondly, the framework implemented in this project was improved upon within the scope of the researcher's hardware configurations by utilizing the hyperparameter fine-tuning method. Thirdly, the Mask R-CNN framework implemented in this project was benchmarked with other state-of-the-art methods.

5.1.2 Project Limitations

The limitation encountered in this project is the computation power of the researcher's hardware configurations. Although the computation time problem has been addressed via the implementation of transfer learning, it was necessary to reduce the hyperparameters of the framework in order to match the researcher's hardware capabilities. Thus, the achieved accuracy for the initial set (*Set A*) is lower than in [21]. Since *Set A* was used as the initial set to which improvement experiments were compared with, the proposed values are still in the neighbourhood of *Set A*. Another limitation encountered by the researcher is the initial time-intensive task of development environment preparation. Each framework has its respective set of required libraries in the development environment. Hence, obtaining and ensuring the correct match of libraries and their dependencies utilizes in the development environment exhausted a lot of time from the schedule of the project. This lost time could have been repurposed for the experimentation phase of the project.

5.1.3 Recommendation

Based on the experimentation phase of this project, the hyperparameter configurations that improves the accuracy performance of the framework is Set D. Thus, this set of hyperparameter configurations would also be recommended for researchers implementing this framework on a similar hardware and virtual development environment set up as the one used in this project. In order to overcome the limitations encountered by the researcher in this project, the researcher recommends the usage of a more powerful GPU. This would eliminate the necessity of the researcher to dial down the hyperparameter configurations of the framework in order to match the capabilities of the machine used, Thus, a closer performance to that of the author providing the source code can be achieved and used as the initial set to base further experiments on. On the other hand, the use of Docker Containers to obtain the required the development environment for the framework saves a lot of time. Hence, this time-saving can be repurposed for the experimentation phase of the project schedule. On the other hand, researchers may take up projects that focus on benchmarking the performance of using cloud-based development environments against using local machine-based development environments as there is no clear performance metrics available to compare the two set-ups. Besides, such work may pave the way for future researchers to consider the utilization of cloud-based development environments with the much anticipated introduction of 5G technologies that may rival data read/ write speeds of local hard disks.

REFERENCES

- [1] Y. Zhang, H. Chen, Y. He, M. Ye, X. Cai, and D. Zhang, "Road segmentation for all-day outdoor robot navigation," *Neurocomputing*, vol. 314, pp. 316–325, Nov. 2018, doi: 10.1016/j.neucom.2018.06.059.
- [2] X. Y. Ye, D. S. Hong, H. H. Chen, P. Y. Hsiao, and L. C. Fu, "A two-stage real-time YOLOv2-based road marking detector with lightweight spatial transformation-invariant classification," *Image Vis. Comput.*, vol. 102, p. 103978, 2020, doi: 10.1016/j.imavis.2020.103978.
- [3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017, doi: 10.1109/TPAMI.2016.2577031.
- [4] X. Li, T. Lai, S. Wang, Q. Chen, C. Yang, and R. Chen, "Weighted feature pyramid networks for object detection," *Proc. - 2019 IEEE Intl Conf Parallel Distrib. Process. with Appl. Big Data Cloud Comput. Sustain. Comput. Commun. Soc. Comput. Networking, ISPA/BDCloud/SustainCom/SocialCom 2019*, pp. 1500–1504, 2019, doi: 10.1109/ISPA-BDCloud-SustainCom-SocialCom48970.2019.00217.
- [5] H. Yu *et al.*, "Methods and datasets on semantic segmentation: A review," *Neurocomputing*, vol. 304, pp. 82–103, 2018, doi: 10.1016/j.neucom.2018.03.037.
- [6] John Vijay, K. Kidono, C. Guo, H. Tehrani, S. Mita, and K. Ishimaru, "Fast road scene segmentation using deep learning and scene-based models," *Proc. - Int. Conf. Pattern Recognit.*, vol. 0, pp. 3763–3768, 2016, doi: 10.1109/ICPR.2016.7900220.
- [7] J. Tang and X. Wang, "Three-Skips CNN for road scene semantic segmentation," *2017 IEEE Int. Conf. Inf. Autom. ICI A 2017*, no. July, pp. 858–863, 2017, doi: 10.1109/ICInfA.2017.8079023.
- [8] X. Zhang, Z. Chen, D. Lu, and X. Li, "Real-time Semantic Segmentation for Road Scene," *ICARM 2018 - 2018 3rd Int. Conf. Adv. Robot. Mechatronics*, pp. 19–23, 2019, doi: 10.1109/ICARM.2018.8610868.

- [9] A. Nurhadiyatna and S. Loncaric, "Multistage shallow pyramid parsing for road scene understanding based on semantic segmentation," *Int. Symp. Image Signal Process. Anal. ISPA*, vol. 2019-Septe, pp. 198–203, 2019, doi: 10.1109/ISPA.2019.8868554.
- [10] X. Zhu, X. Zhang, X. Y. Zhang, Z. Xue, and L. Wang, "A novel framework for semantic segmentation with generative adversarial network," *J. Vis. Commun. Image Represent.*, vol. 58, pp. 532–543, 2019, doi: 10.1016/j.jvcir.2018.11.020.
- [11] X. Song, T. Rui, S. Zhang, J. Fei, and X. Wang, "A road segmentation method based on the deep auto-encoder with supervised learning," *Comput. Electr. Eng.*, vol. 68, pp. 381–388, May 2018, doi: 10.1016/j.compeleceng.2018.04.003.
- [12] P. Meletis and G. Dubbelman, "Training of Convolutional Networks on Multiple Heterogeneous Datasets for Street Scene Semantic Segmentation," *IEEE Intell. Veh. Symp. Proc.*, vol. 2018-June, no. Iv, pp. 1045–1050, 2018, doi: 10.1109/IVS.2018.8500398.
- [13] H. N. Latha, S. Rudresh, D. Sampreeth, S. M. Otageri, and S. S. Hedge, "Image understanding : Semantic Segmentation of Graphics and Text using Faster-RCNN," *2018 Int. Conf. Networking, Embed. Wirel. Syst.*, pp. 1–6, 2018.
- [14] Y. Zhang, Z. Qiu, T. Yao, D. Liu, and T. Mei, "Fully Convolutional Adaptation Networks for Semantic Segmentation," 2018, doi: 10.1109/CVPR.2018.00712.
- [15] M. D. Sulistiyo *et al.*, "Attribute-aware Semantic Segmentation of Road Scenes for Understanding Pedestrian Orientations," *IEEE Conf. Intell. Transp. Syst. Proceedings, ITSC*, vol. 2018-Novem, pp. 2698–2703, 2018, doi: 10.1109/ITSC.2018.8569372.
- [16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.
- [17] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez, "A Review on Deep Learning Techniques Applied to Semantic Segmentation," pp. 1–23, 2017, [Online]. Available: <http://arxiv.org/abs/1704.06857>.
- [18] R. Girshick, "Fast R-CNN," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2015 Inter,

- pp. 1440–1448, 2015, doi: 10.1109/ICCV.2015.169.
- [19] A. Bansal, X. Chen, and B. Russell, “Pixelnet.”
 - [20] S. Di, H. Zhang, C. G. Li, X. Mei, D. Prokhorov, and H. Ling, “Cross-Domain Traffic Scene Understanding: A Dense Correspondence-Based Transfer Learning Approach,” *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 3, pp. 745–757, 2018, doi: 10.1109/TITS.2017.2702012.
 - [21] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 386–397, 2020, doi: 10.1109/TPAMI.2018.2844175.
 - [22] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Region-Based Convolutional Networks for Accurate Object Detection and Segmentation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 1, pp. 142–158, 2016, doi: 10.1109/TPAMI.2015.2437384.
 - [23] B. Shuai, Z. Zuo, B. Wang, and G. Wang, “DAG-Recurrent Neural Networks for Scene Labeling,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 3620–3629, 2016, doi: 10.1109/CVPR.2016.394.

WEBSITE REFERENCES

- [1] M. COCO, “COCO Dataset,” [Online]. Available: <https://cocodataset.org/#download>.
- [2] W. Abdulla, “Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow,” *GitHub repository*, no. matterport_maskrcnn_2017, 2017.
- [3] K. Majek, “Mask R-CNN for Object Detection and Segmentation,” *GitHub repository*, 2017.
- [4] NVIDIA, “CUDA Toolkit Documentation,” [Online]. Available: <https://docs.nvidia.com/cuda/>.

[5] P. SHARMA, “Analytics Vidhya,” [Online]. Available: <https://www.analyticsvidhya.com/blog/2018/12/practical-guide-object-detection-yolo-framework-python/>.