Proposal of a Segmentation Algorithm using Multi-Sensor Fusion for Autonomous Driving in Snowy Environments

積雪環境における自律走行を目的としたマルチセンサフュージョンによるセグメンテーションアルゴリズムの提案

Sirawich Vachmanus

Graduate School of Engineering

Hokkaido University

This dissertation is submitted for the degree of

Doctor of Philosophy

August 2022

ACKNOWLEDGEMENTS

First of all, I greatly appreciate my supervisor, Assoc. Prof. Takanori Emaru, who allowed me to study at Hokkaido University and do research in this Robotics and Dynamics laboratory. He has always supported me in research activities and gave many great suggestions. Furthermore, during the research, I always received valuable suggestions from Prof. Yukinori Kobayashi, who advised some ideas to improve my activity. Apart from that, I am especially grateful for Assis. Prof. Ankit A. Ravankar, who constantly guided me to do research, stimulated me to publish the research and checked for my mistake.

Secondly, I am sincerely thankful to all members of this laboratory for their help and support in experimenting. Some of them, Mr. Masahiro Obuchi, Mr. Yuta Sasaki, and Mr. Kazuya Minamioka, also supported me in data collecting in the experiment. Moreover, I would like to express my gratitude to everyone who supported me during my studying period, including all staff of e^3 program, my family, and my friends.

Lastly, special thanks to the Japan Society for the Promotion of Science (JSPS) KAKENHI, “Robust SLAM by Non-Uniform UGV/UAV Groups for Large Field Management” under Grant JP20K04392, for funding this project. The Sapporo city, “Alternative Technology for Traffic Guides in Sidewalk Snow Removal,” and the Snow Countermeasures Office of the Sapporo City Construction Bureau to carry out the experiment.

ABSTRACT

Currently, research and development of autonomous driving technology is being actively conducted around the world. In addition to vehicle driving technology, autonomous driving requires complex technologies and knowledge, such as sensors to recognize the environment, artificial intelligence (AI) to process vast amounts of data and make appropriate decisions, and software to integrate these technologies. However, most of these research and development efforts have been conducted in non-snowy environments, and almost none have been conducted in snowy or snow-falling environments. On the other hand, one half of Japan's land area (24 prefectures and 532 municipalities) is designated as a heavy snowfall area, and it is essential to deal with snowy and snow-falling environments to maintain the transportation infrastructure. As evidenced by the fact that the top three cities in the world with populations of 100,000 or more in terms of annual snowfall are Japanese cities, research and development of autonomous driving technology for snowy and snow-falling environments is an important issue that Japan should proactively tackle.

In recent years, Japan's declining birthrate, aging population, and labor shortages have become serious social issues, and this has led to a sharp rise in labor costs. Snow removal work in areas with heavy snowfall, such as Sapporo, is essential for maintaining the transportation infrastructure, and is greatly affected by the rising cost of labor. If autonomous driving technology can support or partially automate snow removal work by adapting to snowy and snow-falling environments, labor costs can be greatly reduced.

The challenges of autonomous driving on snow-covered roads can be broadly classified into three categories as shown below.

(1) Infrastructure for automated driving: In cold regions, signs, road boundaries, and lanes are covered with snow due to snowfall, and self-position estimation using dynamic maps (highly accurate spatial information), a commonly used automated driving infrastructure, does not function.

(2) Autonomous driving in compliance with laws and regulations: Signals cannot be recognized due to snow on the traffic signals. Snow on traffic signs prevents recognition of speed limits and road signs.

(3) Vehicle control in cold weather: Anti-skid control (vehicle stability control) for curves and sudden steering. Anti-lock control of brakes. Traction control for starting and accelerating.

For (2), the Civil Engineering Research Institute for Cold Region and other organizations are conducting research on installing covers to prevent snow from sticking to traffic signals and changing the shape and materials of signs. The same issues are being studied for (3) as for conventional vehicles, and research and development of (3) is being conducted by automobile manufacturers. Based on the above, this research aims to solve the issues related to (1) infrastructure for autonomous driving.

The dynamic map method used in general autonomous driving realizes autonomous driving by using a pre-prepared high-precision 3D map and determining where the vehicle is located on the map (i.e., self-position estimation) in real time. However, the existing self-position estimation method may not work on snow-covered roads because of the large discrepancy between the map prepared in advance (non-snow-covered environment) and the actual environment (snow-covered environment). Furthermore, for example, a four-lane road may frequently be reduced to three or two lanes due to snow accumulation during the winter, making accurate self-location estimation by the dynamic map method meaningless. In other words, the surrounding environment on a snow-covered road is not always clear. In other words, on snow-covered roads, it is necessary to recognize in real time the area that can be traveled by sensing the surrounding environment. Currently, RGB cameras and 3D-LiDAR are mainly used as sensors for human detection in autonomous driving technology, and various methods have been proposed. However, in bad weather conditions such as snow, rain, and fog, these sensors are unable to accurately detect people, and noise remains a problem. Especially in poor visibility environments such as snowfall, it is not possible to observe all features with a single sensor because the road surface is often covered with snow. Therefore, this doctoral dissertation proposes a single-input and multiple-input semantic segmentation system for snowy environments to solve the above problems. The proposed method is a multimodal RGB-T semantic segmentation framework using RGB images from RGB cameras and thermal information from thermal imaging cameras, and has unprecedentedly high detection capability as a recognition method for snowy environments.

In this paper, the first system, single-input segmentation, introduces pyramid monitoring paths that can improve the inherent network performance to work even in poor environments. The other system, multiple-input segmentation, uses thermal information as a second input and also includes a pyramid monitoring path as a decoder. The proposed method shows robust human detection capability and high mIoU on a dataset of snowy environments. The network with fusion modules, multiple inputs, and pyramid monitoring paths is a state-of-the-art network for snowfall environments.

The paper consists of six chapters, each of which is summarized below.

Chapter 1 is an introduction, providing an overview of snow removal from sidewalks in Sapporo City as background for this study and describing the automated driving technology that has been used to date. The dynamic map method used in general automatic driving realizes automatic driving by obtaining the location of the vehicle (i.e., self-position estimation) on the map in real time, using a high-precision 3D map prepared in advance. However, existing self-position estimation methods do not work on snow-covered roads because of the large discrepancy between the map prepared in advance (non-snow-covered environment) and the actual environment (snow-covered environment). In other words, on snow-covered roads, it is necessary to recognize in real time the area that can be traveled by sensing the surrounding environment. Therefore, in this paper, we propose a new multimodal RGB-T semantic segmentation method using RGB images from RGB cameras and thermal information from thermal imaging cameras.

Chapter 2 describes the research background and previous work in detail. Since this research is similar to semantic segmentation, which is a method of deep learning, computer vision and deep learning are explained, and related research is surveyed.

In Chapter 3, for the case of a single input, i.e., RGB input, we outline the proposed algorithm, describe the datasets used to train and evaluate the network, and discuss the training and results using these datasets. For single-input segmentation, we introduce a pyramid monitoring path that can improve the intrinsic network performance so that it can operate robustly even in poor environments. To quantitatively evaluate the proposed method, we compared it with existing SOTA methods such as Mask R-CNN, and found that it performs best on a dataset that assumes a snowy environment, and also best on an evaluation using a general public dataset.

Chapter 4 describes the extension of the network of Chapter 3 to multiple inputs. As in Chapter 3, it outlines the proposed algorithm, describes the datasets used to train and evaluate the network, and then describes the training and results using these datasets. The multiple-input segmentation utilizes thermal information as a second input and includes a pyramid monitoring path as a decoder. Furthermore, the methods for combining multiple information are quantitatively evaluated and the optimal combination method is identified. By quantitatively comparing the proposed method with previously proposed general methods, we found that the proposed method exhibits robust human detection capability and high mIoU on a dataset of snowfall environments. In other words, the network with fusion modules, multiple inputs, and pyramid monitoring paths is the world's first state-of-the-art network for snowfall environments.

In Chapter 5, we discuss the single-input and multiple-input semantic partitioning system for snowfall environments proposed in Chapters 3 and 4. We have already discussed that the system has unprecedented recognition capability in terms of performance, but real-time performance is very important for actual applications. Therefore, by evaluating the computation time, we found that the computation time is equivalent to other methods as a relative evaluation, and that the computation time is sufficient for real-time performance.

Chapter 6 is the conclusion, in which the results obtained in this doctoral dissertation are described.

学位論文内容の要旨

現在、世界中で自動運転技術の研究開発が盛んに行われている。自動運転は車の走行に関する技術に加え、環境を認識するためのセンサ、膨大なデータを処理し適切な判断を下す AI(人工知能)、それらを統合するソフトウェアなど、複合的な技術や知見が必要とされる。しかしながら、これらの研究開発の多くが非積雪環境を対象としており、積雪・降雪環境を対象に行われているものはほぼ無い。その一方で日本の国土の 1/2(24 道府県・532 市町村)は豪雪地帯として指定されており、交通インフラを維持するために積雪・降雪環境への対応が必要不可欠である。世界中の人口 10 万人以上の都市について年間降雪量を比較するとそのトップ 3 は日本の都市であることからも分かるとおり、自動運転技術を積雪・降雪環境へ対応させるための研究開発は我が国が積極的に取り組むべき重要な課題である。

また、近年日本において少子高齢化や労働力不足が深刻な社会問題になっており、それに伴い、人件費の高騰が起きている。札幌市のような豪雪地帯における除雪作業は、交通インフラの維持のために必須の作業であるが、この人件費高騰の影響を大きく受けている。自動運転技術を積雪・降雪環境に対応させることによって除雪作業を支援、あるいは一部自動化することができれば、人件費を大きく削減することが期待できる。

雪上路における自動運転の課題は、以下に示すように 3 つに大別される。

(1) 自動運転インフラ基盤:寒冷地域では降雪により標識・道路境界線・レーンが雪で覆われ、一般的に用いられている自動運転インフラであるダイナミックマップ(高精度な空間情報)を利用した自己位置推定が機能しない。

(2) 法規に則った自動走行:信号機に雪が付着し、信号標示が認識できない。標識に雪が付着し、速度規制や道路標識が認識できない。

(3) 寒冷地での車両制御:カーブや急ハンドル時の横滑り防止制御 (車両安定制御)。ブレーキのアンチロック制御。発進・加速時のトラクションコントロール制御。

(2)については信号機の雪固着防止カバー装着や、標識の形状・部材変更などの研究が寒地土木研究所などにおいて進められている。また(3)については従来車両と同様の課題であり、さらには自動車メーカー等における研究開発が進んでいる。以上のことから、本研究では(1)自動運転インフラ基盤に関する課題を解決する。

一般的な自動運転で用いられているダイナミックマップ方式では、事前に準備された高精度な3次元地図を利用し、その地図上のどこに自車が存在するか（＝自己位置推定）リアルタイムで求めることによって自動運転を実現している。しかしながら、雪上路では事前（非積雪環境）に作成した地図と実際の環境(積雪環境)の乖離が大きくなることから、既存の自己位置推定手法が機能しないことが考えられる。さらに、例えば4車線の道路が冬期において積雪の影響で3車線や2車線になることも頻繁にあり得るため、ダイナミックマップ方式によって正確に自己位置推定できたとしても意味をなさないことも考えられる。すなわち、雪上路においては周囲環境をセンシングすることによって走行可能領域をリアルタイムで認識しながら走行する必要がある。現在、自動運転技術における人体検知のためのセンサとして、RGB カメラや 3D-LiDAR が主に用いられており、様々な方式が提案されている。しかし、雪や雨、霧などの悪天候下では、これらのセンサで正確に人物を検出することができず、ノイズが発生するという問題が残されている。特に雪が降るような視界の悪い環境では、雪に覆われた路面が多いため、単一のセンサで全ての特徴を観察することはできない。そこで本論文では、以上のような問題を解決すべく降雪環境を対象とした単一入力と複数入力の意味分割システムを提案する。提案手法は、RGB カメラからの RGB 画像とサーモグラフィからの熱情報を用いたマルチモーダル RGB-T セマンティックセグメンテーションのフレームワークであり、積雪環境の認識方法としてこれまでに無い高い検出能力を持つものである。

本論文では、まず、単一入力セグメンテーションでは、劣悪な環境でも動作するように、本来のネットワーク性能を向上させることができるピラミッド監視路を導入している。もう一つのシステムである複数入力セグメンテーションは、熱情報を第二の入力として利用し、デコーダとしてピラミッド監視パスも含んでいる。提案された方式は、降雪環境のデータセットにおいて、ロバストな人間検出能力と高い mIoU を示す。融合モジュール、複数入力、ピラミッド監視パスを備えたネットワークは、降雪環境に対応した最先端ネットワークとなった。

本論文は全 6 章で構成されており、以下にそれぞれの章の概要を示す。

第 1 章は序論であり、本研究の背景として札幌市で実施されている歩道除雪の概要と、現在までに行われている自動運転技術について述べ、その上で本研究の目的である降雪環境におけるマルチモーダル深層学習を用いた環境認識手法について、そのコンセプトと構成を述べている。一般的な自動運転で用いられているダイナミックマップ方式では、事前に準備された高精度な 3 次元地図を利用し、その地図上のどこに自車が存在するか（＝自己位置推定）リアルタイムで求めることによって自動運転を実現している。しかしながら、雪上路では事前（非積雪環境）に作成した地図と実際の環境（積雪環境）の乖離が大きくなることから、既存の自己位置推定手法が機能しない。すなわち、雪上路においては周囲環境をセンシングすることによって走行可能領域をリアルタイムで認識しながら走行する必要がある。そこで本論文では、RGB カメラからの RGB 画像とサーモグラフィからの熱情報を用いたマルチモーダル RGB-T セマンティックセグメンテーションを新たに提案した。

第 2 章では研究背景と先行研究について詳細に述べている。本研究は深層学習の一手法であるセマンティックセグメンテーションに類するものであることから、コンピュータビジョン、深層学習などについて説明し、また関連する研究についてサーベイした。

第 3 章では、単一入力すなわち RGB 入力の場合について、提案するアルゴズムの概要、ネットワークの学習および評価を行うためのデータセットの説明、さらにそれらを用いた学習とその結果について述べている。単一入力セグメンテーションでは、劣悪な環境でもロバストに動作するように、本来のネットワーク性能を向上させることができるピラミッド監視路を導入している。この提案手法を定量的に評価するために既存の SOTA である Mask R-CNN 等の手法と比較を行い、積雪環境を想定したデータセットで最も優れた性能を持つこと、さらには一般的な公開データセットによる評価において最も優れた性能を持つことを明らかにした。

第 4 章では第 3 章のネットワークを複数入力に拡張した場合について説明している。第 3 章と同様に、提案するアルゴズムの概要、ネットワークの学習および評価を行うためのデータセットの説明、さらにそれらを用いた学習とその結果について述べている。複数入力セグメンテーションは、熱情報を第二の入力として利用し、デコーダとしてピラミッド監視パスも含んでいる。さらに、複数情報を組み合わせるための手法についても定量的に評価し、最適な組み合わせ手法を明らかにした。本提案手法をこれまで提案されている一般的な手法と定量的に比較することに

より、降雪環境のデータセットにおいてロバストな人間検出能力と高い mIoU を示すことを明らかとした。すなわち融合モジュール、複数入力、ピラミッド監視パスを備えたネットワークは、降雪環境に対応した世界初の最先端ネットワークとなった。

第 5 章では第 3 章、第 4 章で提案した降雪環境を対象とした単一入力と複数入力の意味分割システムについて考察を行う。性能の面からこれまでに無い優れた認識能力を持つことは既に議論しているが、実際のアプリケーションに応用するためにはリアルタイム性が非常に重要である。そこで計算時間に関する評価を行うことにより、相対的な評価として他手法と同等の計算時間であること、さらにはその計算時間が十分なリアルタイム性を持つことを明らかにした。

第 6 章は結論であり、本研究で得られた成果を述べている。

IEEE COPYRIGHTED PAPER

This dissertation uses and reprints the published information from the published materials of the same author. The citations of every article and proceeding paper are below.

[1] S. Vachmanus, A. A. Ravankar, T. Emaru and Y. Kobayashi, "Multi-Modal Sensor Fusion-Based Semantic Segmentation for Snow Driving Scenarios," in IEEE Sensors Journal, vol. 21, no. 15, pp. 16839-16851, 1 Aug.1, 2021, doi: 10.1109/JSEN.2021.3077029.

[2] S. Vachmanus, A. A. Ravankar, T. Emaru and Y. Kobayashi, "An Evaluation of RGB-Thermal Image Segmentation for Snowy Road Environment," 2021 IEEE International Conference on Mechatronics and Automation (ICMA), 2021, pp. 224-230, doi: 10.1109/ICMA52036.2021.9512708.

[3] S. Vachmanus, A. A. Ravankar, T. Emaru and Y. Kobayashi, "Semantic Segmentation for Road Surface Detection in Snowy Environment," 2020 59th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), 2020, pp. 1381-1386, doi: 10.23919/SICE48898.2020.9240402.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

Inclement weather conditions such as fog, rain, smoke, or snow can severely hamper drivers' visibility and pose a serious risk of accidents. In the United States, casualties due to vehicular accidents are totally more than 1.5 million annually, with 800,000 injuries [4]. Weather conditions can significantly change road surface dynamics, causing delays and warnings [5]. The identical accident risk based on the road type from 2014 to 2016 in Finland was the highest on the slushy, snowy roads and higher on expressways than two-lane and multi-lane roads [6]. Similarly, in Japan, wet and frozen road surfaces, even under clear weather conditions, can cause severe traffic accidents. Moreover, snowy weather conditions also have the highest number of injuries compared with other conditions [7]. Figure 1-1 shows some examples of the road environment in the Hokkaido prefecture of Japan. This region receives the highest annual snowfall and is covered with snow for up to four months, meaning there is a high probability of accidents and vehicle slippage. Such situations cause severe delays in transportation and loss of business. Moreover, driving in such challenging conditions with poor visibility and slippery roads is very stressful for drivers.

During the winter season, road-heating services are generally not available because of their higher costs, and most city governments employ snow graders to sweep snow off the surface, hence making piles of snow on the roadside. The snow piles are cleared out by snow removal machines during the night. This process is hazardous for pedestrians and other vehicles in the surrounding areas.

Figure 1-1 Snowy environments in Hokkaido, Japan

Recently, autonomous vehicle technology has shown a great deal of promise in improving road safety. Under clear weather conditions, autonomous driving systems can navigate the vehicle with high accuracy. In contrast to indoor environments, outdoor environments are far more challenging because of the lack of features that are supported working under such conditions [8], [9]. Moreover, challenging weather conditions can cause poor visibility, directly affecting the accuracy of the vehicle's perception, which is one of the three main functions of an autonomous system. When it comes to perception, many sensors, such as cameras, Light Detection And Ranging (LiDAR), and thermal imaging, have been used to detect the environment [10], [11], [12]. The research on road perception and recognition has gained significant importance in recent years, and many methods have been developed to support drivers' perception and recognition [13], [14]. The classical methods in road detection are based on image processing techniques [15], [16], [17]. For road recognition, image processing has become a preferred method because of the low cost of sensors, higher performance, and better computation.

At the moment, deep learning and neural network–based methods have replaced the classical approaches because of the reliability and resilience of detection. Deep learning techniques have been developed in many fields, such as fingerspelling identification [18], [19] or traffic recognition [20]. Many of the new studies employing machine learning and deep learning methods have shown improved accuracy in detection compared with the classical approaches. Semantic segmentation is one learning technique gaining a lot of attention when it comes to understanding objects in the image at the pixel level [21]. By employing millions of images for training, this method can detect the road and other objects in the images simultaneously, including objects such as traffic signs, electric poles, or even pedestrians. However, the most common methods that work well under normal conditions will fail in snowy environments. The challenge is to detect environmental features under the snow cover, an issue that arises because of the minimum separation between color pixels. In addition, snow and rain can obscure sensors, hide road signs and lane markings, and affect the car's performance. Bad weather such as snow represents a difficult test for artificial intelligence algorithms. For example, it is almost impossible to separate the road and sidewalk when both are under snow cover, or some snow piles may look similar to snow layers over parked cars on the roadside. Programs trained for detecting cars and people in sunshine and snowless environments will fail to make sense of vehicles that are topped with piles of snow and people who are wearing several layers of clothing.

In Sapporo city of Hokkaido, Japan, the city bureau uses a heavy machine called a snow remover machine to remove the snow from the road surface. This machine works by spinning some snow on the ground and throwing them in a controlled direction. Figure 1-2 shows the snow remover machine. The operation of this heavy machine requires at least two staff at the operation site. First is a driver, who drives and controls the machine. Another one is an observer. This staff has to observe the process's safety, which means looking for pedestrians or vehicles. The observer also commands the driver to drive or stop in different situations. Figure 1-3 shows the snow remover machine's operating site overview.

Figure 1-2 Snow remover machine of Sapporo city

Figure 1-3 Overview of the operating site

This work aims to develop a system that can work as an observer. The system must be able to recognize the environments of the snow remover machine. Figure 1-4 shows the overview of the snow remover machine with a detecting system. It has to classify which part is a road, snow, pedestrians, vegetation, buildings, vehicles, or obstacles. The recognition in this dissertation is based on the semantic segmentation technique, which can separate the environmental image into different regions. Due to the poor visibility environments particularly snowy conditions and snow-covered surfaces, a single sensor is not enough to observe every feature. Therefore, this dissertation utilizes multi-modal RGB-T semantic segmentation using the RGB images from an RGB camera and a Thermal map from a thermal camera for the snow remover machine task. This combination of information in snowy conditions provides excellent information about the subject.

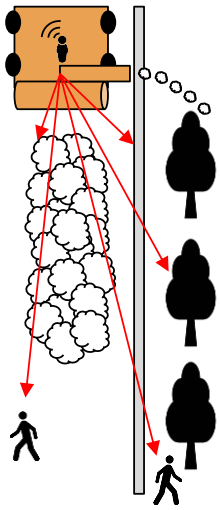


Figure 1-4 Overview of the operating site with an object detection system

CHAPTER 2

BACKGROUND KNOWLEDGE

2. BACKGROUND KNOWLEDGE

This section will explain every theoretical method used in the experiment in technical terms. The system in the experiment is based on image processing, deep learning, Robot Operating System (ROS), and computer vision technique. Many techniques are applied to approach the final goal of the research. Another part of this section will be the literature review. It will conclude many pieces of related research in the past. The original networks, the introduced network is developed from, are also explained in this part. The specifications of the equipment used in the experiment are described in the Appendix section.

2.1 Theories

a. Computer Vision

Computer Vision [22] transforms information from a camera or video camera into a new representation. It deals with how the machine can provide a deep understanding of digital images or videos. The input data generally include some contextual information such as color image, Depth map, or Thermal map, for example, the data from the camera mounted on a car. The detection might be a person in front of the car or some trees on the right side. A computer understands an image as a grid of numbers in a machine vision system. Figure 2-1 shows a picture of the automobile, but it recognizes only a grid of numbers for the computer. Therefore, computer vision aims to make the computer understand the image the same way humans understand from this grid of numbers.

A huge problem for computer vision is noise. Typically, noise is removed by using statistical methods. For example, it is impossible to detect an edge by only comparing a point in edge detection. However, in statistics over a local region, the actual edge should appear as a string, each orientation consistent with its neighbors. Many vision problems require various tools to solve. OpenCV[23] has become a tool that helps developers deal with computer vision problems.

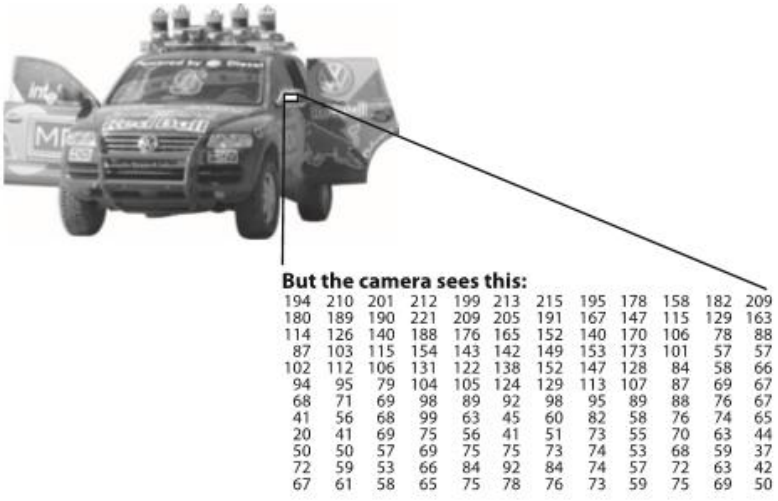


Figure 2-1 Grid of numbers from the regular image [22]

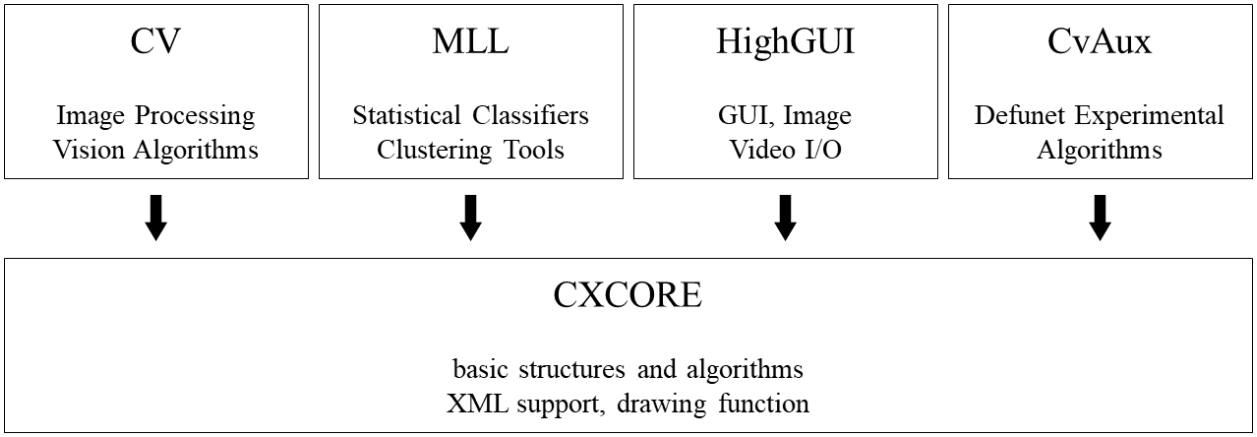


Figure 2-2 Structure of OpenCV

b. OpenCV

OpenCV [22] is an open-source computer vision library to help developers solve the vision issue. Gary Bradski designed it in 1999 to accelerate computer vision and provide a simple-to-use computer vision infrastructure. The library contains more than 500 functions in a wide area of use in vision. It consists of five main parts: CV (image processing), MLL (machine learning), HighGUI (GUI image and video I/O), CXCORE (basic structure and component), and CvAux (defunct areas and experimental algorithms). Figure 2-2 shows the OpenCV structure.

c. RGB Sensor

RGB digital camera is a camera equipped with a CMOS or CCD sensor through which the color of objects is obtained. Sensors are set in the Bayer filter arrangement, with RGB in a ratio of 1:2:1 to achieve high resolution. Figure 2-3 shows the Bayer filter arrangement. Figure 2-4 shows the CMOS sensor in the digital camera. The resolution of the camera is defined in pixels unit, which is a box of pigment shown in the image. RGB camera is designed based on human sensitivity. For example, human eyes are sensitive to red, green, and blue light. Therefore, the output of the RGB camera is an image, i.e., three layers of a value grid.

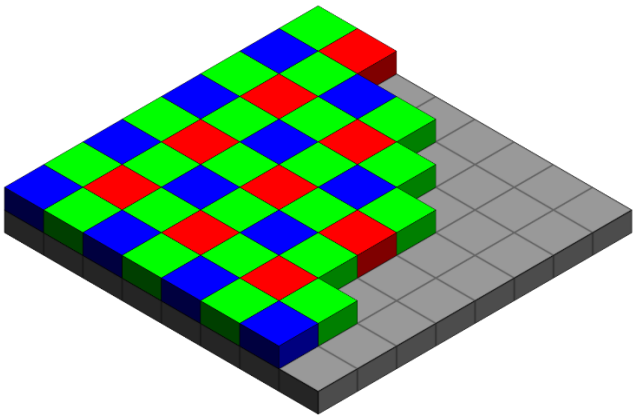


Figure 2-3 Bayer filter arrangement

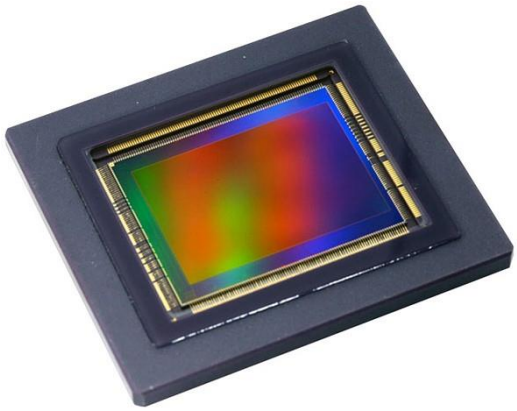


Figure 2-4 CMOS image sensor [24]

d. Infrared Sensor

Thermal images, or thermograms, are the appearance of the infrared energy emitted, transmitted, and reflected by an object. A thermographic camera (an infrared camera, thermal imaging camera, thermal camera, or thermal imager) is a device that creates an image using infrared (IR) radiation. Unlike the RGB camera that uses visible light to generate the image. The infrared cameras are sensitive to wavelengths from about 1 – 14 μm. Figure 2-5 shows the infrared wavelength in the electromagnetic spectrum. A significant difference with RGB cameras is that the focusing lenses cannot be glass due to the ability to block long-wave infrared light. Special materials, such as Germanium, calcium fluoride, or crystalline silicon must be used to ignore the problem. Figure 2-6 shows the infrared camera. Although the image approximates the object's temperature, the camera also detects the multiple infrared sources surrounding the object. The sensor's measuring thermal includes the emitted power, transmitted power, and reflected power [25].

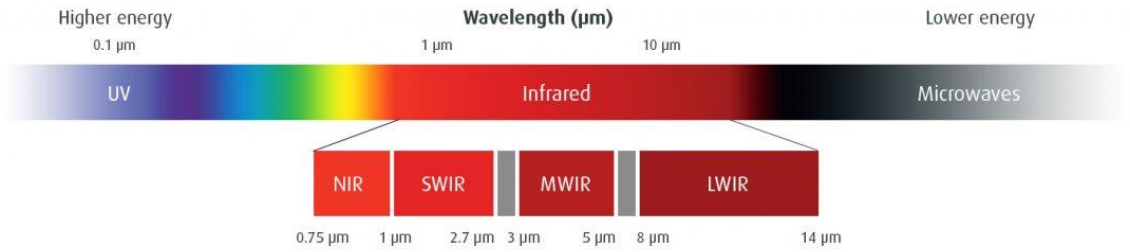


Figure 2-5 Infrared wavelength [26]

Figure 2-6 Thermal camera

e. Image Processing

In general, digital image analysis requires computer algorithms to operate on images, extract some focused information, and enhance the image. It is a type of signal processing based on two-dimensions processing. The purpose of image processing is to improve image quality. The input is a low-quality image in image processing, and the output will be an improved image. Currently, not only the technique for image processing is rapidly growing, but various computer tools for image processing are also available. A tool used in this dissertation is the OpenCV library [22], which was previously introduced in Section 2.1-b. This section will explain the processing method and technique in detail.

i) Color Space Conversion

The base color space of the image is RGB, which will be aforementioned. However, some features do not appear in focus; therefore, changing to other representative colors is useful. Three color spaces are used to extract the feature of a snowy road, including HSV, grayscale, and binary image.

Red-Green-Blue (RGB)

The RGB color space is a model represented in red (R), green (G), and blue (B) colors of light. These three primary colors can generate various kinds of colors. Figure 2-7 shows the combination of lights in RGB based. The model is for sensing, representing, and displaying the image in the computer system. An image in this model contains three layers of the component, grids of numbers, which represent red, green, and blue values. Figure 2-8 shows layers of color from an image. This part introduces the primary image that is used in this dissertation. Then, the details of the camera sensor and information collecting are explained.

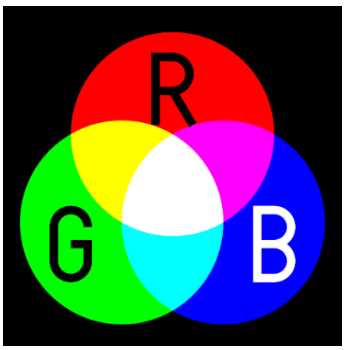


Figure 2-7 Combination of RGB light

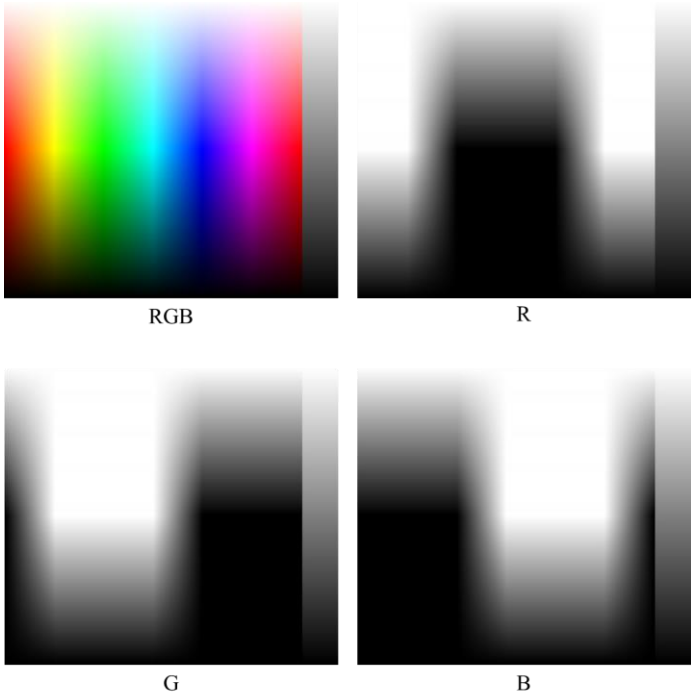


Figure 2-8 RGB layers

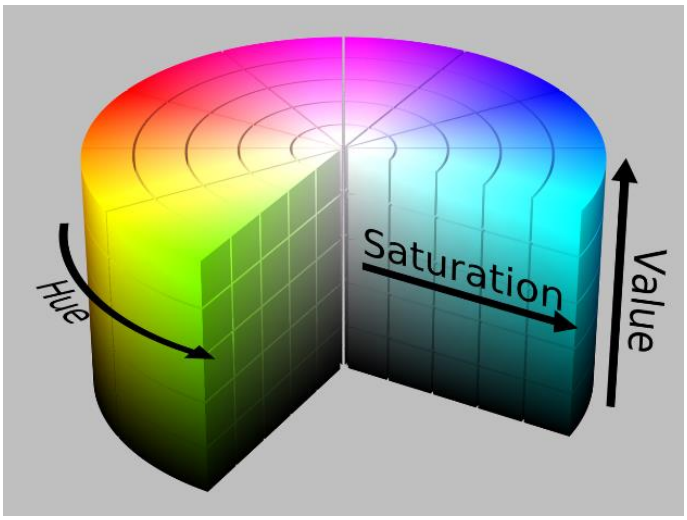


Figure 2-9 HSV cylinder

Hue-Saturation-Value (HSV)

HSV is an alternative representation of RGB color space. The color is presented in Hue, Saturation, and Value in this model. The model is usually represented in an HSV cylinder. Figure 2-9 shows the HSV cylinder. Hue is arranged in a radial slice, around a central axis of neutral colors which range from black at the bottom to white at the top.

The cylindrical geometric consists of a Hue in angular dimension, red at 0° passing through the green at 120°, and blue at 240°. The central vertical axis of Value comprises gray colors, ranging from black at value 0 at the bottom to white at value one at the top. The radius from the axis represents saturation from 0 to 1. In the conversion of RGB to HSV color space, each value of R, G, and B are used to calculate the value of H, S, and V following Eqs. (2-1), (2-2), and (2-3). Figure 2-10 shows an example of conversion to HSV color space. Where C is calculated by Eq. (2-4).

$$V = 255 \max(R, G, B) \tag{2-1}$$

$$S = \begin{cases} 255\frac{C}{V} & ; V \neq 0 \\ 0 & ; V = 0 \end{cases} \tag{2-2}$$

$$H = \begin{cases} \frac{30(G-B)}{C} & ; V = R \\ \frac{60+30(B-R)}{C} & ; V = G \\ \frac{120+30(R-G)}{C} & ; V = B \end{cases} \tag{2-3}$$

$$C = V - \min(R, G, B) \tag{2-4}$$

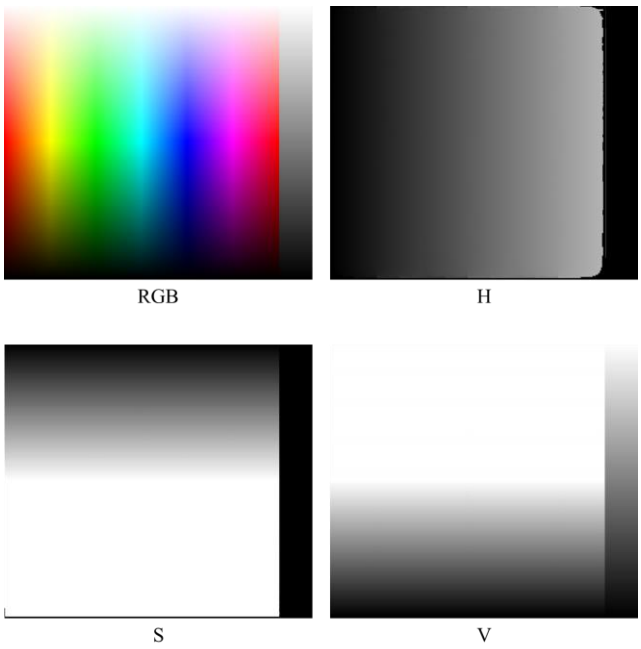


Figure 2-10 Conversion of RGB to HSV

Grayscale

The explained two color spaces, RGB and HSV, contain three layers of the component in an image. Alternatively, the grayscale image has only one layer. The grayscale image represents the amount of light, also called brightness. It carries only intensity information. The high-intensity is shown as white, and the low-intensity is shown as black in the image. The scale ranges from 0 (black) to 1 (white). Figure 2-11 shows the grayscale image.

The technique in the conversion of RGB image to grayscale image is not unique; the different weighting of color becomes a different shade of the grayscale. In this dissertation, OpenCV weight is used in the conversion. The calculation of grayscale can be done by Eq. (2-5). Figure 2-12 shows an example of the transformation from RGB to grayscale.

$$Gray = 0.299R + 0.587G + 0.114B \tag{2-5}$$

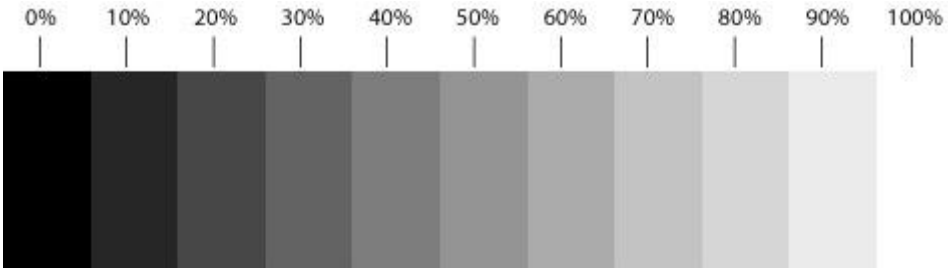


Figure 2-11 Grayscale

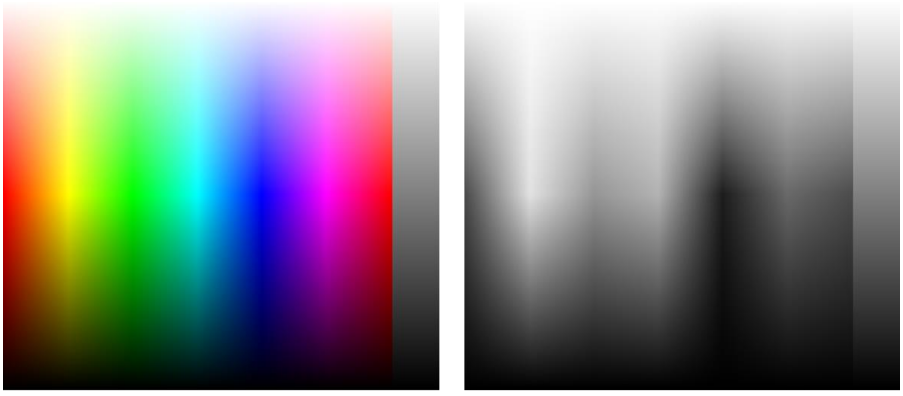


Figure 2-12 Conversion of RGB to grayscale

Binary Image

A binary image is a digital image in which only two values are available for each pixel. It represents an image in black-white color, value 0 is for black, and value 1 is for white. Figure 2-13 shows a binary image from grayscale. A pixel higher than half brightness in grayscale will be white in the binary image. Threshold operation is used to convert from grayscale images to binary images. If the value is larger than the threshold value for each pixel, it will be set to 1; otherwise, it will be set to 0. Equation (2-6) is the calculation of threshold operation. Figure 2-14 shows the value of each pixel of the binary image.

$$binary = \begin{cases} 1 \ ; & gray \geq \textbf{threshold value} \\ 0 \ ; & \textbf{otherwise} \end{cases} \tag{2-6}$$

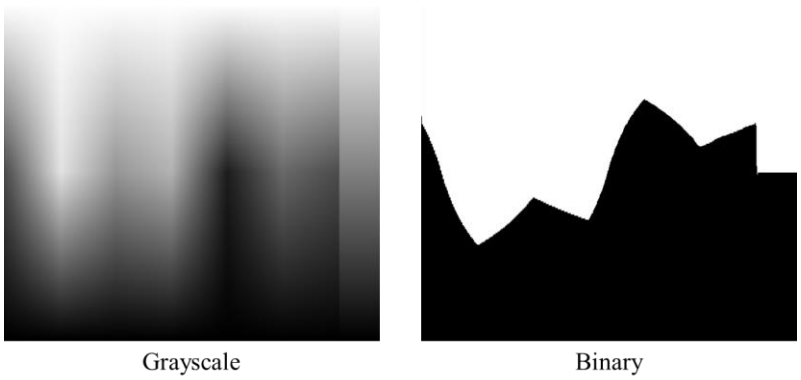


Figure 2-13 Binary image

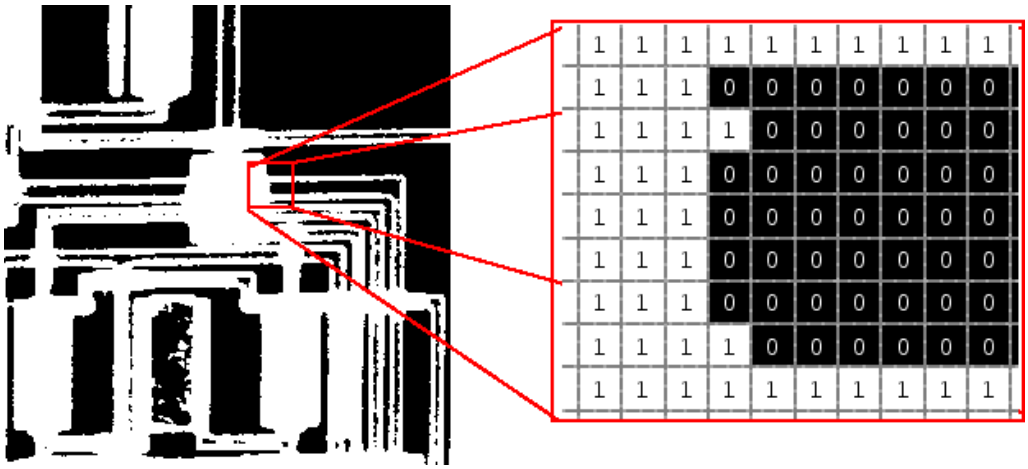


Figure 2-14 Pixel value in a binary image [27]

ii) Colormap Conversion

A colormap is a matrix of values [28] that define the colors for graphics objects such as surface, image, and patch objects. It must be three columns wide that can be any length. Each row in the matrix represents each color by using an RGB triplet. The colormap is usually used to represent the single-channel image or grayscale image for better understanding by a human. Typically, colormap contains a gradian from one color to another color. Color defines the lowest value in the image, and another shows the highest value. There are many kinds of colormap such as parula, turbo, HSV, RGB, spring, summer, autumn, winter, bone, gray, jet, etc.

In this research, the jet-colormap is chosen to represent the image from a thermal camera. The thermal camera provides the thermal image or Thermal map that contains the gradian of temperature in the environment. However, due to the only one data, temperature, the Thermal map is a grayscale image shown in a gray-colormap. Therefore, the jet-colormap that contained three channels in the image is used instead of the gray-colormap to enhance the image. The white area in the gray-colormap is converted to the red area in jet-colormap, and the black color is blue. Figure 2-15 shows the R, G, and B values changing in jet-colormap.

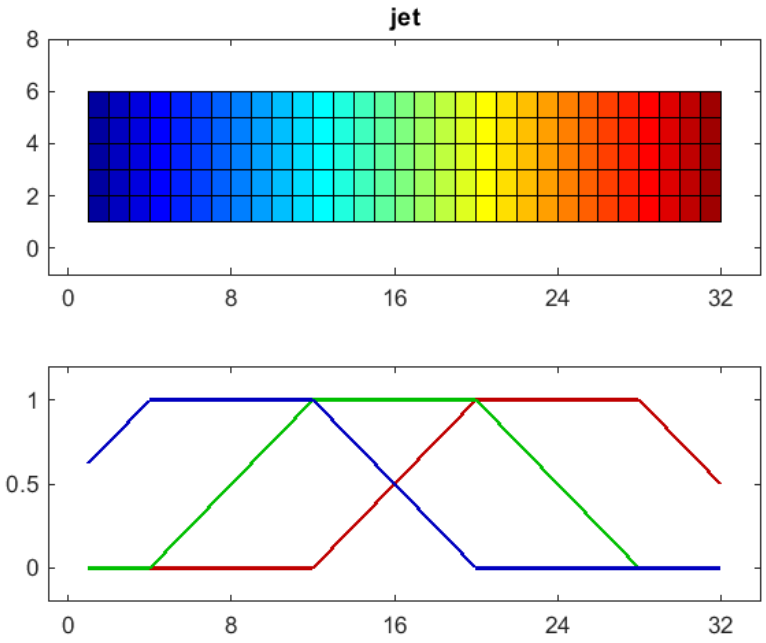


Figure 2-15 Color changing in jet-colormap [29]

iii) Histogram Equalization

Histogram equalization is one of the image processing techniques that adjusts the contrast of an image by using the histogram. The process is done by spreading out the most frequent pixel intensity values or stretching out the intensity range of the image. It can be used with the low contrast image or the image lacking highlights and shadows. Figure 2-16 shows the changing of the histogram after the equalization.

Figure 2-17 shows the example image and its histogram before and after equalization. The red bars represent the number of pixels in each intensity. The black graph is the cumulative pixel. The result of equalization is an image with more details than the input image. Therefore, the process can be said that it is defogging and enhances the contrast.

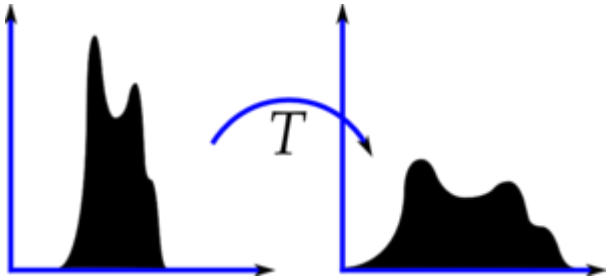


Figure 2-16 Histogram equalization chart

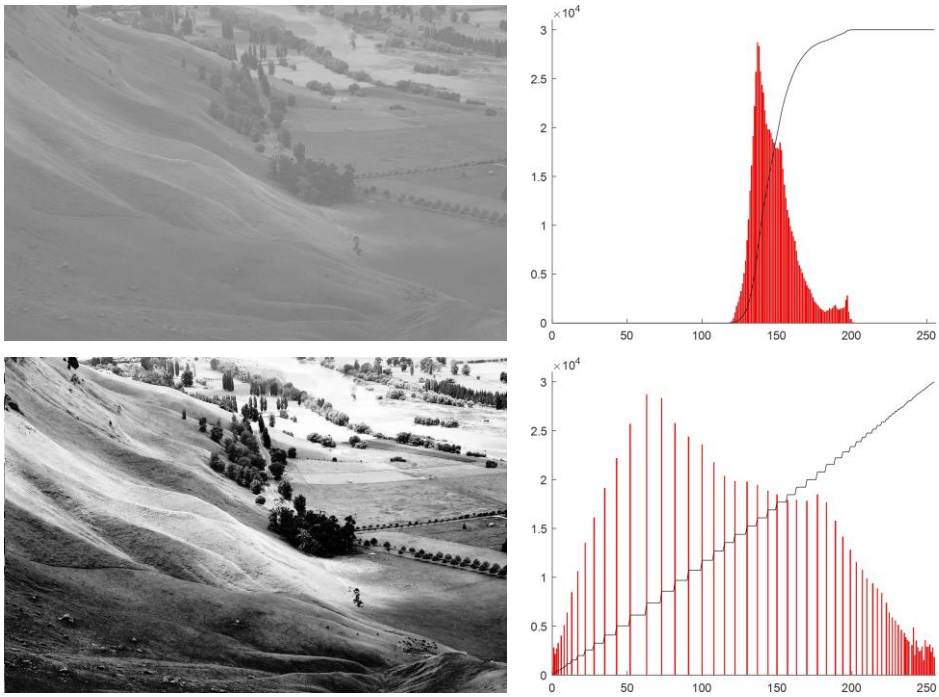


Figure 2-17 Equalization result; Upper row is the input, Lower row is the result

iv) Geometric Transformation

The geometric transformation is used as image transformation in this dissertation. It is used to modify the image shape to the new shape. In the camera calibration, the transformation is applied to match many images. The transformation consists of two primary operations in digital image processing: a spatial transformation of coordination and intensity interpolation that assigns intensity values to the spatially transformed pixels. The transformed coordinates can be described by Eq. (2-7) [30].

$$(x, y) = T\{(v, w)\} \tag{2-7}$$

Where v, w denote pixel coordinates in the original image and x, y denote corresponding pixel coordinates in the transformed image. T is a transformation operation. One of the most famous transformations is the affine transform [31]. Affine transformation is a transformation matrix used to modify the input image. It consists of two basic metrics: the rotational matrix and the translational matrix. The rotational matrix can be explained by Eq. (2-8), and the translational matrix can be shown by Eq. (2-9) [32].

$$\mathbb{R} = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix}_{2\times2} \tag{2-8}$$

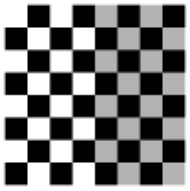
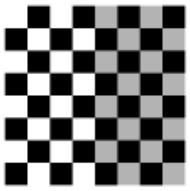
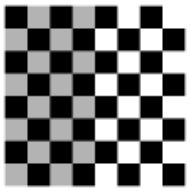
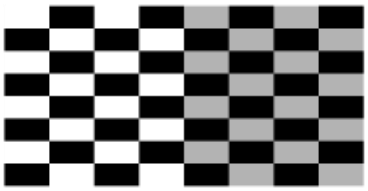
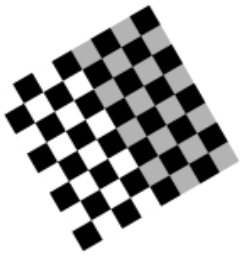
$$\mathbb{T} = \begin{bmatrix} b_{00} \\ b_{10} \end{bmatrix}_{2\times1} \tag{2-9}$$

The affine matrix is the combination of the two matrics above. The final matrix is the 3×3 matrix representing various kinds of transformations. Equation (2-10) defines the whole transformation matrix of an affine transformation.

$$\mathbb{M} = \begin{bmatrix} a_{00} & a_{01} & b_{00} \\ a_{10} & a_{11} & b_{10} \\ 0 & 0 & 1 \end{bmatrix}_{3\times3} \tag{2-10}$$

The transform relocates pixels to approximate the value of moved pixels. The examples of the application of Affine transformations, such as scale, rotate, translate, mirror, and shear, are shown in Table 2-1 [30].

Table 2-1 Example of affine transformation

Transformation name	Affine matrix ($\mathbb{M}$)	Example
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	
Translation	$\begin{bmatrix} 1 & 0 & v_x > 0 \\ 0 & 1 & v_y = 0 \\ 0 & 0 & 1 \end{bmatrix}$	
Reflection	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	
Scale	$\begin{bmatrix} c_x = 2 & 0 & 0 \\ 0 & c_y = 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	
Rotate	$\begin{bmatrix} \cos(30°) & -\sin(30°) & 0 \\ \sin(30°) & \cos(30°) & 0 \\ 0 & 0 & 1 \end{bmatrix}$	
Shear	$\begin{bmatrix} 1 & c_x = 0.5 & 0 \\ c_y = 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	

v) Interpolation

Some losses may occur when scaling the graphic image and affect the image quality. The losses are course by missing pixel information in the transformed image. Figure 2-18 shows the example of the upscaled image without interpolation. The pixels from the original image are spread off to make a larger result. Therefore, the space between each pixel occurs and makes the loss of pixels in the image. In this case, the interpolation technique is necessary to enhance the image quality and fix the missing information. The interpolation is required in digital image processing when resizing, remapping, inpainting, morphing, and non-linear transformation. There are many techniques to interpolate the image, but three of them are used in this dissertation.

Nearest-neighbor Interpolation

The nearest-neighbor interpolation or proximal interpolation is the most straightforward method that can be applied to one or multi-direction. The concept of this method is selecting the value of the nearest point as a missing point value. Figure 2-19 shows the interpolated image by nearest-neighbor operation.

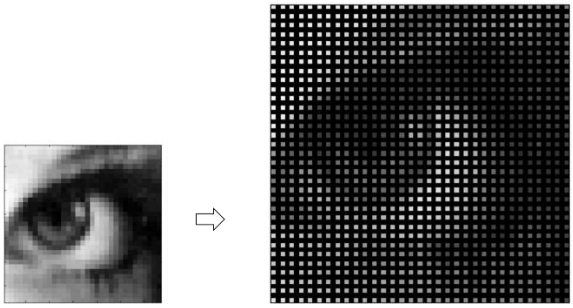


Figure 2-18 Upscaled image without interpolation

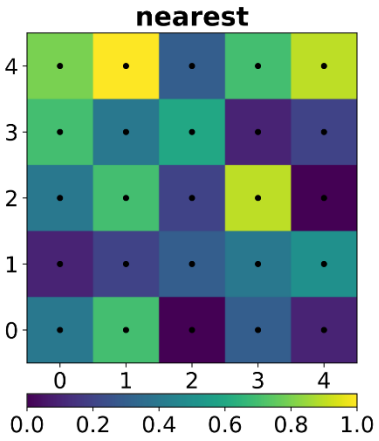


Figure 2-19 Nearest-neighbor interpolation [33]

Bilinear Interpolation

The bilinear interpolation is an interpolation of two variables using repeated linear interpolation. It is performed using linear interpolation, first in one direction and then in another direction. This interpolation is the basic resampling technique in image processing. The calculation of the bilinear approach can be expressed as a multilinear polynomial. Equation (2-11) describes a general form of a multilinear polynomial, and the two variables can be written as Eq. (2-12) and expanded as Eq. (2-13) [34].

$$f(x) = \sum_{i_1=0}^{1} \sum_{i_2=0}^{1} \cdots \sum_{i_n=0}^{1} a_{i_1 i_2 \cdots i_n} x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n} \tag{2-11}$$

$$f(x,y) = \sum_{i=0}^{1} \sum_{j=0}^{1} a_{ij} x^i y^j \tag{2-12}$$

$$f(x,y) = a_{00} + a_{10}x + a_{01}y + a_{11}xy \tag{2-13}$$

Where $f(x,y)$ is a value of the unknown function at the point (x,y), the coefficients a_{00} to a_{11} are found by solving the linear system as Eq. (2-14). Figure 2-20 shows the interpolated image by bilinear operation.

$$\begin{bmatrix} a_{00} \\ a_{10} \\ a_{01} \\ a_{11} \end{bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 & x_1 y_1 \\ 1 & x_1 & y_2 & x_1 y_2 \\ 1 & x_2 & y_1 & x_2 y_1 \\ 1 & x_2 & y_2 & x_2 y_2 \end{bmatrix}^{-1} \begin{bmatrix} f(x_1, y_1) \\ f(x_1, y_2) \\ f(x_2, y_1) \\ f(x_2, y_2) \end{bmatrix} \tag{2-14}$$

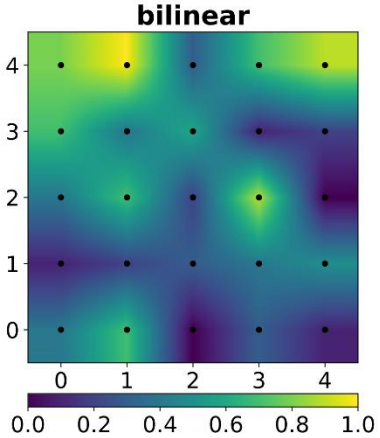


Figure 2-20 Bilinear interpolation [33]

Bicubic Interpolation

The bicubic interpolation is an interpolation method for a two-dimensional grid. This technique is an extension of cubic interpolation. This interpolation result is smoother than the other interpolation methods, nearest-neighbor and bilinear. However, the bicubic needs at least 16 pixels(4×4) to operate, while only 4 pixels(4×4) are required for bilinear. Figure 2-21 shows the interpolated image by bicubic operation.

Figure 2-22 shows the comparison of each interpolation method in both 1D and 2D interpolation. The black point in every image represents the interpolation result. The other points are the neighboring sample points. Their heights above the ground, vertical lines, correspond to their values.

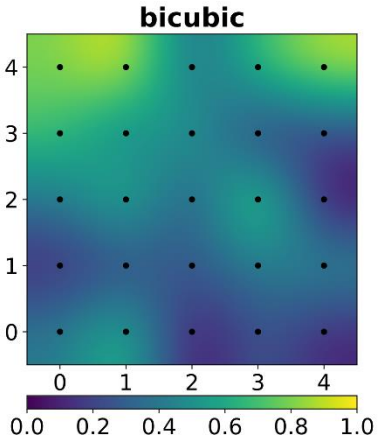


Figure 2-21 Bicubic interpolation [33]

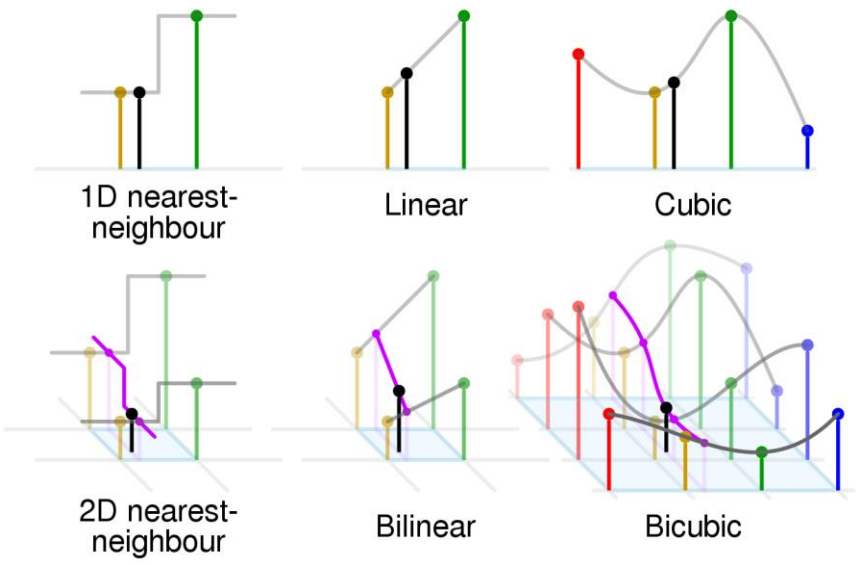


Figure 2-22 Comparison of interpolation methods [33]

vi) Camera Model

The simplest camera model used in computer vision is the pinhole camera. This model assumes light as a single ray enters from a particular point, then projected onto an imaging surface. As a result, the image on the plane is in focus, the size of the image relative to the focal length. Figure 2-23 shows the pinhole camera model, where f denotes the focal length, Z is the distance between camera and object, X is the object's height, and x is the height of the object's image. The model is represented in the form of a similar triangle. Therefore, the image height can be calculated by Eq. (2-15) [22].

$$x = -f\frac{X}{Z} \tag{2-15}$$

The original model is adapted in a similar triangle rule to make the object's image upside up. The new model can represent in a 3D world that the image is shown in a 2D image, width and height can be represented. Figure 2-24 shows the new pinhole camera model, where Q is an image point that is projected as point q on the image plane. The position (x_{screen}, y_{screen}) is calculated by Eq. (2-16) [22]. Where c_x and c_y are the displacement from the optic axis.

$$x_{screen} = f_x\left(\frac{X}{Z}\right) + c_x\,, \quad y_{screen} = f_y\left(\frac{Y}{Z}\right) + c_y \tag{2-16}$$

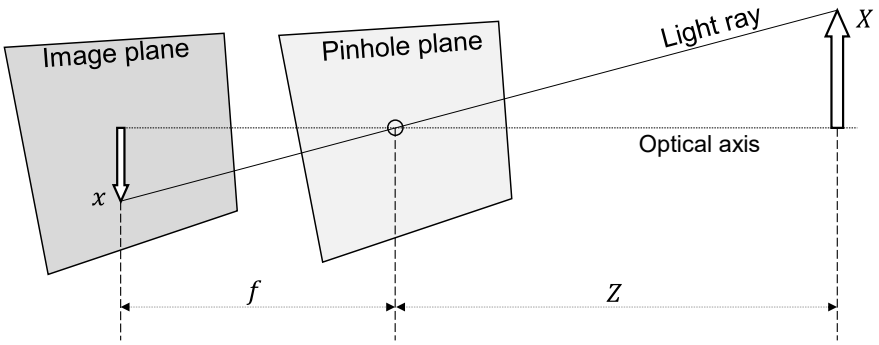


Figure 2-23 Pinhole camera model

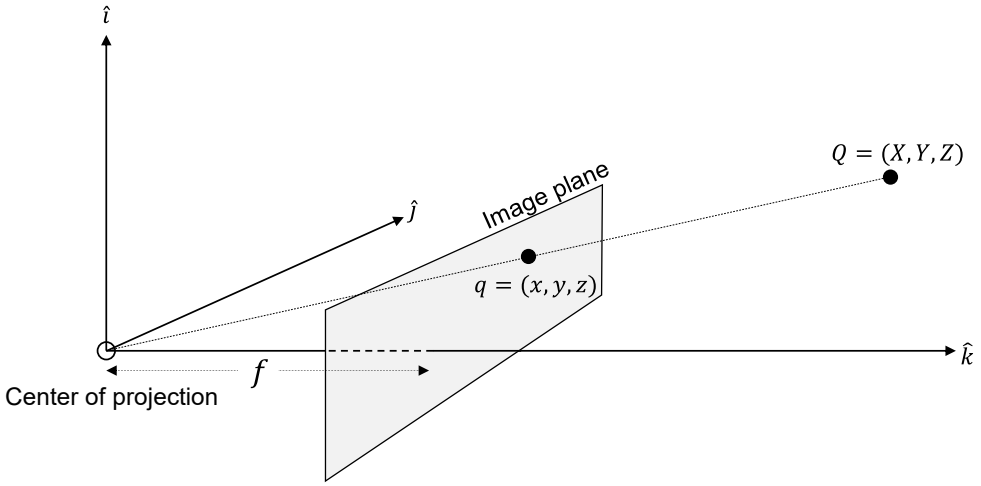


Figure 2-24 New pinhole camera model

Basic Projective Geometry

The relationship between point Q in the physical world and point q on the projection screen is called a projective transform. The homogeneous coordinates are utilized to work in the projective transform. In this case, the two dimensions image plane is the projective space, so the three-dimensional vectors, $q = (q_1, q_2, q_3)$, can be represented on the plane. All points which have proportional values in the projective space can be recovered their actual coordinates by dividing through by q_3. Therefore, the camera parameters, the focal length and the optical centers, can be arranged as a single 3×3 matrix, called the camera intrinsics matrix, expressed by Eq. (2-17). Where $\boldsymbol{M}$ is a camera matrix, as shown by Eq. (2-18) [35]. These intrinsic parameters are specific to a camera, so the camera matrix is unique to a specific camera.

$$\boldsymbol{q} = \boldsymbol{MQ} \tag{2-17}$$

$$\boldsymbol{M} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2-18}$$

Lens Distortions

In general, all camera lenses make the distortions due to the manufacturing reason. There are two kinds of distortion from the lens: radial distortions and tangential distortions. Radial distortions are the results of the shape of lenses. In contrast, tangential distortions happen from the assembly process of the whole camera.

The radial distortion usually occurs at the pixel near the edge of the image. This phenomenon is caused by the barrel or fish-eye effect [36]. Figure 2-25 shows the radial distortion model. The distortion is zero at the center of the image and increases when moving toward the periphery. To undistort or to recover the image, three constant values are used to remake the distorted image. Equations (2-19) and (2-20) explain how the corrected pixel is calculated.

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \quad (2\text{-}19)$$

$$y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \quad (2\text{-}20)$$

Where (x, y) is the original location on the image, and $(x_{corrected}, y_{corrected})$ is the new location after the correction. k_1, k_2, and k_3 represent the radial distortion coefficients. r is the distance from the center of the image that the distortion equals zero.

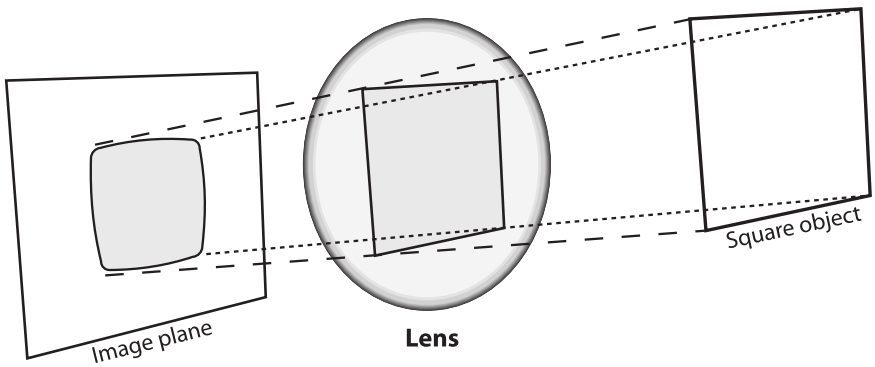


Figure 2-25 Radial distortion [22]

Another distortion, tangential distortion, is due to the manufacturing defects resulting from the not being perfectly parallel between the lens and image plane. Figure 2-26 shows the tangential distortion results when the lens is not fully parallel to the image plane. This distortion can be minimized by Eqs. (2-21) and (2-22), where p_1 and p_2 are the tangential distortion coefficients.

$$x_{corrected} = x + [2p_1y + p_2(r^2 + 2x^2)] \tag{2-21}$$

$$y_{corrected} = y + [2p_2x + p_1(r^2 + 2y^2)] \tag{2-22}$$

In practice, both distortions occurred concurrently. Thus all five distortion coefficients are required to do the undistortion process. In OpenCV, every distortion coefficient is typically bundled into a distortion vector, a 5×1 matrix as expressed in Eq. (2-23).

$$Distortion\ coefficients = (k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3) \tag{2-23}$$

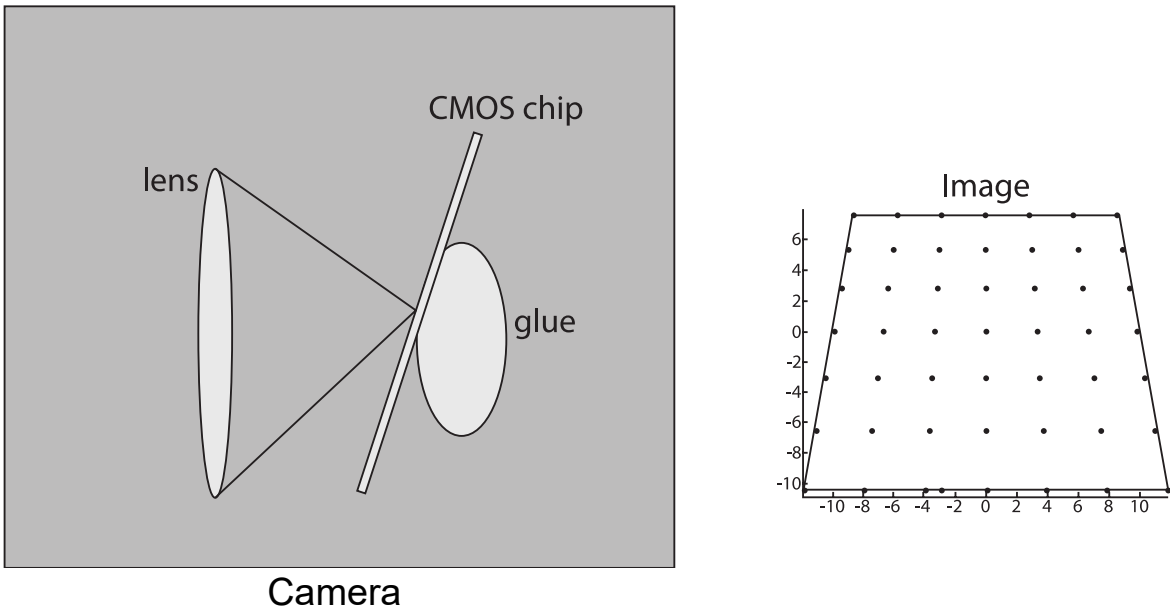


Figure 2-26 Tangential distortion [22]

f. PyTorch

PyTorch is an open-source machine learning framework that accelerates the path from research prototyping to production deployment. It is utilized on GPUs and CPUs for deep learning [37]. This library has been developed for working with the Python language. The PyTorch library is based on the Torch library used in the application of computer vision and natural language processing [38].

PyTorch provides two high-level features. First is tensor computing with strong acceleration by GPUs. Another is deep neural networks built on a type-based automatic differentiation system. Table 2-2 shows the specification information of PyTorch.

Table 2-2 Specification information of PyTorch [39]

Original authors	Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan
Developer	Facebook's AI Research lab (FAIR)
Initial release	September 2016
Stable release	Version 1.10.0 / 21 October 2021
Repository	github.com/pytorch/pytorch
Written in	Python ,C++ ,CUDA
Operating System	Linux, macOS, Windows
Platform	IA-32, x86-64
Available in	English
Type	Library for machine learning and deep learning
License	BSD
Website	pytorch.org

g. Deep Learning

Deep learning is a part of machine learning methods based on artificial neural networks. The learning can be supervised, semi-supervised, or unsupervised [40], [41]. Deep-learning architectures such as deep neural networks, deep reinforcement learning, and convolutional neural networks have been applied to various fields of use, including computer vision, natural language processing, and machine translation. They can produce results comparable to and surpass human expert performance [42], [43].

A convolutional neural network (CNN) is a class of artificial neural networks, most typically applied to analyze visual problems [44]. CNN is developed based on biological processes [45], [46], [47]. The connectivity pattern between neurons resembles the organization of the animal visual cortex. It uses less pre-processing than other image classification algorithms. The network learns to optimize the filters automatically, whereas these filters are hand-processed in traditional methods. A convolutional neural network consists of an input, hidden, and output. Figure 2-27 shows the overall structure of CNN. The hidden layers include layers that perform convolutions. Generally, this includes a layer that performs a dot product of the convolution filter with the input feature. This product is usually the Frobenius inner product, and its activation function is typically ReLU. The convolution operation generates a feature map that will be the input of the next layer. This is followed by other layers such as pooling, fully connected, and normalization layers. Figure 2-28 shows operations in the hidden layer.

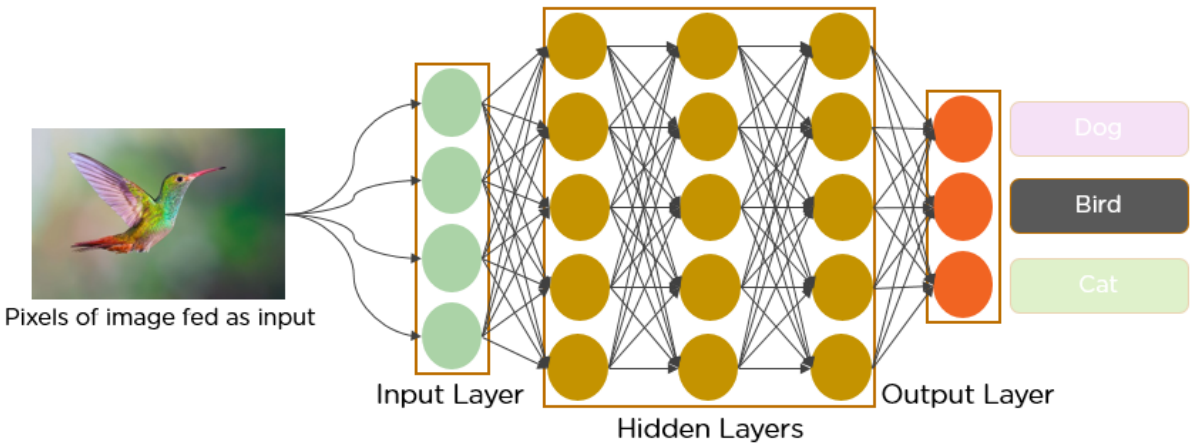


Figure 2-27 CNN structure on image classification [48]

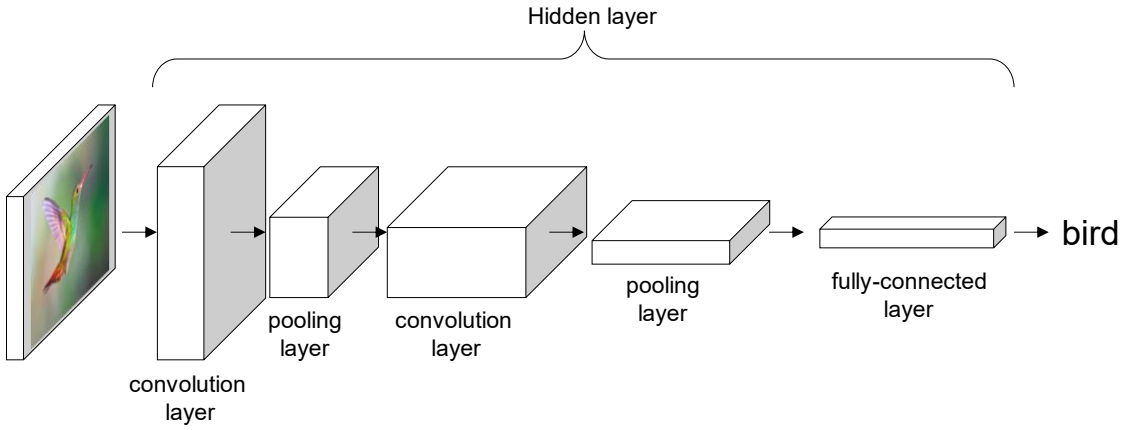


Figure 2-28 Hidden layers in CNN

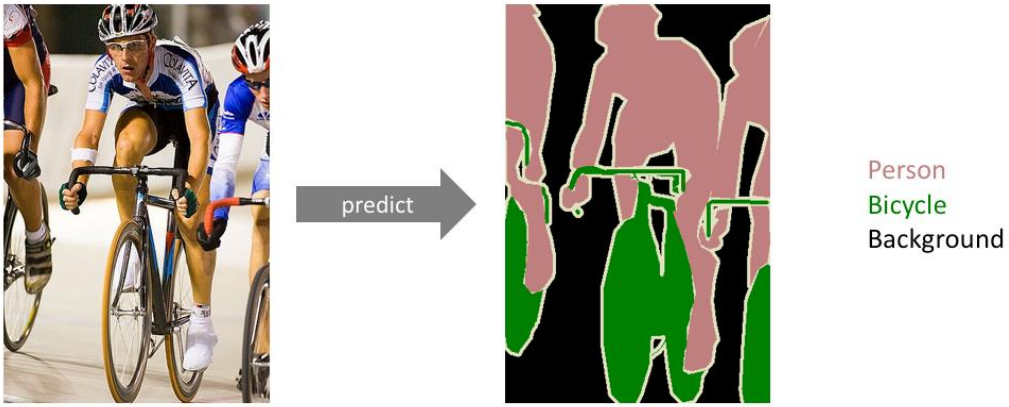


Figure 2-29 An prediction result of semantic segmentation [49]

i) Semantic Segmentation

A famous task of the CNN is semantic image segmentation, which is a computer vision task in which the image is labeled into specific regions according to the designed object. The semantic segmentation is based on the idea, "What is in this image, and where in the image is it located?" [49]. The goal is to label each pixel in an image with a corresponding *class* of the object that it is representing. Figure 2-29 shows an example of the result of the semantic segmentation task. The segmentation is used in various tasks such as autonomous vehicles, medical image diagnostics, or agriculture. To represent the segmentation task, the segmentation output is created as a segmentation map in which each pixel contains a class label represented by an integer. Figure 2-30 shows the relation between the original image and the segmentation result.

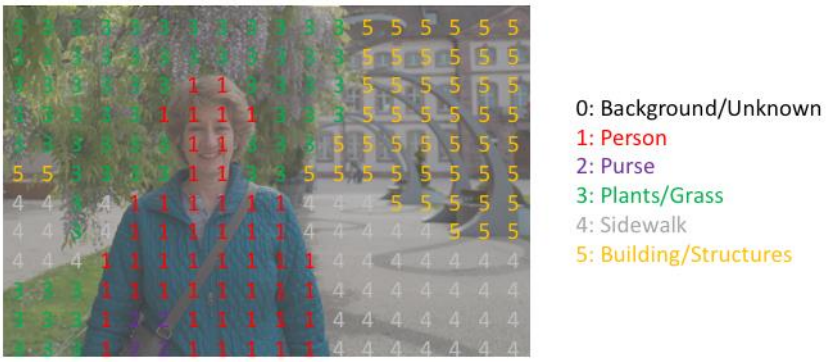


Figure 2-30 Semantic image segmentation [49]

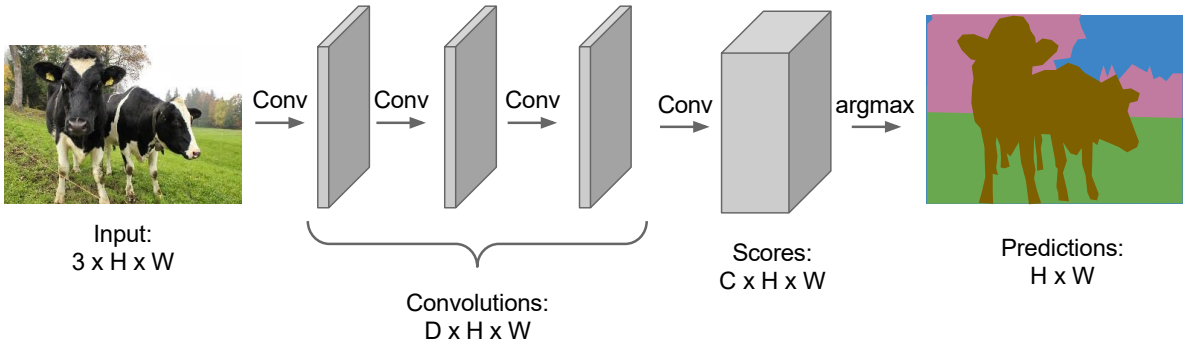


Figure 2-31 A network structure of semantic segmentation [50]

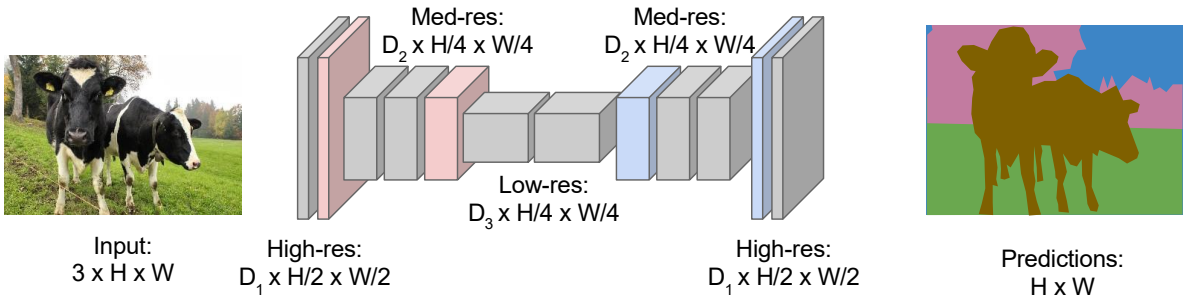


Figure 2-32 A network structure of encoder-decoder semantic segmentation [50]

An original method to create a neural network architecture of semantic segmentation is to rearrange several convolutional layers and an output map. Then, this created structure can be learned to map the corresponding result from the original input image. Figure 2-31 shows the semantic segmentation network architecture. However, to improve the efficiency of the segmentation, the number of feature maps has to increase to get deeper into the network. One of the most famous structures for image segmentation is an encoder-decoder model. It can learn to discriminate between classes efficiently and represent the feature map into a full-resolution segmentation map.

ii) Convolutional Layers

A convolutional layer is a filter that is a CNN's core operation and where main computations occur. The input of this layer typically is a three dimensions image consisting of height, width, and depth. The filter in this layer, also known as a kernel or a feature detector, will move across the input image to calculate the resulting product. This process is known as a convolution. The filter is a two-dimensional array, usually a 3×3 matrix, with weights that indicate each input image part. Lastly, the kernel slides by a stride repeatedly until it has swept across the whole target. The final output, the dot products from the input and the filter, is a feature map, also called an activation map or a convolved feature [51]. Figure 2-33 shows the working of the convolutional filter. In the figure, the filter contains various values called weight and can be updated by the training process. The filter calculation can be expressed as Eq. (2-24), where C, H, and W are the image size: channel, height, and width. N denotes the batch size or the number of the input image [52]. The symbol $*$ is the valid 2D cross-correlation operator.

$$\text{out}\left(N_i, C_{\text{out}_j}\right) = \text{bias}\left(C_{\text{out}_j}\right) + \sum_{k=0}^{C_{\text{in}}-1} \text{weight}\left(C_{\text{out}_j}, k\right) * \text{input}(N_i, k) \quad (2\text{-}24)$$

The size of the feature map or output array can be calculated by Eqs. (2-25) and (2-26). Where $\mathcal{P}$ denotes the padding number, $\mathcal{D}$ denotes the dilation number, $\mathcal{K}$ is a kernel size, $\mathcal{S}$ is a stride number. All four parameters will be described in detail as follow.

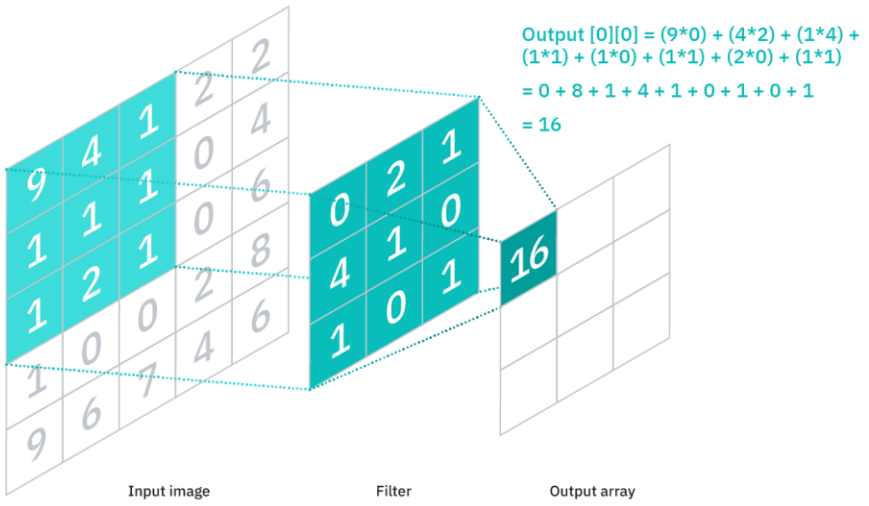


Figure 2-33 The calculation of convolutional filter [51]

$$H_{out} = \left\lfloor \frac{H_{in} + 2\mathcal{P}_y - \mathcal{D}_y(\mathcal{K}_y - 1) - 1}{\mathcal{S}_y} + 1 \right\rfloor \tag{2-25}$$

$$W_{out} = \left\lfloor \frac{W_{in} + 2\mathcal{P}_x - \mathcal{D}_x(\mathcal{K}_x - 1) - 1}{\mathcal{S}_x} + 1 \right\rfloor \tag{2-26}$$

Padding

Padding is an extra pixel added to the input image to estimate the loss problem at the edge of the image when applying a filter. The addition pixel typically is zero, called zero-padding. The number of padding represents the number of pixels that will be added to each side of the image. Figure 2-34 shows the image with padding.

Stride

Stride is a number of rows and columns traversed kernel per slide. The typical use of stride is one, which means shifting the filter to the next pixel. A large value of stride makes a small output size. Figure 2-35 shows the image with the stride of two and non-padding.

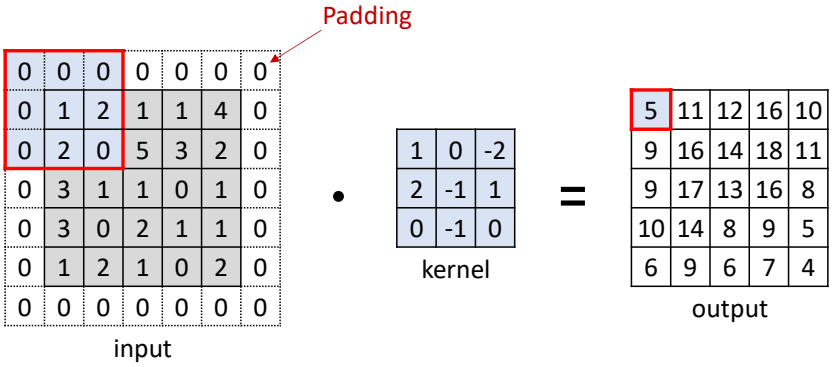


Figure 2-34 Two-dimensional cross-correlation with padding

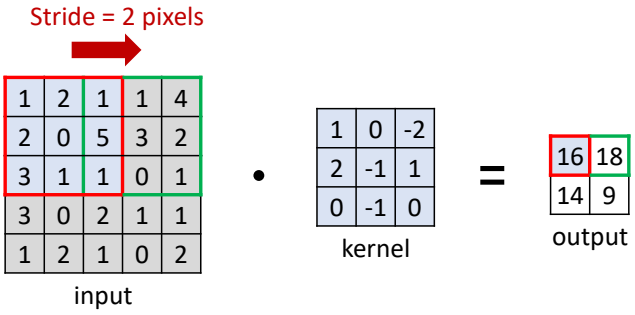


Figure 2-35 Two-dimensional cross-correlation with a stride of two

Dilation

Dilation is a space between the kernel elements. The default dilation value is one, which is no space between each kernel. A large value of dilation makes a small output size. Figure 2-36 shows the image with the dilation of two and non-padding.

iii) Pooling

A pooling operation is a window's fixed shape that slides over the image area. Unlike the convolutional layer, the pooling does not contain the weight and bias for calculation, no kernel. Instead, it determines only the value of the input image without an additional parameter. In general, there are two types of pooling: max-pooling and average-pooling. The max-pooling returns the maximum pixel value in the sliding window on the input image. In comparison, the average pooling calculates the mean of all values in the sliding window. Figure 2-37 shows both pooling operation outputs with a stride of two and a window size of three.

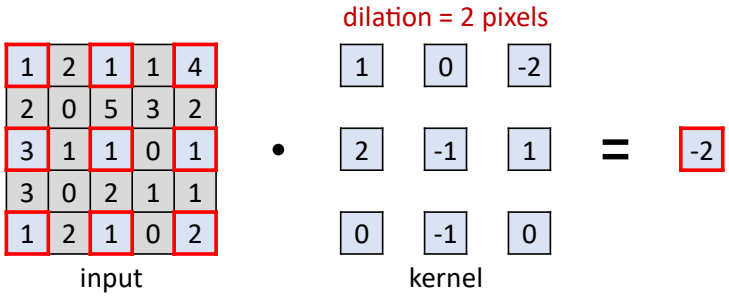


Figure 2-36 Two-dimensional cross-correlation with dilation of two

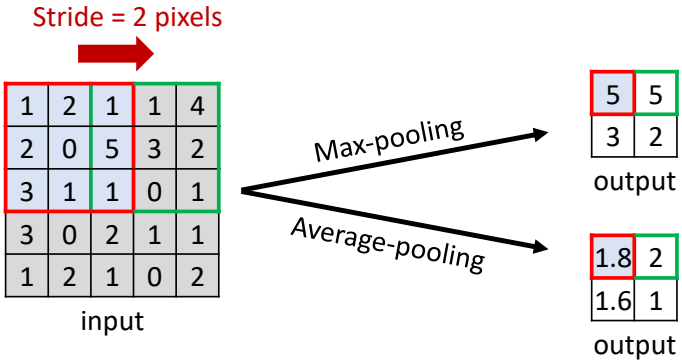


Figure 2-37 Comparison of each pooling

iv) Batch Normalization

During the network training, the change in the distributions of internal nodes makes the training slow. To eliminate this problem, a technique called batch normalization is introduced [53]. It dramatically accelerates the training of the neural network. In addition, it also affects the gradient flow by reducing the gradient dependence of the initial parameters. This operation reduces the necessity of the dropout [54].

The application of batch normalization on a 4D input, a mini-batch of 2D inputs with additional channel dimension, can be calculated by Eq. (2-27). The batch normalization normalizes the elements x of the input by first calculating the mean $\mathrm{E}[x]$ and variance $\mathrm{Var}[X]$ over the spatial, time, and observation dimensions for each channel independently. ϵ is a constant that improves numerical stability when the variance is very small. γ and β are learnable parameter vectors of the input size.

$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[X] + \epsilon}} * \gamma + \beta \tag{2-27}$$

v) Activation Function

An activation function is a controller of the output exporting from a node to node in the neural network. It is used to transfer, limit, and squash an output before sending it out. This function plays a significant impact on the performance of the neural network. Therefore, it is usually used as the last internal operation of each node in the network [55]. There are several types of activation functions, but three of them are used in this experiment. Figure 2-38 shows the relationship between the input and output of each activation function.

Rectified Linear Unit

Rectified Linear Unit (ReLU), also called rectifier, is the most famous activation function used in the neural network. In addition, it finds applications in computer vision and speech recognition [56]. Equation (2-28) explains the calculation of ReLU [57].

$$\mathrm{ReLU}(x) = (x)^{+} = \max(0, x) \tag{2-28}$$

Sigmoid

Sigmoid is a mathematical function with an S-curve characteristic, the sigmoid curve. It is defined for all real input values and has a non-negative derivative at each point [58]. Equation (2-29) explains the calculation of sigmoid function [59].

$$\text{Sigmoid}(x) = \sigma(x) = \frac{1}{1 + e^{-x}} \tag{2-29}$$

Softplus

Softplus, also called SmoothReLU, is a smooth approximation to the ReLU. It is related to the sigmoid function when the value is near the negative infinity [60]. Equation (2-30) explains the calculation of the softplus function[61].

$$\text{Softplue}(x) = \frac{1}{\beta}\log\left(1 + e^{\beta x}\right) \tag{2-30}$$

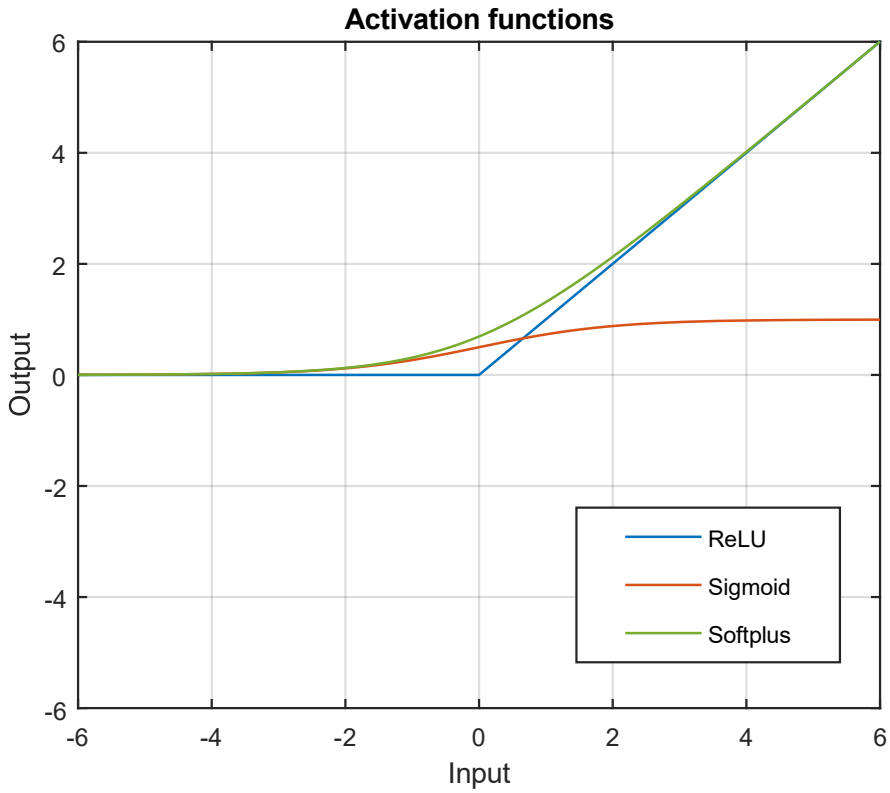


Figure 2-38 Relationship between input and output of activation functions

vi) Dropout

The dropout is a mask that eliminates the contribution of some neurons towards the next layer and leaves all others unmodified. It is used to prevent the overfitting of the training data. The application of dropout can prevent the first batch training from a disproportionately high manner. In addition, it also protects the feature learning that appears only in the later batch.

The dropout can be operated by randomly zeros some elements of the input tensor. It is an efficient technique to regularize and prevent the co-adaptation of the neuron [62]. Figure 2-39 shows the comparison diagram of using dropout. The dropout process inactivates some nodes in the network. Figure 2-40 shows example results of using dropout with max-pooling.

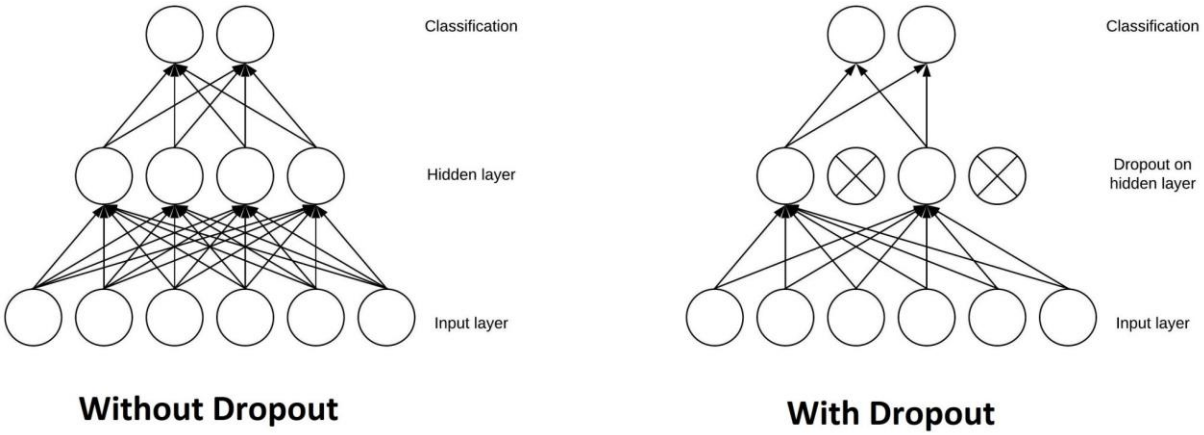


Figure 2-39 Comparison diagram of applying dropout layer [63]

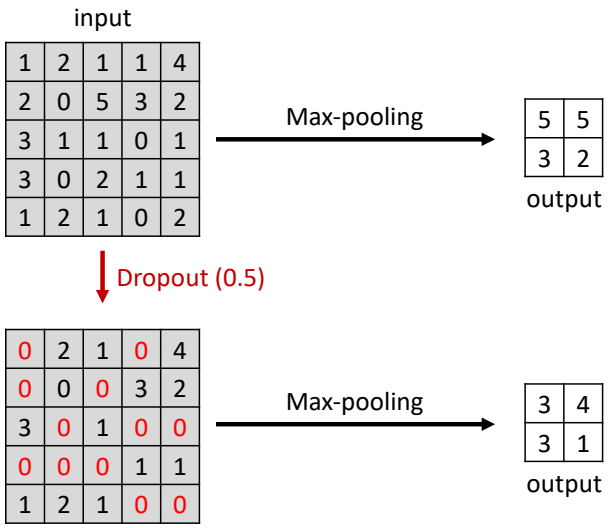


Figure 2-40 Example of dropout 50% before max-pooling

vii) Loss Function

The loss function is the function that determines the difference between the output of the prediction and the reference output. It is used for evaluating model algorithms. In addition, the result of the function is used as feedback to improve the working of the algorithm [64].

Cross-entropy loss, or log loss, has the output of a probability value from 0 to 1. It is the famous loss function used to measure the performance of a classification model. The loss value increases when the predicted output differs from the expected output. The perfect model would have a cross-entropy loss of zero. In segmentation, the average cross-entropy loss can be calculated by averaging each binary cross-entropy loss, as shown by Eq. (2-31).

$$Loss = -\frac{1}{m}\sum_{m} q \cdot \mathbf{log}(p) + (1-q)\mathbf{log}\,(1-p) \tag{2-31}$$

Where m is the number of classes, p is the predicted probability observation, and q is the binary indicator (0 or 1). Figure 2-41 shows the relationship curve between the predicted image and the reference labeled. Moreover, this dissertation introduces the method with multiple outputs, called side output, described in Section 3.2-c and Section 4.2-d. Therefore, the loss function is modified to cover every output. The losses of all outputs are summed before differentiating the losses as Eq. (2-32), where i is the number of outputs.

$$Losses = \sum_{i} loss_i \tag{2-32}$$

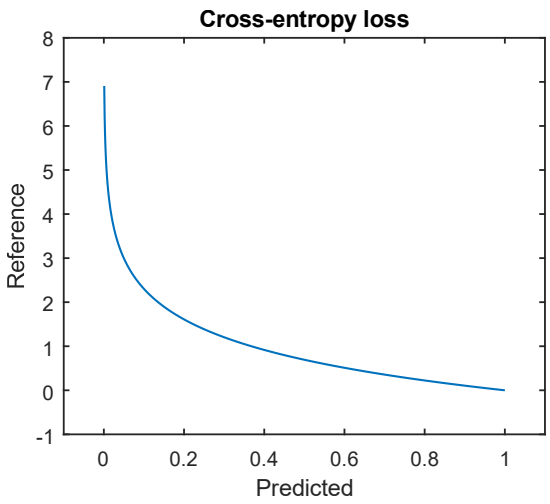


Figure 2-41Cross-entropy loss

viii) Optimization

An optimization algorithm is a method to update the model parameters and minimize the value of the loss function. In this dissertation, Stochastic Gradient Descent (SGD) is selected as the optimizer. It is an iterative method to optimize an objective function with suitable smoothness properties. The SGD is an important method in machine learning due to its basic idea [65]. Table 2-3 shows the algorithm of SGD. Where a $params$ is iterable of parameters to optimize parameter groups, lr is learning rate, $momentum$is momentum factor, $dampening$ is dampening for momentum $nesterov$ is enabling value of Nesterov momentum [66].

Table 2-3 SGD algorithm [67]

input : $\gamma\ (lr), \theta_0(params), f(\theta)(objective), \lambda\ (weight\ decay),$
$\mu\ \ (momentum), \tau\ (dampening), nesterov$

for $t = 1$ **to … do**
  $g_t \leftarrow \nabla_\theta f_t(\theta_t - 1)$
  if $\lambda \neq 0$
    $g_t \leftarrow g_t + \lambda\theta_{t-1}$
  if $\mu \neq 0$
    if $t > 1$
      $\boldsymbol{b}_t \leftarrow \mu\boldsymbol{b}_{t-1} + (1-\tau)g_t$
    else
      $\boldsymbol{b}_t \leftarrow g_t$
    if $nesterov$
      $g_t \leftarrow g_{t-1} + \mu\boldsymbol{b}_t$
    else
      $g_t \leftarrow \boldsymbol{b}_t$
  $\theta_t \leftarrow \theta_{t-1} - \gamma g_t$

return θ_t

h. Robot Operating System

Robot Operating System (ROS) is an open-source robotics middleware suite. It is not an operating system, but it contains many software frameworks for robot software development. The software in ROS can be separated into three sets. First is language-and platform-independent tools used to build and distribute ROS-based software. The other is ROS client library implementations such as roscpp, rospy, and roslisp. Another is packages containing application-related code that uses one or more ROS client libraries. The goal of ROS is to support code reuse in robotics research and development [68].

Various tools augment the ROS core to help developers visualize, record, create scripts, and set up processes. For examples of ROS tools, the rviz is used to visualize the robot environment and the sensor data. In addition, the rosbag is used to record and playback the ROS message data. Lastly, roslanch is a tool for launching multiple ROS nodes locally and remotely.

The ROS communication packages include core client libraries, roscpp and rospy, and the implementation of concepts such as topics, nodes, parameters, and services. Figure 2-42 shows the structure of the ROS.

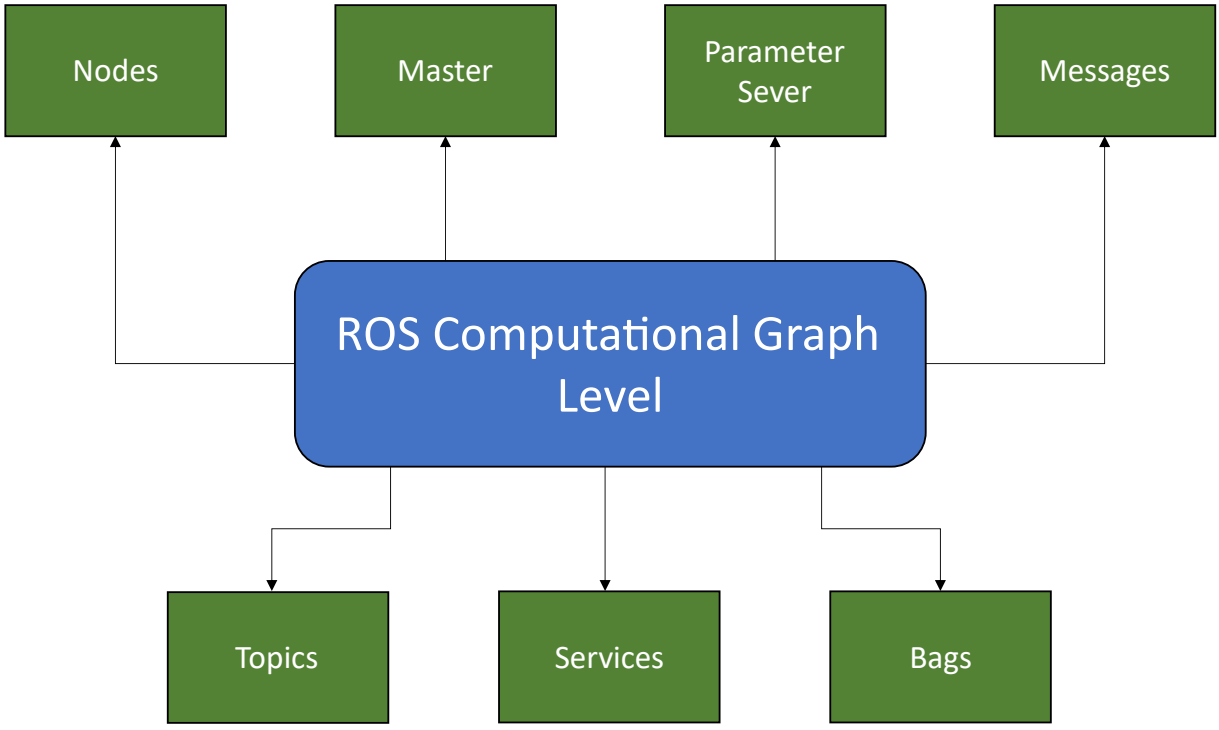


Figure 2-42 Structure of the ROS Graph layer [69]

i. Evaluation

There are various evaluation methods to determine the approach efficiency in the object detection or segmentation problem. Most of the techniques are based on the matching area of the prediction. Figure 2-43 shows the related areas in the segmentation problem. The TP denotes true-positive, which is the right predicted area. The TN denotes true-negative, which is the right unpredicted area. The FP denotes false-negative, which is the wrong predicted area. The FN denotes false-negative, which is the wrong unpredicted area. For example, in human detection, TP is the human area that is predicted as human. TN is the non-human area that is predicted as non-human. FP is the non-human area that is predicted as human. Lastly, FN is the human area that is predicted as non-human. Two techniques are used to evaluate the network efficiencies in this dissertation.

i) Intersection over Union

Intersection over Union (IoU) is a technique used to describe the area of overlap between two regions. It is famously used in the application of object detection and semantic segmentation. The IoU can be calculated by the area of the intersection divided by the area of union. Figure 2-44 shows the diagram of the IoU calculation. The IoU calculation can also be described by Eq. (2-33).

$$IoU = \frac{TP}{TP + FP + FN} \tag{2-33}$$

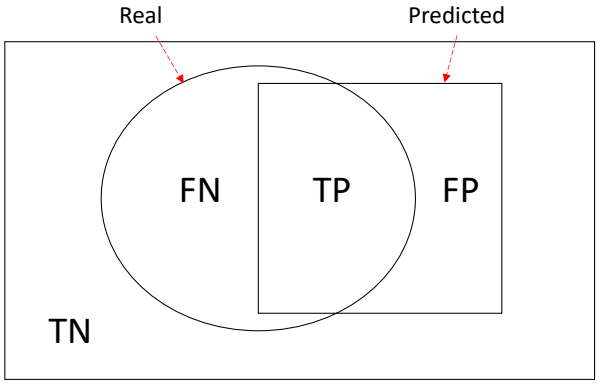


Figure 2-43 Related regions of the prediction problem

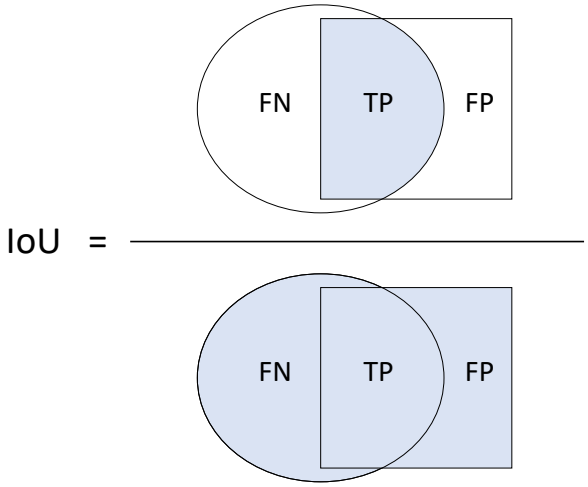


Figure 2-44 IoU calculation

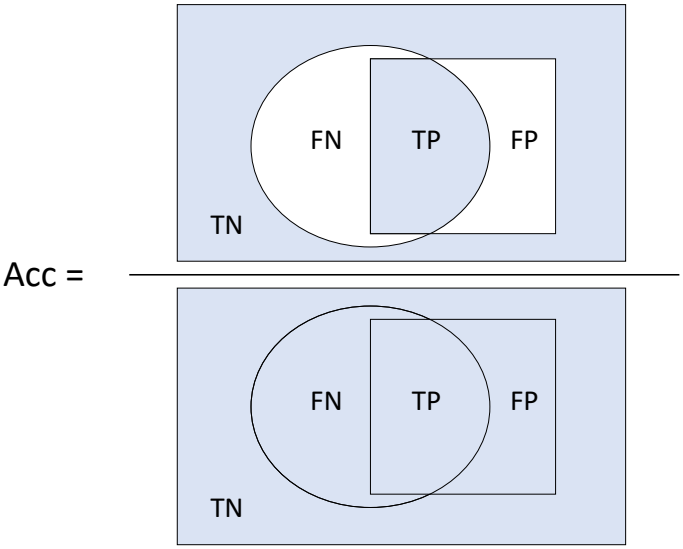


Figure 2-45 Acc calculation

ii) Accuracy

Accuracy (Acc) is an overall evaluation of the segmentation problem. It includes all correct predicted regions in the calculation. Therefore, the Acc can be used to represent the correct predicted rate. However, it may make a misleading result due to the imbalanced dataset. Equation (2-34) explains the calculation of the Acc based on the related area. Figure 2-45 shows the diagram of the Acc calculation.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{2-34}$$

2.2 Literature Review

a. Perception of Road Environments

The research on the perception of road environments has been rapidly progressing. Previous work, such as Jokela et al. [70], introduced a method to identify road surface types, such as snowy, icy, and wet, here by using a graininess analysis of acquired images with stereo image pairs. Another road status classification research, Troiano et al. [71], focused on sensors installed under the road surface, while some researchers such as Omer et al. [72], Jonsson et al. [73], and Kawai et al. [74] have used a support vector machine (SVM) and the K-nearest neighbor (KNN) to determine road conditions. For road region detection, John et al. [75] and Kong et al. [75] recognized drivable areas of the road using a classical image processing technique based on vanishing point detection.

b. Semantic Segmentation

Semantic segmentation is a neural network that identifies every pixel in the image and classifies it into different classes. Moreover, various kinds of road surfaces support the neural network's working potential. Deep neural networks can solve some deficiencies of this method because it is a preferred method over other alternatives. Consequently, semantic segmentation was used based on various methods. A network called Fully Convolutional Networks (FCN) [50] was able to segment the image from any size by use of convolution neural networks (CNN) without fully connected layers making it a standard approach for new techniques. Figure 2-46 shows FCN's network structure.

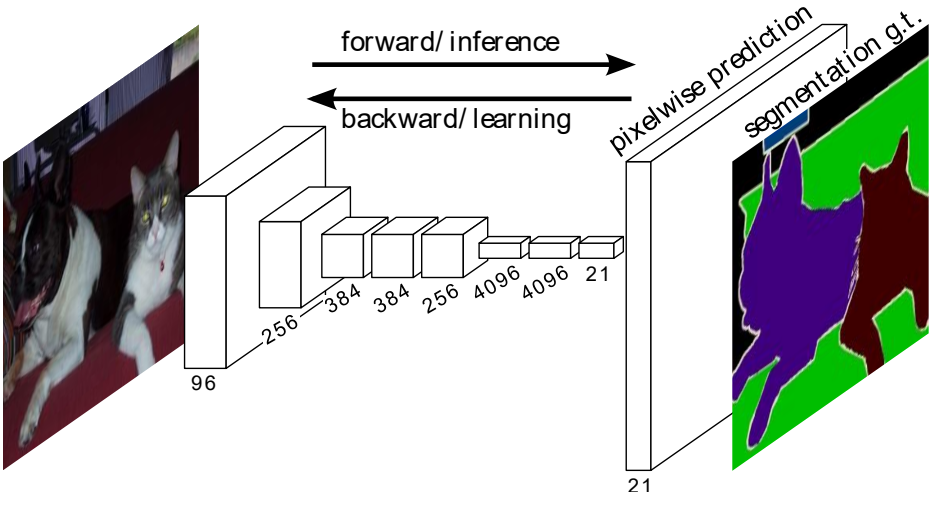


Figure 2-46 FCN network structure [50]

For instance, research [76] applied FCN to detect objects on a snowy road environment with the system accuracy, showing a mean intersection over union (mIoU) of about 50% with 7.84 FPS (frame per second). The FCN-based network name DSNet [77], which maintained an accuracy from most previous ones, got 69.1% mIoU on the Cityscape dataset and 72.6% on the CamVid dataset. A real-time semantic segmentation [78] named ICNet used cascade feature fusion units to obtain segmentation, resulting in the Cityscape dataset was 69.5% mIoU with 30.3 FPS. Figure 2-47 shows the ICNet network structure. An encoder-decoder-based developed network called Data-dependent Upsampling (DUpsampling) [79] was proposed to replace bilinear, which can recover the pixel-wise prediction from low-resolution outputs of CNNs. This model's main points were the improvement of reconstruction capability and flexibility of the decoder in leveraging almost arbitrary combinations. Figure 2-48 shows the DUpsampling network structure.

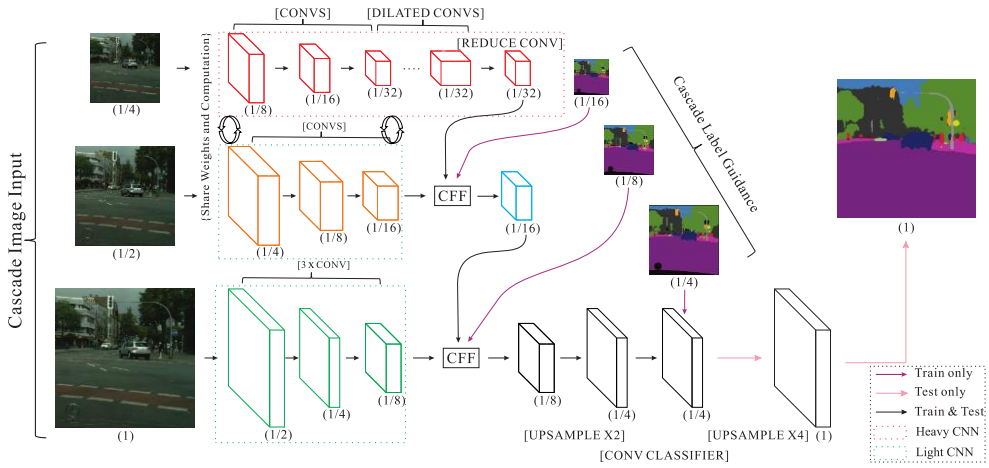


Figure 2-47 ICNet network structure [78]

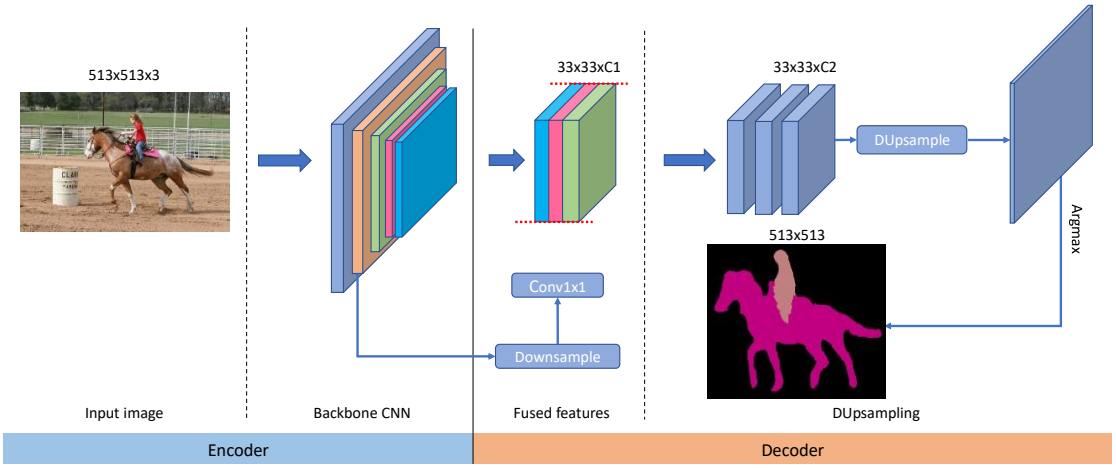


Figure 2-48 DUpsampling network structure [79]

Work on a slippery road caused by water, ice, and snow named D-UNet [80], developed from U-Net [81], used dilated convolutions for the sensible field of network. This technique got the highest performance as compared to classical machine learning. One of the high-performance networks called Pyramid Scene Parsing Network (PSPNet) [82] was used for scene parsing of semantic segmentation. They applied the Pyramid Pooling Module, which consisted of four different pyramid scales to fuse the feature map, and the efficiency on the Cityscape dataset was about 82.6% mIoU. Figure 2-49 shows the PSPNet network structure. Gao et al. [83] developed an end-to-end framework from PSPNet called Multiple Feature Pyramid Network (MFPN) to function with road detection from satellite view, which further introduces a Tailored Pyramid Pooling Module to advance the accuracy of the original network. This model reached up to 7.8% mIoU higher than PSPNet with Massachusetts dataset. Zhao et al. [84] proposed the Pointwise Spatial Attention Network (PSANet) to relieve the local neighborhood's constraints. The training with fine data and coarse+fine data were tested on the Cityscape dataset and got about 80.1% and 81.4% mIoU in sequence. Figure 2-50 shows the PSA module structure.

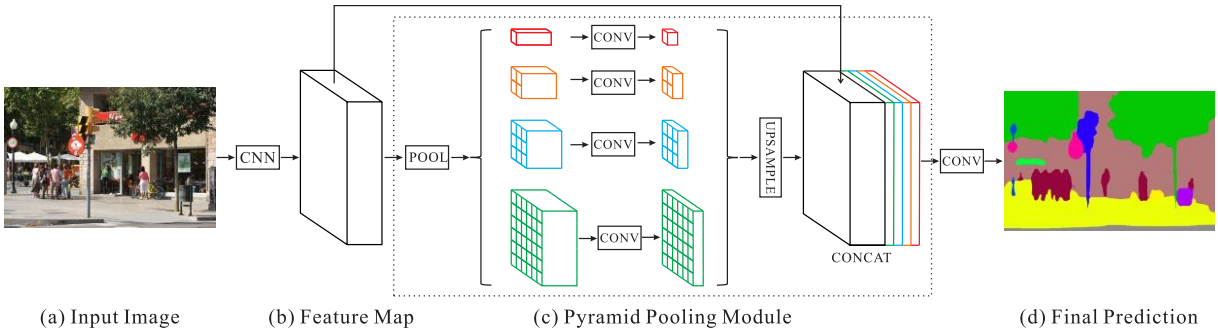


Figure 2-49 PSPNet network structure [82]

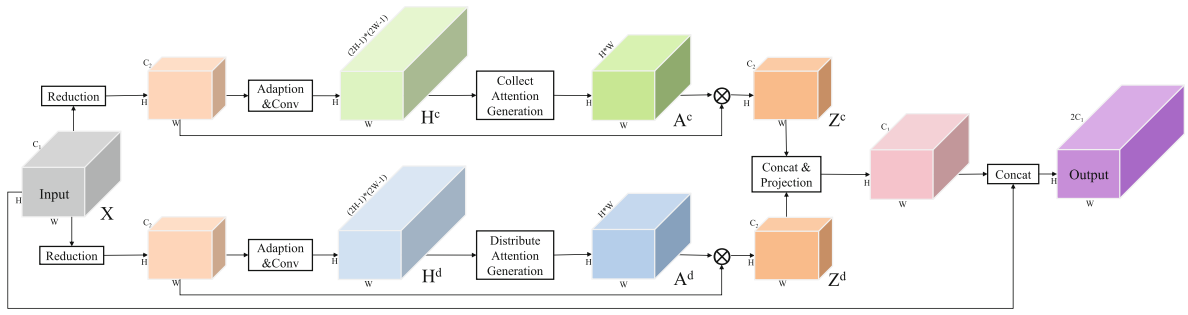


Figure 2-50 PSA module structure in PSANet [84]

A network name Dual Attention Network (DANet) [85] was developed to integrate local features with global dependencies. This model included two types of attention modules: the position attention module and the channel attention module. The position attention module combines each position feature by a weighted sum at all positions. The channel attention module emphasizes channel maps by adding features of all channel maps with a mIoU of 81.5% on the Cityscape dataset. Figure 2-51 shows the DANet network structure. Another significant research by Google Inc. [86] introduced a high-efficiency network called DeepLabv3, which applied Atrous Convolutions in parallel to extract the multi-scale context with different atrous rates. Figure 2-52 shows the comparison of multi-scale capturing. In addition, the atrous convolution with rate was processed in parallel, called Atrous Spatial Pyramid Pooling (ASPP). This module consisted of a 1×1 convolution filter, three 3×3 convolution filters with various atrous rates and a pooling feature. The efficiency on the Cityscape dataset was 81.3% mIoU. Figure 2-53 shows the feature extraction structure of DeepLabv3 network. Our previous work [3] tried to improve the training process for segmentation in a snowy environment. The network is based on DeepLabv3 with extracted auxiliary outputs to enhance the training performance.

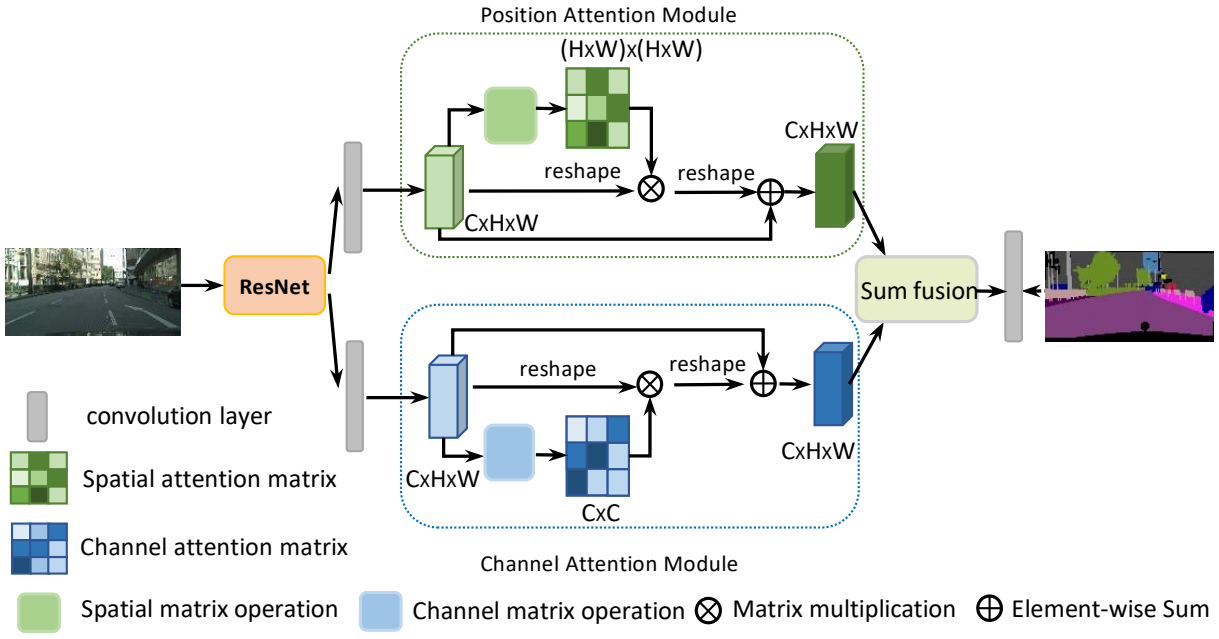


Figure 2-51 DANet network structure [85]

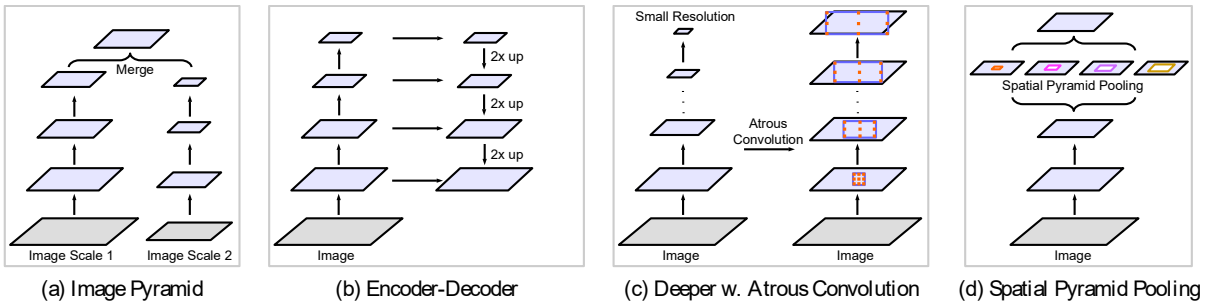


Figure 2-52 Comparison of multi-scape capturing [86]

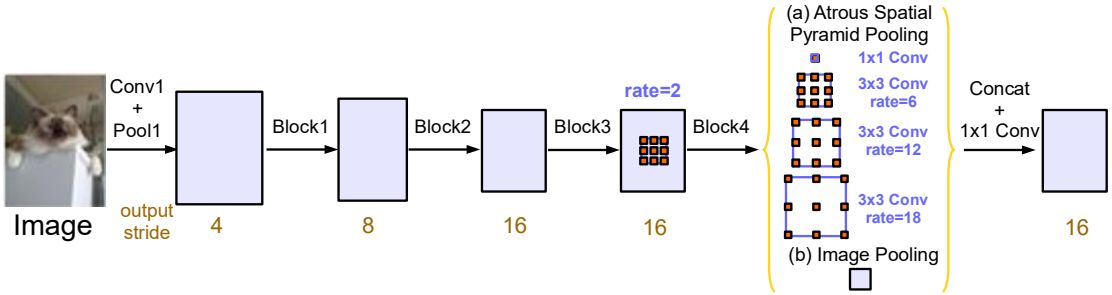


Figure 2-53 Feature extraction in DeepLabv3 network [86]

c. Multiple Input Semantic Segmentation

A semantic segmentation technique can accommodate multiple inputs to enhance the segmentation's accuracy. For example, RedNet [87] improved the training procedure by fixing the gradient vanishing problems; this method was based on an encoder–decoder network operated with dual inputs, RGB, and depth images. It was able to calculate four advanced outputs called the pyramid supervision training scheme in the decoder part. The encoder part of this network was based on the ResNet-50 model [88], with the decoder part operated by using the reversed ResNet-34 and ResNet-50 models, making this scheme achieve the best results on the SUN RGB-D dataset. Figure 2-54 shows the RedNet model structure. Research on modal fusion for indoor environments [89] has proposed a model based on SIFT features and MRFs. Other works, such as Gupta et al. [90], have combined RGB and depth features using CNNs. This approach classifies pixels in the detection window as the foreground or background. The UpNet network [91] uses multispectral and multi-modal images for segmentation. The network contains fusion architecture that merges RGB, near-infrared channels, and depth information. Valada et al. [92] introduced a network architecture consisting of two modality-specific encoder streams and a self-supervised model adaptation fusion module. Here, tests on the Cityscape, Synthia, and SUN RGB-D datasets achieved state-of-the-art performance compared with other

networks. Hazirbas et al. [93] also combined RGB with depth data for the SUN RGB-D dataset's indoor environment. The network architecture is an encoder–decoder type. The two encoder branches were used to extract the feature maps from both the RGB and depth data in parallel. Also, Wang and Neumann [94] introduced their network called depth-aware CNN. The two operations–depth-aware convolution and depth-aware average pooling were integrated into CNNs for segmentation in the RGB-D dataset. Not only was depth information used to improve the RGB segmentation, but the thermal information was also considered too. A work on an RGB-T dataset [95] introduced the use of RGB image and IR image segmentation for autonomous vehicles; the network was an encoder–decoder that extracted the features from two encoders in parallel and fused them in the decoder part. This network reached a higher accuracy than the other state-of-the-art segmentation methods. The RGB-thermal fusion network, or RTFNet [96], was developed for segmentation in the urban scenario. Different from FuseNet [93] and MFNet [96], this network utilized ResNet [88] as the feature extractor. The RGB feature and thermal feature were fused in the RGB encoder part by element-wise addition. The Upception block has also been proposed as an initial part of the decoder. The works mentioned above describe multi-input networks that are accurate in good environment conditions, but they come up short in snowy scenarios.

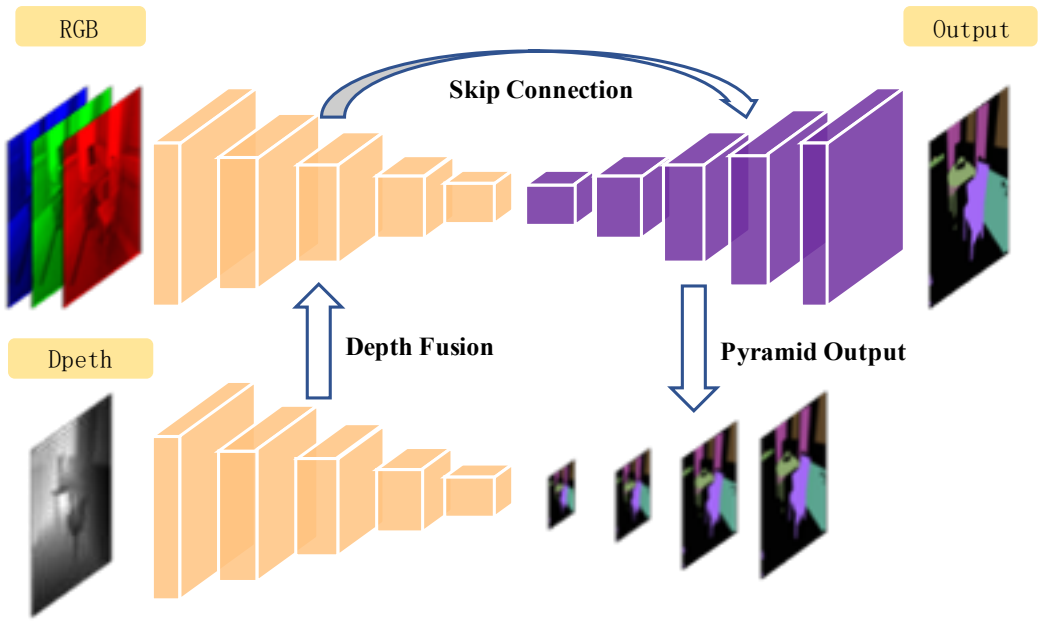


Figure 2-54 RedNet network structure [87]

CHAPTER 3

SINGLE INPUT EXPERIMENT

3. SINGLE INPUT EXPERIMENT

This experiment was based on recognizing the snowy road environment by using only one information as an input. This was the most uncomplicated experiment because it did not request much initial information to classify and used a low-cost sensor to obtain the dataset. The dataset in this experiment was only an RGB image. Therefore, this section did not include the sensors combination and 3D visualization. The objective of this experiment was to classify all pixels in the image into each different objects.

3.1 Proposal of Algorithm

This experiment was set up to introduce the most robust network structure for semantics segmentation in snowy environments by using only an RGB image from the RGB camera.

3.2 Datasets

There were two datasets used in the single input experiment. One was a self-made dataset that included the snowy environment of Hokkaido prefecture in Japan. Another was the published dataset from the Mapillary Vistas dataset, which contained many images worldwide.

a. Snowy road in Hokkaido dataset

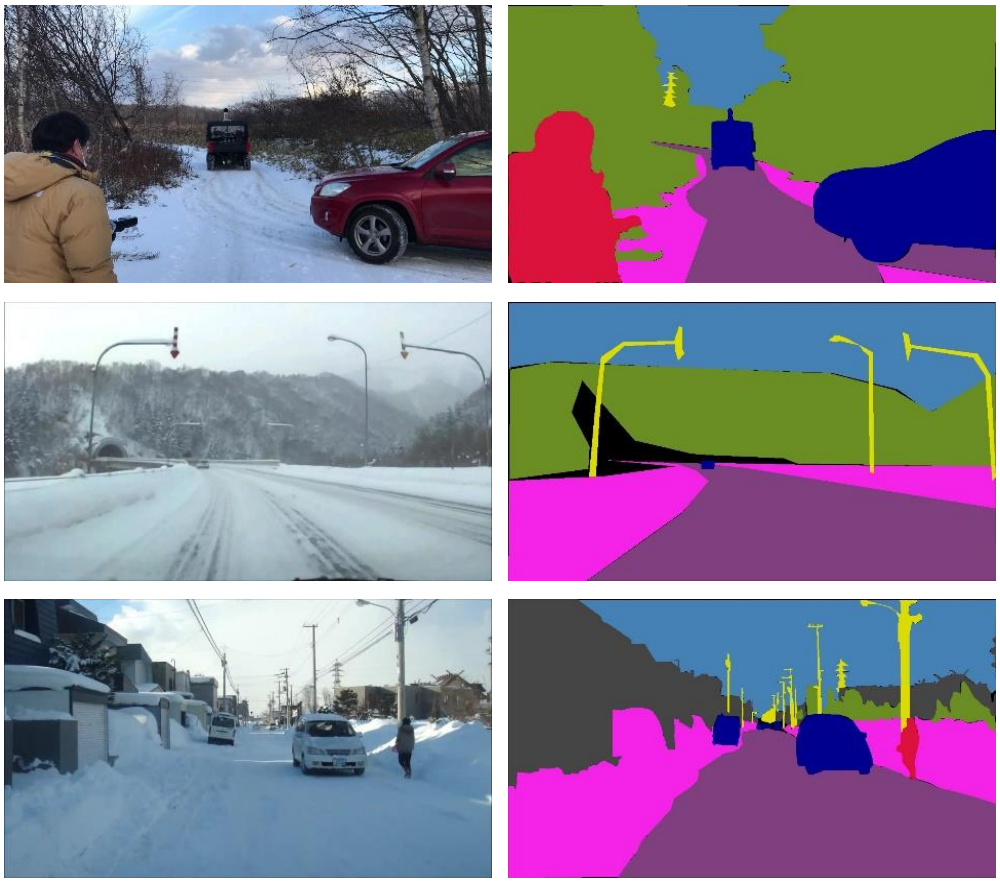
This dataset consisted of the snowy road scenario in various places of Hokkaido during the winter of 2018, such as towns, highways, forests, and the countryside. This dataset was only the RGB image dataset collected using many kinds of cameras: RGB camera, webcam, and dash camera. The camera was installed on the front windshield of the car and then drove along the road in the experimental area. The dataset included 1446 images with different resolutions, as described in Table 3-1. As described in Table 3-2, every image was labeled into nine classes in this dataset. Figure 3-1 shows the example images of this dataset, and Figure 3-2 shows an example of each class. Dataset generating was supported by Witz & Co., Ltd.

Table 3-1 Image resolution (pixel) of Snowy road dataset

Height	Width	Images	Height	Width	Images	Height	Width	Images
267	400	1	442	660	1	682	1024	1
281	450	1	444	710	1	683	1024	1
281	500	1	450	600	2	720	1280	725
298	448	1	452	680	1	733	1100	1
300	400	3	462	690	1	738	1280	73
343	513	1	480	640	1	832	1248	1
344	612	1	484	728	3	900	1200	1
360	480	1	485	728	1	960	1280	1
375	500	1	487	650	1	1000	1500	1
378	721	1	520	780	1	1080	1920	579
380	680	1	525	700	1	1090	1920	14
387	580	1	533	700	1	1118	2300	1
399	600	1	533	800	1	1188	1918	1
423	500	1	600	800	2	1200	1600	2
426	640	1	640	480	1	1214	2300	1
427	640	1	669	1000	1	3024	4032	5

Table 3-2 Snowy road dataset

ID	Name	Pixels (pixel)	Pixels (%)	Color
1	Road	3.28E+08	15.9	
2	Sidewalk	1.77E+08	8.6	
3	Building	6.99E+07	3.4	
4	Traffic object	1.67E+07	0.8	
5	Vehicle	7.63E+08	37.1	
6	Human	3.82E+08	18.6	
7	Sky	3.97E+06	0.2	
8	Vegetation	3.65E+07	1.8	
9	Unidentified	2.81E+08	13.7	



(a) (b)

Figure 3-1 Examples of snowy road environment; (a) RGB images, (b) labeled images

Figure 3-2 Example of each class

b. Mapillary Vistas dataset

Mapillary [97] published the Mapillary Vistas dataset in 2017. It consisted of more than 25,000 high-resolution images from around the world. The dataset had various situations which were labeled into 124 classes. The whole dataset was selected only snowy environments to use in this experiment. The dataset used in this experiment included 650 images with different resolutions, as described in Table 3-3. As described in Table 3-4, every image was labeled into 13 classes in this dataset. Figure 3-3 shows the example images of this dataset.

Table 3-3 Image resolution (pixel) of Mapillary Vistas dataset

Height	Width	Images	Height	Width	Images	Height	Width	Images
1080	1920	5	2160	3840	17	2988	5312	4
1341	2012	20	2250	3000	1	3000	4000	2
1342	2013	14	2341	3121	3	3024	4032	28
1520	2048	1	2448	3264	428	3025	4033	5
1536	2048	12	2449	3265	20	3072	4096	1
1728	3072	2	2592	4608	14	3096	4128	14
1836	3264	10	2640	3520	2	3456	4608	6
1920	2560	4	2760	3680	8	3752	5376	4
1936	2592	5	2976	3968	1	3936	5248	5
1944	2592	1	2988	3984	12	4240	5664	1

Table 3-4 Mapillary Vistas dataset

ID	Name	Pixels (pixel)	Pixels(%)	Color
1	Road	9.34E+08	17.2	
2	Sidewalk	8.50E+07	1.6	
3	Building	7.64E+08	14.1	
4	Lane marking	8.33E+07	1.5	
5	Traffic object	1.22E+08	2.3	
6	City object	1.01E+07	0.2	
7	Vehicle	1.86E+08	3.4	
8	Human	1.08E+07	0.2	
9	Snow	1.66E+09	30.6	
10	Sky	8.28E+08	15.3	
11	Vegetation	2.97E+07	0.5	
12	Landscape	4.87E+08	9.0	
13	Unidentified	2.21E+08	4.1	

(a) (b)

Figure 3-3 Examples of Mapillary Vistas dataset; (a) RGB images, (b) labeled images

3.3 Network Architecture

This part proposes a developed neural network for segmentation in the snowy environment. The network was adapted from a famous and influential network structure named DeepLabv3. In this network, a structure called ResNet50 was used as a network backbone for feature extraction. The Atrous Spatial Pyramid Pooling layers (ASPP) of the DeepLabv3 were applied to capture multi-scale information for the segmentation head module. Short decoder layers substituted the output map-generating layers. This network also combined the pyramid supervision of RedNet to enhance the training process. Figure 3-4 shows the overall structure. Each part of the network will be described as follow.

a. Encoder

The encoder of this network was the ResNet-50 structure. The purpose of the encoder was to increase the number of feature maps from 3 to 2048 channels. It begins with the downsample operation, downsample 0, of a sequence of a 7×7 convolution layer sequence with stride two padding three, normalization, Rectified Linear Unit, and 3×3 max-pooling layer with stride two paddings one. This first layer reduced feature size to 1/4 of the original image and increased the channel to 128 channels. The following four layers of this encoder were processes of bottlenecks of convolution filters. Bottleneck no.1, named downsample 1, generated 256 channels of feature map but retained the feature map size at 1/4 of the original image. Bottleneck no.2, named downsample 2, generated 512 channels of feature map and reduced the feature map size to 1/16. Finally, bottlenecks no.3 and 4, named downsample 3 and 4, generated 1024 and 2048 channels of feature map without size reduction. Figure 3-5 shows an example of a bottleneck diagram. The information of each layer is explained in Table 3-5, where 7×7, 64 represents 64 convolution filters of 7×7 filter size. The original image was converted to the feature map with the shape of 2048×60×80 by using this encoder. The left five blocks in Figure 3-4 show the feature maps during each step in the encoder.

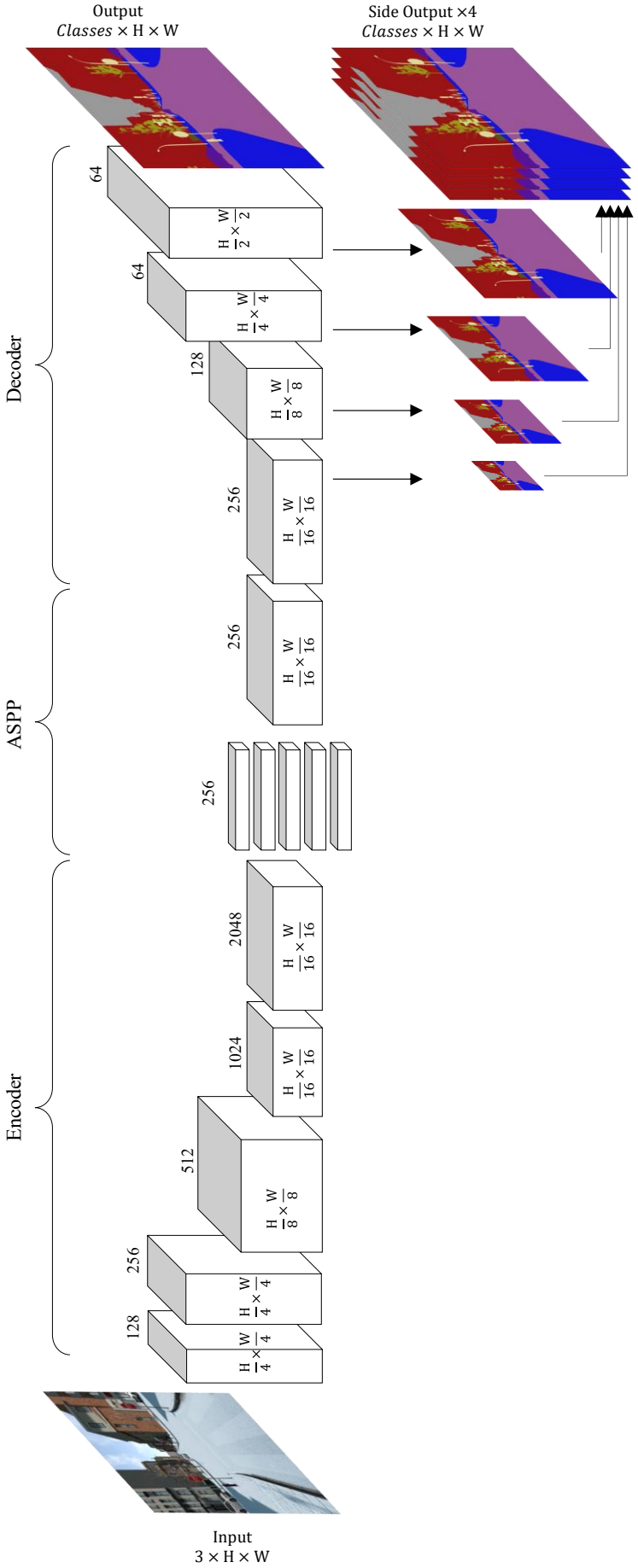


Figure 3-4 Overall network diagram

Table 3-5 Detail of each layer in encoder

Layer name	Output size	Convolution filter
input	$H \times W$	
downsample 0	$\frac{H}{4} \times \frac{W}{4}$	$7 \times 7, 64$
downsample 1	$\frac{H}{4} \times \frac{W}{4}$	$\begin{bmatrix} 1 \times 1, & 64 \\ 3 \times 3, & 64 \\ 1 \times 1, & 256 \end{bmatrix} \times 3$
downsample 2	$\frac{H}{16} \times \frac{W}{16}$	$\begin{bmatrix} 1 \times 1, & 128 \\ 3 \times 3, & 128 \\ 1 \times 1, & 512 \end{bmatrix} \times 4$
downsample 3	$\frac{H}{16} \times \frac{W}{16}$	$\begin{bmatrix} 1 \times 1, & 256 \\ 3 \times 3, & 256 \\ 1 \times 1, & 1024 \end{bmatrix} \times 6$
downsample 4	$\frac{H}{16} \times \frac{W}{16}$	$\begin{bmatrix} 1 \times 1, & 623 \\ 3 \times 3, & 512 \\ 1 \times 1, & 2048 \end{bmatrix} \times 3$

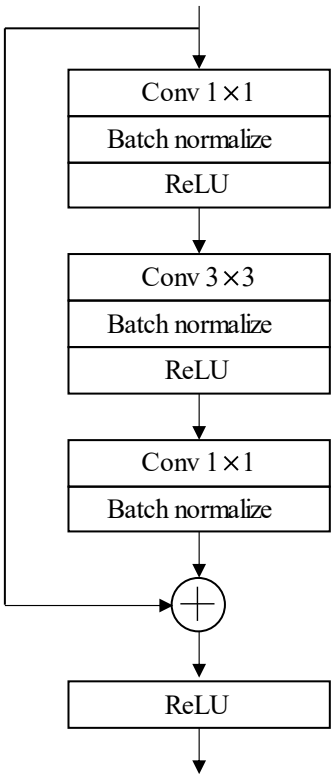


Figure 3-5 Bottleneck block diagram

b. Atrous Spatial Pyramid Pooling (ASPP)

The Atrous Spatial Pyramid Pooling or ASPP module, introduced in the DeepLabv3 network, was utilized to capture long-range context and multi-scale information. The module consisted of 5 parallel branches of 4 convolution branches (one 1×1-kernel, three 3×3-kernels) and one pooling branch. Figure 3-6 shows the ASPP module. Each 3×3-kernel convolution branch included convolution with rates = {12, 24, 36}, batch normalization, and rectified linear unit (ReLU). The pooling branch includes adaptive average pooling, 1×1 convolution branch, and interpolation. Each branch reduced the feature map channels from 2048 to 256 channels. All five outputs were combined with the concatenation operator to obtain a feature map with 1280 channels. Then, it was fed into another 1×1 convolution branch with a 0.5 value of dropout. The final output had the same size as the module's input, but the channels were decreased to 256 channels. The information of each branch is explained in Table 3-6.

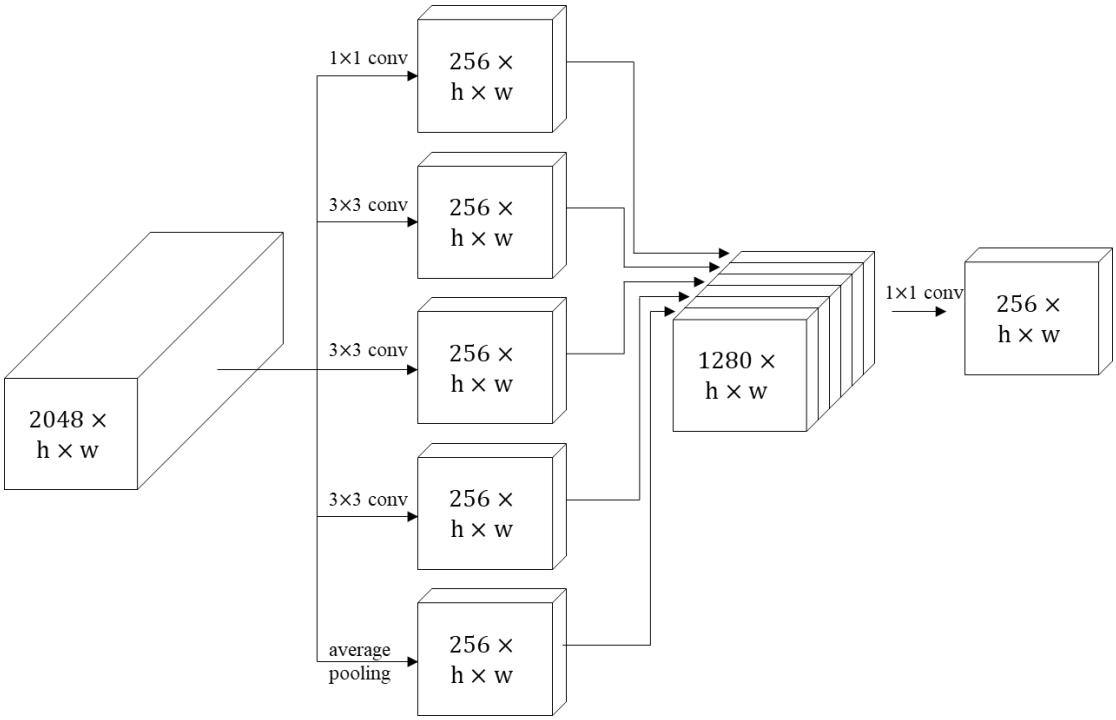


Figure 3-6 Feature map shape in ASPP

Table 3-6 Detail of each branch in ASPP

Branch name	Average pooling	Convolution filter	Normalizer and ReLU	Interpolation	Dropout
conv1	-	1×1	✓	-	-
conv3_1	-	$3 \times 3, \text{rate} = 12$	✓	-	-
conv3_2	-	$3 \times 3, \text{rate} = 24$	✓	-	-
conv3_3	-	$3 \times 3, \text{rate} = 36$	✓	-	-
pooling	✓	1×1	✓	✓	-
final branch	-	1×1	✓	-	0.5

c. Decoder

The decoder was an inverse direction of the encoder structure. In this part, the feature map size was expanded by the transpose of the convolution filter. The decoder consisted of three layers that were processes of reverse convolution filters. Reverse layer no.1, named upsample 1, generated 128 channels of feature map and increased the feature map size to 1/8 of the original image. Reverse layer no.2, named upsample 2, generated 64 channels of feature map and increased the feature map size to 1/4 of the original image. The last reverse layer, named upsample 3, retained feature map channels but increased the feature map size to half of the original image. The output from the last layer was passed through the final convolution filter to generate the segmentation result image. This final filter reduced feature channels from 64 into the desired class, nine classes. The detail of each layer is explained in Table 3-7, where 3×3, 256 represents 256 convolution filters of 3×3 filter size. Figure 3-7 shows a sequence of each layer. Each layer generated a result, called side output, by 1×1 convolution. The output from ASPP was also generated to another side output. This approach computed three side outputs and one final output in the training mode. However, it generated only a final output in prediction mode. All outputs have different resolutions from the groundtruth map. Therefore, each output was resized with a bi-linear interpolation function to the exact resolution as a groundtruth map. Figure 3-8 shows the result feature maps from each decoder step; the bottom color images represent the side outputs. The final and side outputs are passed through the cross-entropy function to create the loss function.

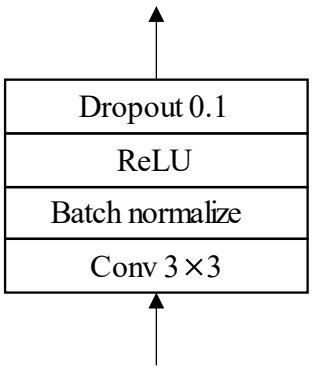


Figure 3-7 Each decoder layer block diagram

Table 3-7 Detail of each layer in the decoder

Layer name	Output size	Convolution filter
upsample 1	$\frac{H}{8} \times \frac{W}{8}$	$3 \times 3, 128$
upsample 2	$\frac{H}{4} \times \frac{W}{4}$	$3 \times 3, 64$
upsample 3	$\frac{H}{2} \times \frac{W}{2}$	$3 \times 3, 64$
final conv	$H \times W$	$1 \times 1, 9$
side conv		$1 \times 1, 9$

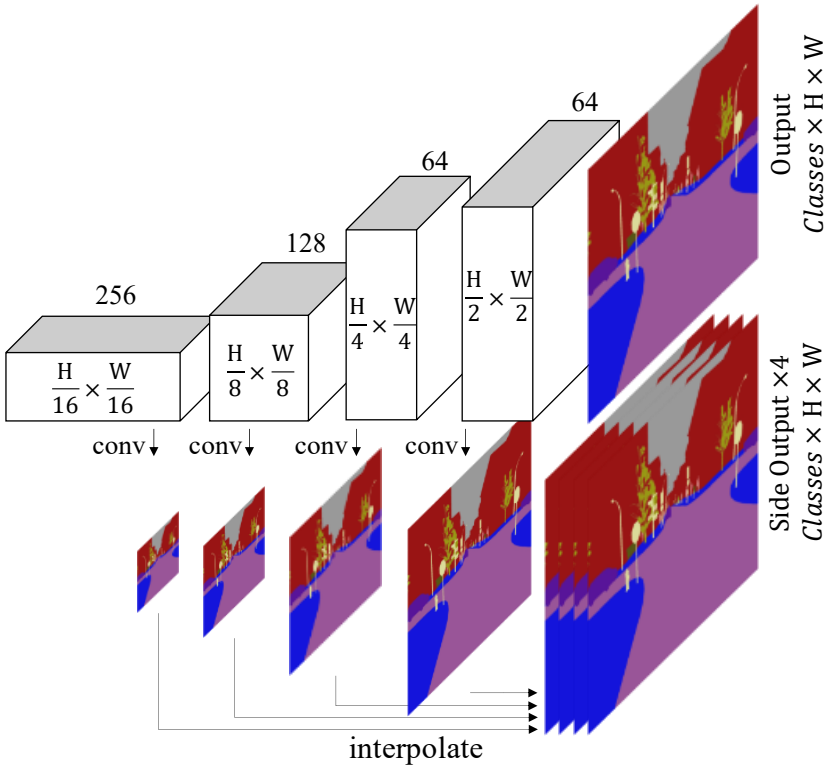


Figure 3-8 Feature map shape in decoder

3.4 Training and Losses

The dataset was separated into two parts; the training and evaluation datasets. The process was done on the platform that specifications are shown in Table 3-8. First, the network structure was trained and compared to the other state-of-the-art networks, BiSeNet[98], DUpsampling, FCN32, ICNet, PSANet, PSPNet, and DeepLabv3. Each network was trained for 287,500 iterations or 250 epochs to decrease the convergent loss. All images in the dataset were resized from their original size into 640×480 pixels and then were fed through the model. The initial parameters for training are described in Table 3-9.

Table 3-8 Training environment

Equipment	Specifications
Central processing unit (CPU)	Intel® Core™ i7-8700K @ 3.70GHz
Random access memory (RAM)	32 GB
Graphics processing unit (GPU)	NVIDIA Geforce GTX™1080Ti
Operating System (OS)	Ubuntu 18.04.6 LTS
Training framework	Pytorch 1.3

Table 3-9 Initial parameters

Parameters	Values
Platform	CUDA
Optimizer	Stochastic Gradient Descent (SGD)
Momentum	0.9
Initial learning rate ($rate_{ini}$)	0.001
weight decay	0.0001
$decay$	0.95
Loss function	cross-entropy function

Most of the models were calculated for losses by comparing final results and groundtruth, but the model introduced in Section 3.2 had five results in total. Hence, the average of all losses calculated the loss of this model. Figure 3-9 shows the training losses from all models on the Snowy road dataset. From the graph, most of the losses, except the DUpsampling model, began to be constant after 10,000 iterations. The black line in the graph represents the average loss of the introduced network. This network achieved the lowest loss value during training compared to other networks at the same training iteration. This loss was also lower than the DeepLabv3 loss, its original model structure. The difference between the proposed model and DeepLabv3 was only the decoder part not included in DeepLabv3. Therefore, using a decoder with pyramid supervision could enhance loss value during the training process. Figure 3-10 shows the feature map in the decoder part step-by-step. The tiny top image is a feature map, containing 256 channels with the size of 1/16 of the final image, after passing the ASPP module. The large bottom image is a result feature map or segmentation image that contains nine channels of the same size as the original image.

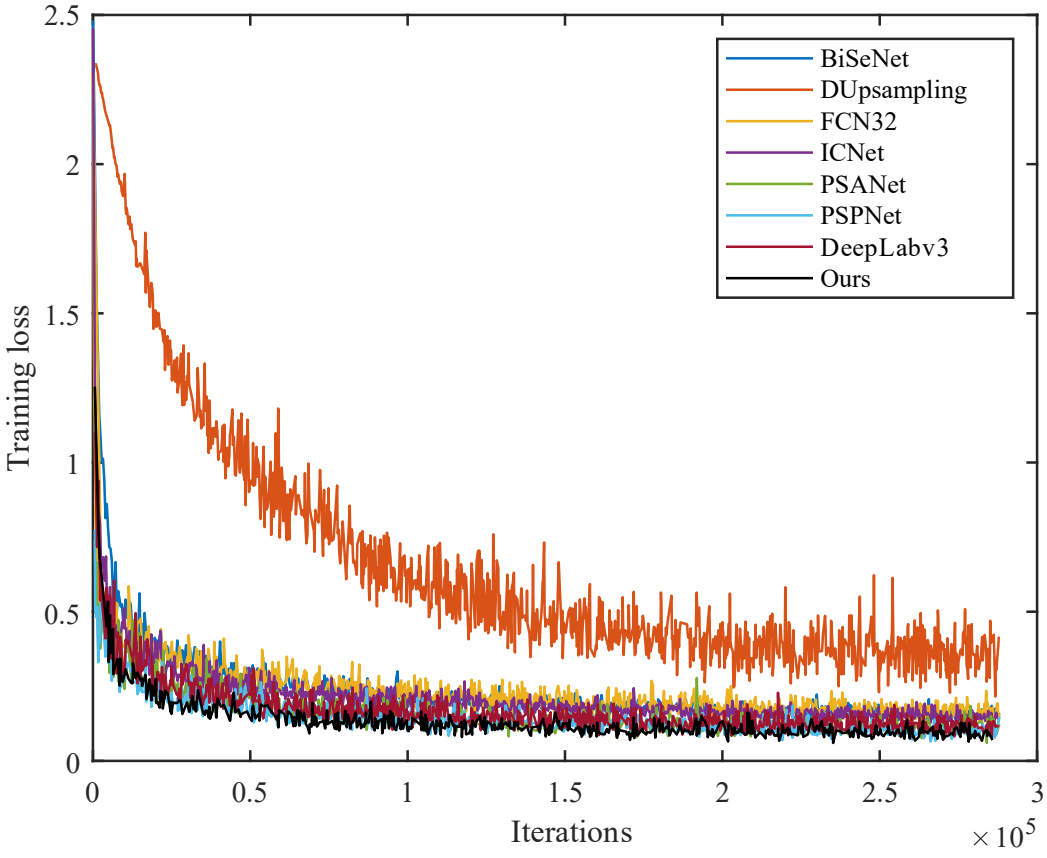


Figure 3-9 Training loss

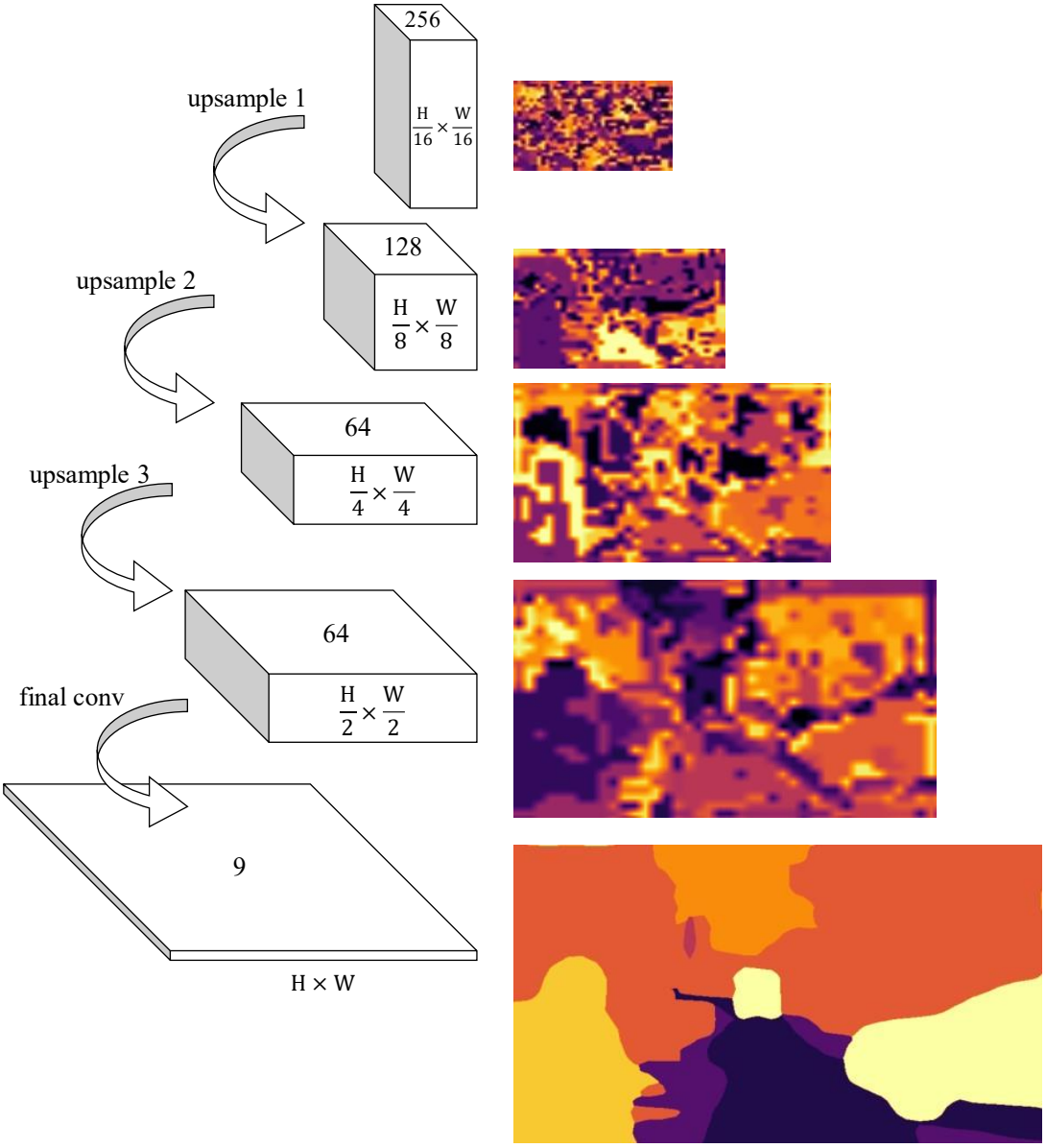


Figure 3-10 Feature map in the decoder

3.5 Results & Evaluation

The semantic segmentation problem based on deep learning could be considered as a pixel classification task. Every model was evaluated by employing the Intersection-over-Union (IoU) and Accuracy (Acc) as indicators to explain the similarity between the segmentation result and groundtruth map. The IoU and Acc were calculated for every class in the result image. Therefore, the mean-IoU (mIoU) and mean-Acc (mAcc) of the image were calculated to represent the overall efficiency of each network. The evaluation was done by feeding a test image into the model one-by-one. Some of the results [3] were published in the 2020 59th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE).

a. Network backbone evaluation

The backbone of every network used in this experiment was based on ResNet architecture. There were five types of the ResNet model: ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152. The larger number of networks represented the more complexity of the network and the processing time. This part tested each ResNet model as a network backbone for DeepLabv3 architecture, which was the original of the proposed model. The dataset used for this comparison was the Mapillary Vistas dataset, a published dataset. Table 3-10 shows the average segmentation results of each network backbone in the DeepLabv3 model.

The results show that the more efficient backbone with the greatest mIoU was ResNet-50. On the other hand, the deeper or more complex network made longer predicting time used. However, every backbone made an adequate processing time, less than 0.035 s or faster than 30 fps. Therefore the best network backbone should be ResNet-50 due to its precision.

Table 3-10 Segmentation performance of each backbone in DeepLabv3

Backbone	mIoU (%)	Processing time (s)
ResNet-18	47.1	0.008
ResNet-34	50.0	0.011
ResNet-50	53.3	0.015
ResNet-101	35.8	0.025
ResNet-152	36.9	0.034

b. Segmentation on Snowy road dataset

The evaluation of this part contained 200 images, which included many situations in the snowy road environment. Table 3-11 shows the efficiency of the proposed model is compared to other models. It reached the highest score in both mIoU and mAcc indicators when compared to other networks. The results of traffic objects were significantly improved up to 7% and 5% for pedestrians. The mIoU of the proposed method was about 2.8% greater than its original model (83.6%).

Figure 3-11 shows the results from each test set. The proposed improved approach could classify with higher accuracy in the small target regions such as a traffic pole, as shown in Figure 3-11 (a) – (c). The small object boundary characterization of the proposed method had higher efficiency than the original one, as shown in the example in Figure 3-11 (c) – (e). This is because the object's shape was more similar to the groundtruth and the detail near the boundary edges are better than others. The red region in Figure 3-12 shows the difference pixel between each result and groundtruth of Figure 3-11 (c) – (e), and Table 3-12 shows the number of error pixels for these images. The errors of the proposed model were the lowest error compared to the other networks.

Table 3-11 Semantic segmentation performance on Snowy road dataset

	IoU(%)									mIoU(%)	mAcc(%)
ID	1	2	3	4	5	6	7	8	9		
BiSeNet	95.0	90.8	81.6	29.2	95.4	93.5	70.5	83.1	91.4	81.2	99.2
DUpsampling	91.0	83.3	30.6	0.2	92.4	90.0	0.7	3.3	83.5	52.8	98.2
FCN32	94.9	89.9	78.8	17.0	94.5	91.1	67.8	81.8	91.0	78.5	99.0
ICNet	93.0	87.2	78.7	27.2	94.7	92.9	69.9	80.4	89.9	79.3	99.0
PSANet	96.1	92.5	86.2	39.6	96.4	94.4	77.4	85.2	92.4	84.5	99.3
PSPNet	96.3	93.0	86.5	41.1	96.4	94.4	77.8	85.2	92.6	84.8	99.3
DeepLabv3	95.7	92.3	84.3	38.0	96.3	94.3	73.0	86.3	92.3	83.6	99.3
Ours	**97.3**	**94.7**	**88.3**	**45.2**	**97.3**	**95.6**	**78.2**	**87.5**	**93.4**	**86.4**	**99.5**

1:Road, 2:Sidewalk, 3:Building, 4:Traffic object, 5:Vehicle, 6:Human, 7:Sky, 8:Vegetation, 9:Unidentified

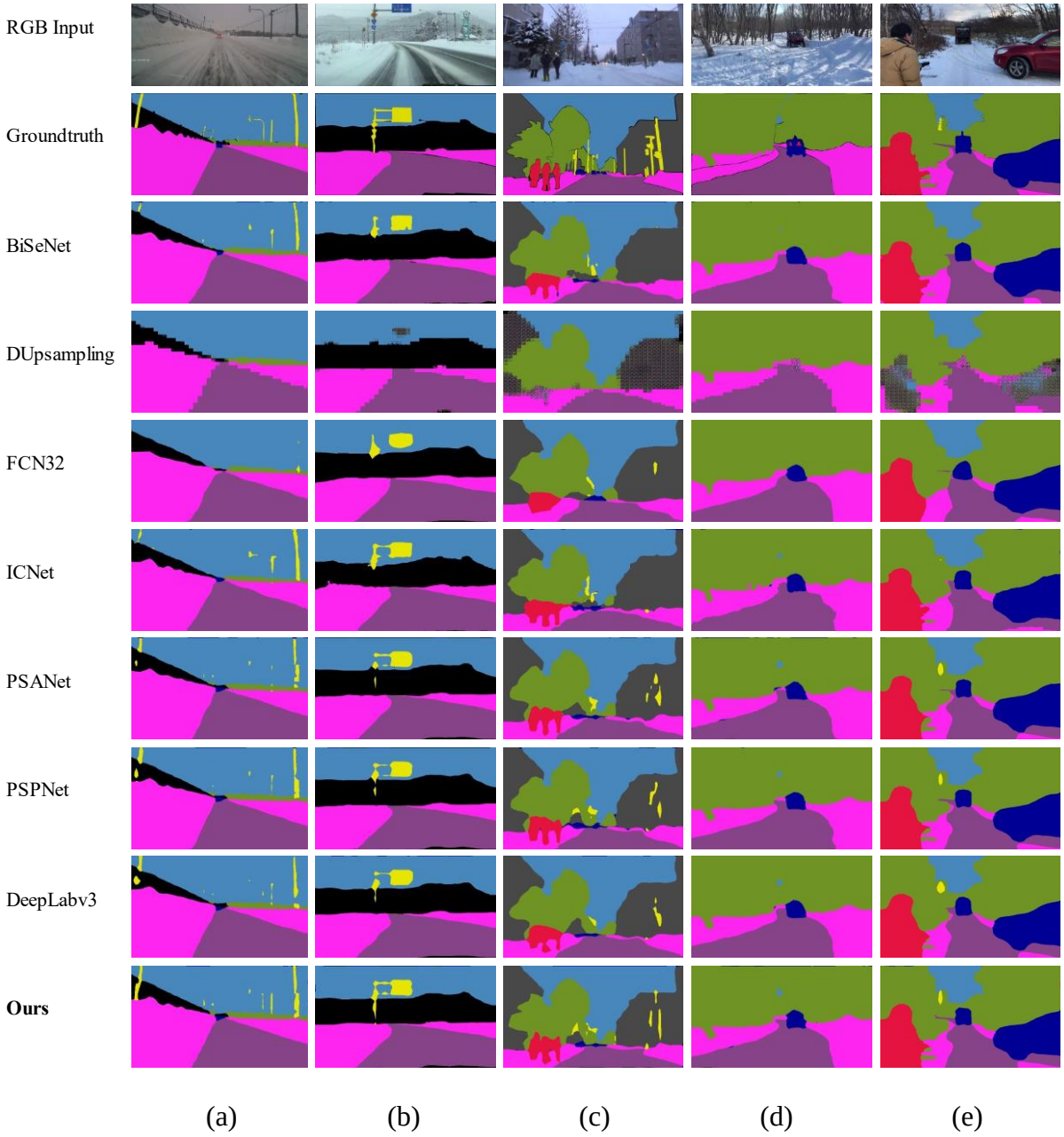


Figure 3-11 Segmentation results on Snowy road dataset

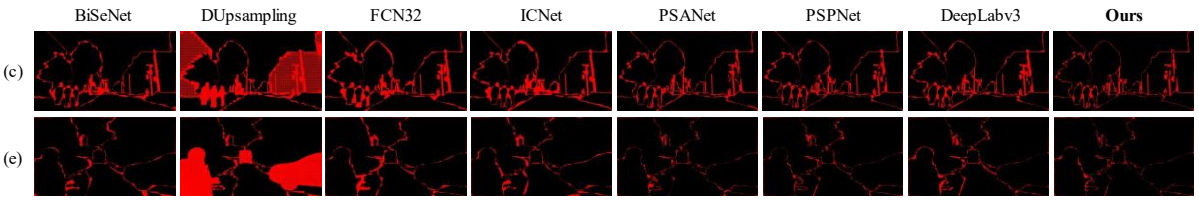


Figure 3-12 Segmentation error of Figure 3-11 (c),(e)

Table 3-12 Segmentation error pixels of Figure 3-11 (c),(e)

	Figure 3-11 (c)		Figure 3-11 (e)	
	Error (pixels)	Error (%)	Error (pixels)	Error (%)
BiSeNet	91,363	9.8	92,639	4.5
DUpsampling	246,998	26.8	576,558	27.8
FCN32	105,946	11.5	104,977	5.1
ICNet	103,831	11.3	94,455	4.6
PSANet	80,311	8.7	60,087	2.9
PSPNet	75,941	8.2	60,861	2.9
DeepLabv3	82,105	8.9	77,665	3.7
Ours	**62,858**	**6.8**	**55,130**	**2.7**

c. Segmentation on Mapillary Vistas dataset

The evaluation of this part contained 62 images, which included the snowy road environment in many locations around the world. Table 3-13 shows the proposed model's efficiency compared to other models. It reached the highest score in both mIoU and mAcc indicators compared to other networks. The results on this dataset were significantly lower than the results on the Snowy road dataset. Most of the results were around 50% of mIoU on the Mapillary Vistas dataset but around 80% of mIoU on the Snowy road dataset. The problem might be from a too-small number of classes to train the network. The result shows that class ID 2,4,6,8 and 12 were significantly less efficient than other classes.

Figure 3-13 shows the results from each test set. The proposed improved approach could classify with higher accuracy in the small target regions such as a human or bicycle, as shown in Figure 3-13 (a) – (b). The small objects boundary characterization of the proposed method had higher efficiency than the original one, as shown in the example in Figure 3-13 (a) – (b), the object shape was more similar to the groundtruth and the detail near the boundary edges are better than others. The red region in Figure 3-14 shows the difference pixel between each result and groundtruth of Figure 3-13 (a) – (b). Table 3-14 shows that these classes, sidewalk, lane marking, city object, human, and landscape, were a tiny number of pixels in the dataset. The numbers of mentioned objects were around 1.5% or lower. Not only the number of pixels cause the efficiency but also the various appearance of the class. With these two main variables, the number of pixels and various class appearances could predict the network become poor.

Table 3-13 Semantic segmentation performance on Mapillary Vistas dataset

	IoU (%)													mIoU(%)	mAcc(%)
ID	1	2	3	4	5	6	7	8	9	10	11	12	13		
BiSeNet	75.8	10.2	72.3	15.1	20.5	0.0	69.9	1.8	67.9	93.8	79.3	16.5	71.3	45.7	97.8
DUpsampling	65.7	0.1	68.6	0.2	0.1	0.0	5.7	0.1	51.2	91.6	76.8	0.0	7.3	28.3	96.7
FCN32	75.1	19.6	73.1	25.5	14.4	0.0	67.9	29.1	65.4	91.9	79.3	24.4	73.4	49.2	97.8
ICNet	74.2	3.3	67.8	12.3	17.5	0.0	61.5	17.1	66.3	92.8	76.5	16.5	66.1	44.0	97.6
PSANet	78.4	22.8	76.3	24.4	26.6	3.0	75.2	38.5	70.7	93.7	81.7	27.4	75.3	53.4	98.1
PSPNet	78.7	21.5	76.8	23.6	25.1	3.7	75.2	36.0	71.1	94.0	81.6	30.1	76.7	53.4	98.1
DeepLabv3	78.3	21.9	76.1	26.3	26.6	0.3	74.3	36.0	69.8	93.9	81.8	31.7	75.5	53.3	98.1
Ours	**79.5**	**23.6**	**77.5**	**25.7**	**28.5**	**6.4**	**77.4**	**32.7**	**70.6**	**94.1**	**82.7**	**29.0**	**76.8**	**54.2**	**98.2**

1:Road, 2:Sidewalk, 3:Building, 4:Lane marking, 5:Traffic object, 6:City object, 7:Vehicle, 8:Human, 9:Snow, 10:Sky, 11:Vegetation, 12:Landscape, 13:Unidentified

Table 3-14 Segmentation error pixels of Figure 3-13 (a),(b)

	Figure 3-13 (a)		Figure 3-13 (b)	
	Error (pixels)	Error (%)	Error (pixels)	Error (%)
BiSeNet	366,983	4.6	1,209,634	15.1
DUpsampling	1,374,017	17.2	2,904,495	36.4
FCN32	454,166	5.7	1,289,159	16.1
ICNet	390,783	4.9	1,161,764	14.5
PSANet	258,006	3.2	833,104	10.4
PSPNet	272,993	3.4	837,189	10.5
DeepLabv3	272,518	3.4	966,059	12.1
Ours	**210,113**	**2.6**	**560,178**	**7.0**

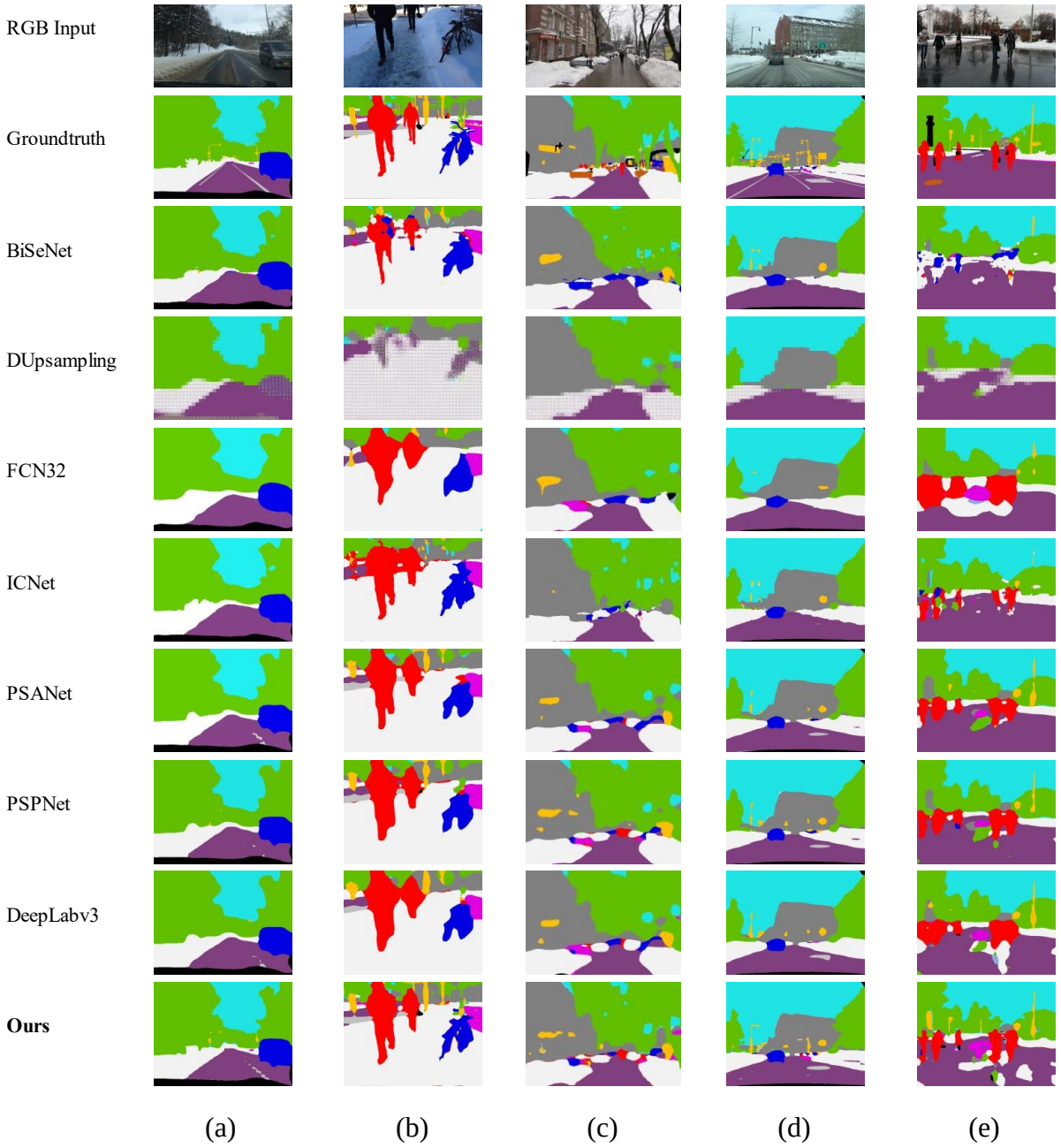


Figure 3-13 Segmentation results on Mapillary Vistas dataset

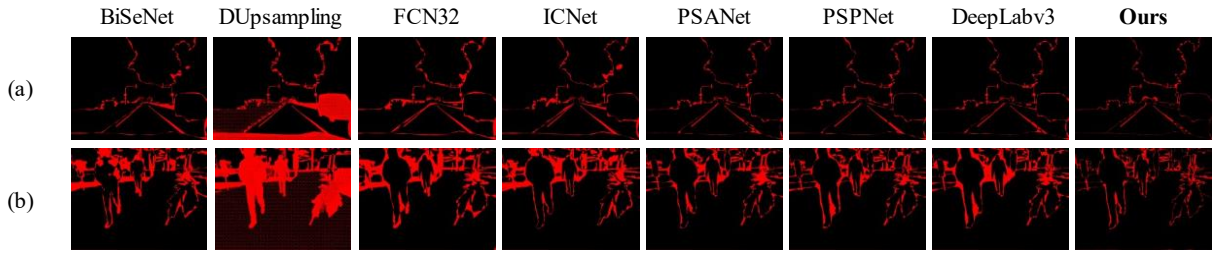


Figure 3-14 Segmentation error of Figure 3-13 (a),(b)

d. Segmentation speed

The processing speed of each model was evaluated by measuring the time consuming from before importing the image until exporting a result. This process was based on only single image segmentation. Table 3-15 shows the processing time of each network. The real-time segmentation networks, the BiSeNet and ICNet, had longer processing times than usual. However, these networks could process many images in a batch, processing many images in parallel. For example, BiSeNet could operate 20 images simultaneously, and ICNet could do 16 images. Therefore, the network's actual processing speed should be calculated on its maximum performance. Although some networks could predict many images in parallel, they would operate only a single image during actual usage. In actual operation, the image was taken by RGB camera one by one and then fed into the network to make the segmentation result. Accordingly, the evaluation in the processing time should calculate only one image at a time.

The proposed network developed from DeepLabv3 had the same operating time as its original network. Therefore, it could be concluded that using the pyramid supervision as a decoder could improve the network efficiency but had a small effect on the processing speed.

Table 3-15 Single input segmentation processing performance

	Processing performance	
	Time (s)	Speed (fps)
BiSeNet	0.010	100
DUpsampling	0.013	77
FCN32	0.003	333
ICNet	0.019	53
PSANet	0.014	71
PSPNet	0.014	71
DeepLabv3	0.013	77
Ours	**0.013**	**77**

CHAPTER 4

MULTIPLE INPUTS EXPERIMENT

4. MULTIPLE INPUTS EXPERIMENT

This experiment recognized the snowy road environment by using multiple information as inputs. This was a more complicated experiment than the single one because different processes were required. The datasets in this experiment were RGB images, Depth maps, and Thermal maps; therefore, the sensors combination and 3D visualization were included in this section. The objective of this experiment was to classify all pixels in the image into each different object as same as the single input experiment. In addition, this experiment would represent the result in 3D visualization.

4.1 Proposal of Algorithm

This experiment was set up to introduce the most robust network structure with the feature fusion structure for semantics segmentation in snowy environments by using an RGB image from the RGB camera and a Thermal map from the thermal camera.

4.2 Datasets

There were five datasets used in the multiple inputs experiment. Three were self-made datasets, including the snowy environment of Hokkaido prefecture in Japan. The others were the published dataset from the Cityscapes and Synthia datasets, containing many images worldwide. All datasets used in the experiment included more than one information such as RGB image – Depth map, RGB image – Thermal map, or RGB image – Depth map – Thermal map.

a. Snowy Sidewalk dataset (SSW)

This dataset consisted of the snowy sidewalk scenario at Hokkaido University during the winter of 2020-2021. The dataset contained RGB images, Thermal maps, and Depth maps collected by many sensors. An RGB camera captured the RGB image, a thermal camera collected the Thermal map, and LiDAR generated the Depth map, as shown in Figure 4-1. All sensor information was represented in Table 4-1. The sensors were installed at the front of the remoted controlled wheelchair, then drove along the sidewalk in the experimental area. The dataset included 887 images of each piece of information with resolutions of 380×290 pixels. As described in Table 4-2, every image was labeled into nine classes in this dataset. Figure 4-2 shows the example images of this dataset. Figure 4-3 shows an example of each class.

Table 4-1 Equipment information

Equipment	Specifications
RGB camera	ZED Mini
Thermal camera	Optris PI 640i
LiDAR	OS1-64
GPS	AQLOC-Light
IMU	MTi-100 IMU
Wheelchair	WHILL Model CR

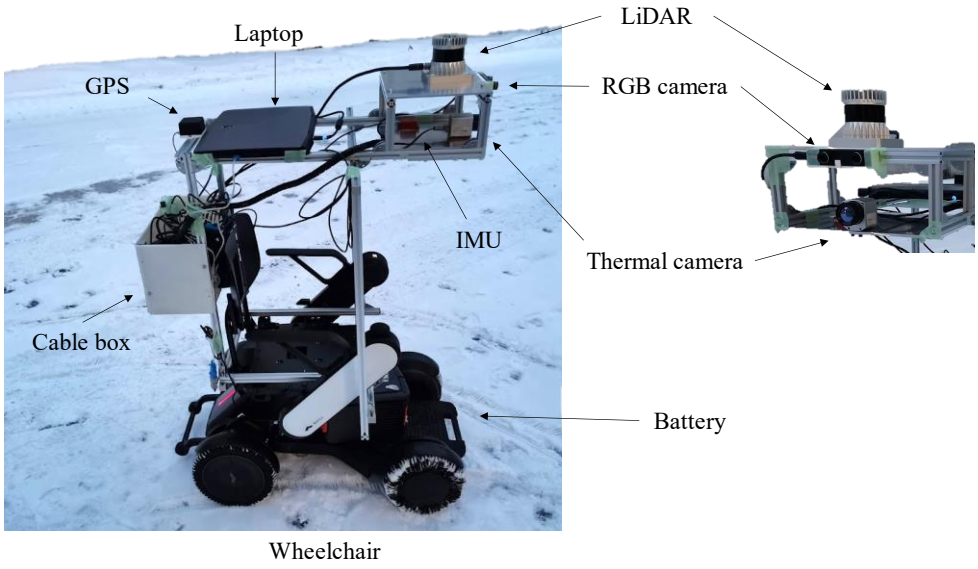
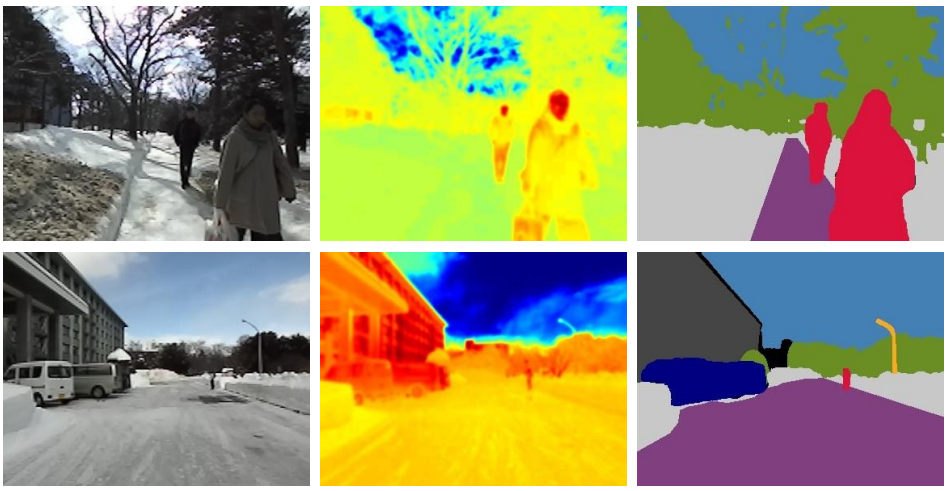


Figure 4-1 Sensors installation on automatic wheelchair



(a) (b) (c)

Figure 4-2 Examples of snowy road environment; (a) RGB images, (b) Thermal map, (c) labeled images

Table 4-2 Snowy Sidewalk dataset (SSW)

ID	Name	Pixels (pixel)	Pixels (%)	Color
1	Road	1.8E+07	18.0	
2	Snow mountain	2.4E+07	24.7	
3	Building	7.5E+06	7.7	
4	Traffic object	4.2E+05	0.4	
5	Vegetation	2.7E+07	27.1	
6	Sky	1.8E+07	18.4	
7	Human	1.8E+06	1.8	
8	Vehicle	1.5E+06	1.5	
9	Unidentified	2.5E+05	0.3	

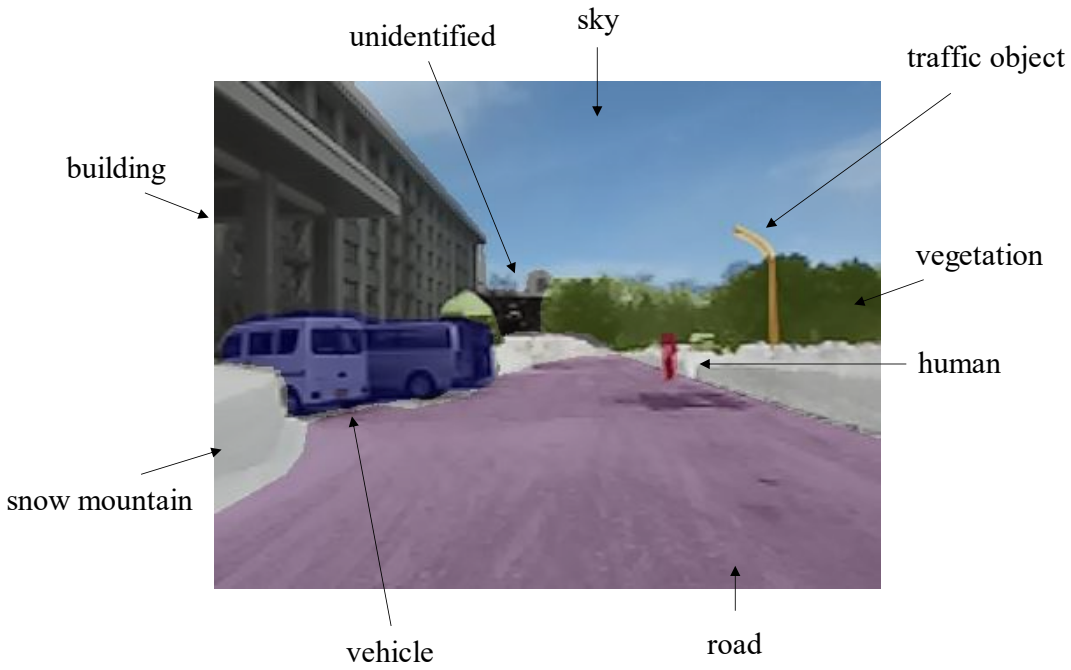


Figure 4-3 Example of each class

b. Snow Remover Machine datasets (SRM-1, SRM-2)

These datasets consisted of the snowy sidewalk scenario in Sapporo city of Hokkaido prefecture during the winter of 2020–2021. The datasets were collected at the operating site of the snow remover machine. The SRM-1 was the simulation of snow remover machine operation and collected at the parking area of Miyanosawa area in Sapporo city. Therefore, this dataset contained both daytime and nighttime situations. The SRM-2 was the real environment of snow remover machine operation taken at the sidewalk of Miyanosawa area in Sapporo city. However, the machine operation was done at nighttime due to the convenience of pedestrians; therefore, this dataset contained only the nighttime situation. Figure 4-4 and Figure 4-5 show the installation of the sensor of both datasets, SRM-1 and SRM-2.

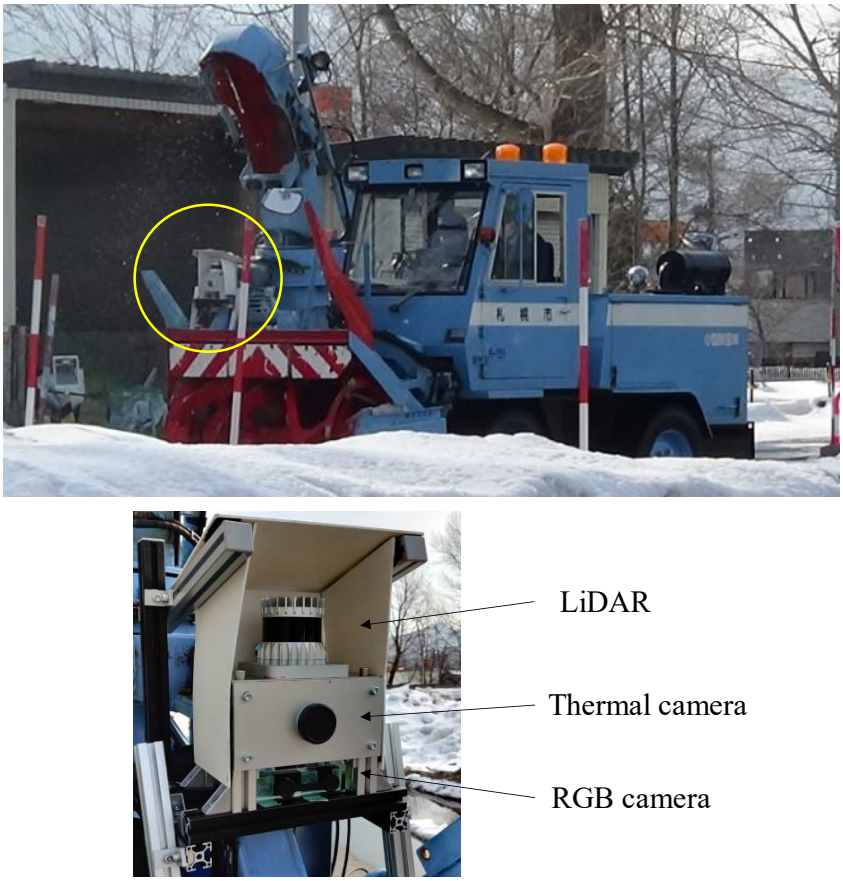


Figure 4-4 Sensors installation on snow remover machine for SRM-1

Table 4-3 Equipment information

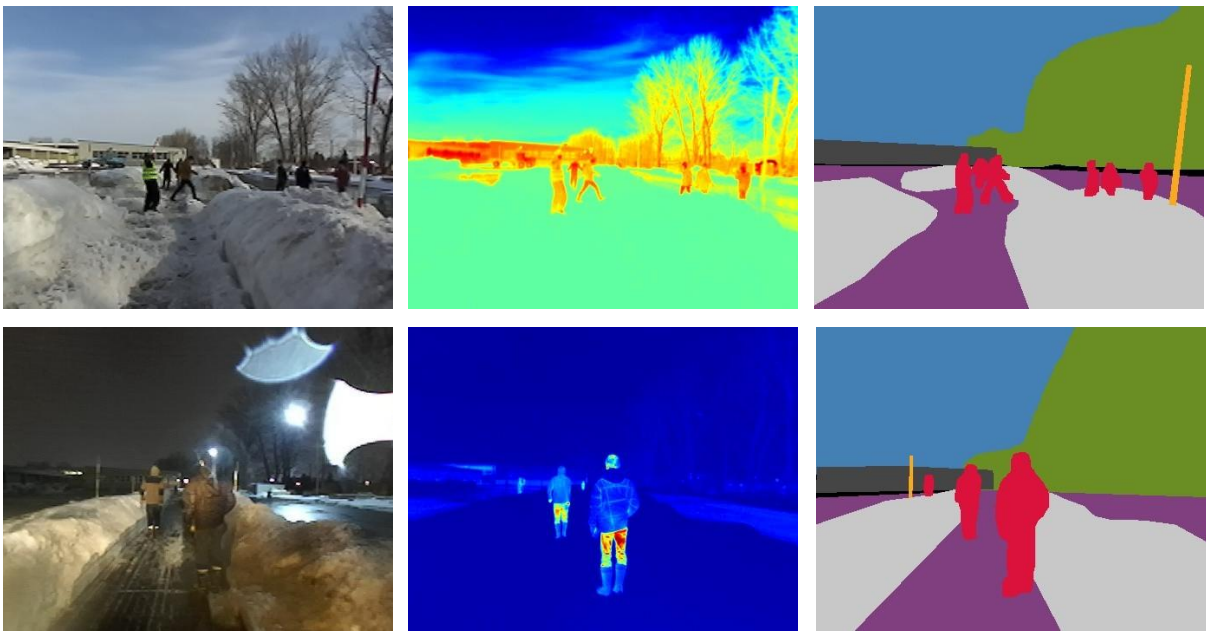
Equipment	Specifications
RGB camera (SRM-1)	ZED Mini
RGB camera (SRM-2)	ELP USB camera module
Thermal camera	Optris PI 640i
LiDAR	OS1-64

Figure 4-5 Sensors installation on snow remover machine for SRM-2

Both datasets contained RGB images, Thermal maps, and Depth maps collected by many sensors. An RGB camera captured the RGB image, a thermal camera collected the Thermal map, and LiDAR generated the Depth map. All sensor information was represented in Table 4-3. The sensors were installed at the front of the machine for SRM-1 and the top-front for SRM-2, as shown in Figure 4-5. The SRM-1 dataset included 1571 images of each piece of information with resolutions of 450×350 pixels. The SRM-2 dataset included 780 images with a resolution of 1064×586 pixels. Table 4-4 and Table 4-5 described that every image was labeled into nine classes in both datasets. Figure 4-6 and Figure 4-7 show the example images of both datasets, SRM-1 and SRM-2.

Table 4-4 Snow Remover Machine-1 dataset (SRM-1)

ID	Name	Pixels (pixel)	Pixels (%)	Color
1	Road	4.3E+07	17.3	
2	Snow mountain	6.3E+07	25.5	
3	Building	1.1E+07	4.4	
4	Traffic object	6.0E+05	0.2	
5	Vegetation	5.3E+07	21.4	
6	Sky	6.4E+07	26.0	
7	Human	5.3E+06	2.2	
8	Vehicle	5.7E+05	0.2	
9	Unidentified	6.8E+06	2.8	

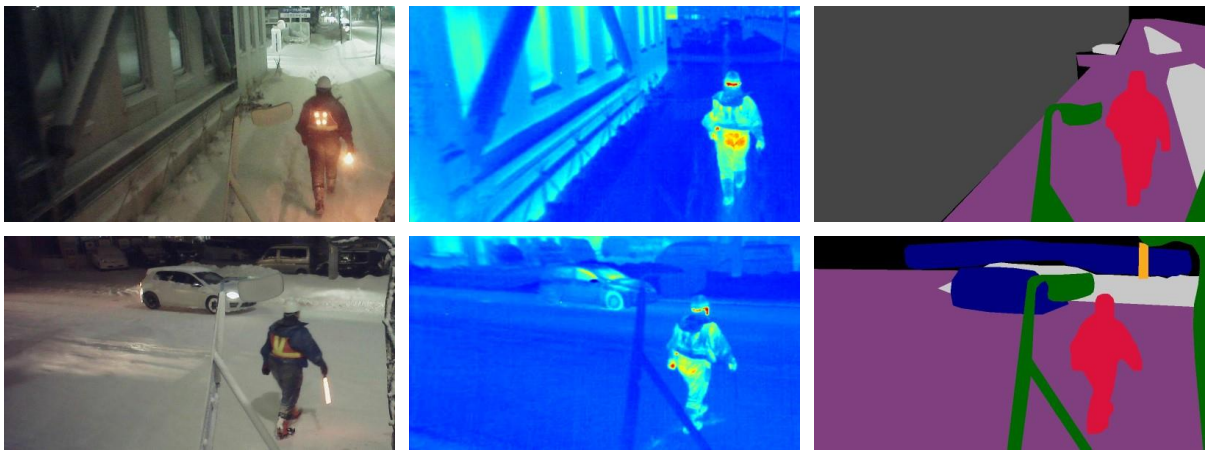


(a) (b) (c)

Figure 4-6 Examples of SRM-1; (a) RGB images, (b) Thermal maps, (c) labeled images

Table 4-5 Snow Remover Machine-2 dataset (SRM-2)

ID	Name	Pixels (pixel)	Pixels (%)	Color
1	Road	2.0E+08	40.6	
2	Snow mountain	4.4E+07	9.0	
3	Building	1.4E+08	29.1	
4	Traffic object	1.5E+07	3.1	
5	Vegetation	6.2E+06	1.3	
6	Sky	9.9E+06	2.0	
7	Human	2.0E+07	4.2	
8	Vehicle	3.7E+07	7.7	
9	Unidentified	1.4E+07	2.9	



(a) (b) (c)

Figure 4-7 Examples of SRM-2; (a) RGB images, (b) Thermal maps, (c) labeled images

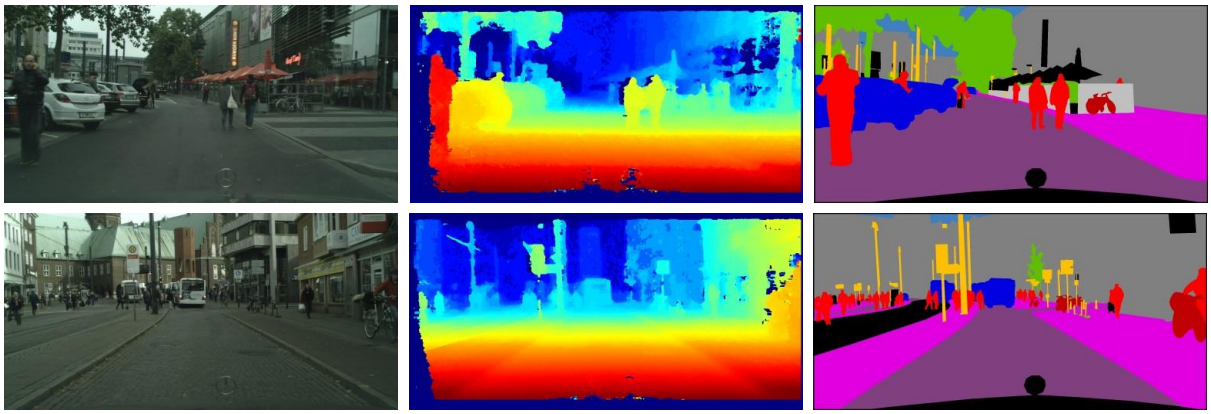
c. Cityscapes dataset

The Cityscapes dataset was published by [99] in 2016. It contained authentic RGB images and Depth maps (RGB-D images) for road and city environments. The dataset consisted of complex surroundings with various weather conditions in more than 50 cities. A stereo camera collected the dataset with a resolution of 2048 × 1024 pixels. Hence, it contained RGB images and disparity images, representing the Depth information. The left 8-bit image sets of this Cityscapes dataset were used in this experiment. In addition, this dataset contained 3474 training, validation, and testing set labeled into 11 classes, as shown in Table 4-6. Figure 4-8 shows example images of the Cityscapes dataset. Figure 4-9 shows examples of each class.

The Cityscapes dataset did not contain the snowy condition in the image. However, this dataset was one of a few city scenario datasets that included multiple initial information. Therefore, the dataset was used to evaluate the combined performance of the model.

Table 4-6 Cityscaps dataset

ID	Name	Pixels (pixel)	Pixels (%)	Color
1	Road	2.4E+09	33.4	
2	Sidewalk	3.9E+08	5.3	
3	Building	1.5E+09	20.3	
4	Fence	9.9E+07	1.4	
5	Traffic object	1.3E+08	1.8	
6	Vegetation	1.1E+09	15.2	
7	Sky	2.5E+08	3.5	
8	Human	8.9E+07	1.2	
9	Vehicle	5.0E+08	6.9	
10	Bicycle	2.9E+07	0.4	
11	Unidentified	7.7E+08	10.5	



(a) (b) (c)

Figure 4-8 Examples of Cityscapes; (a) RGB images, (b) Depth maps, (c) labeled images

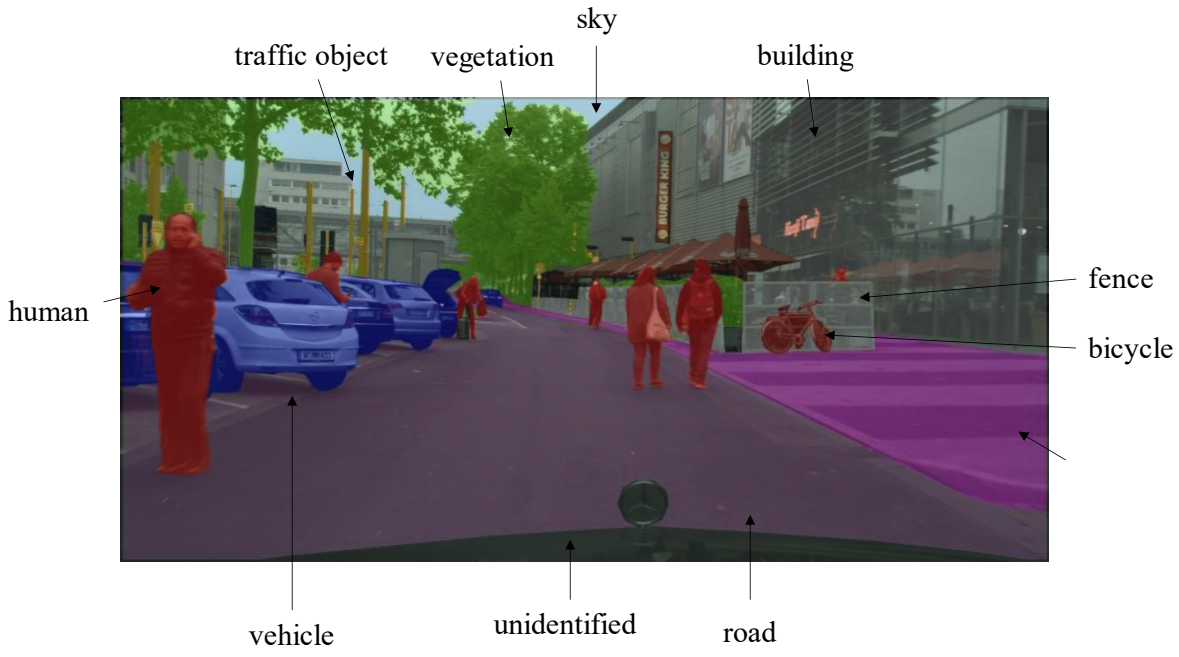


Figure 4-9 Example of each class

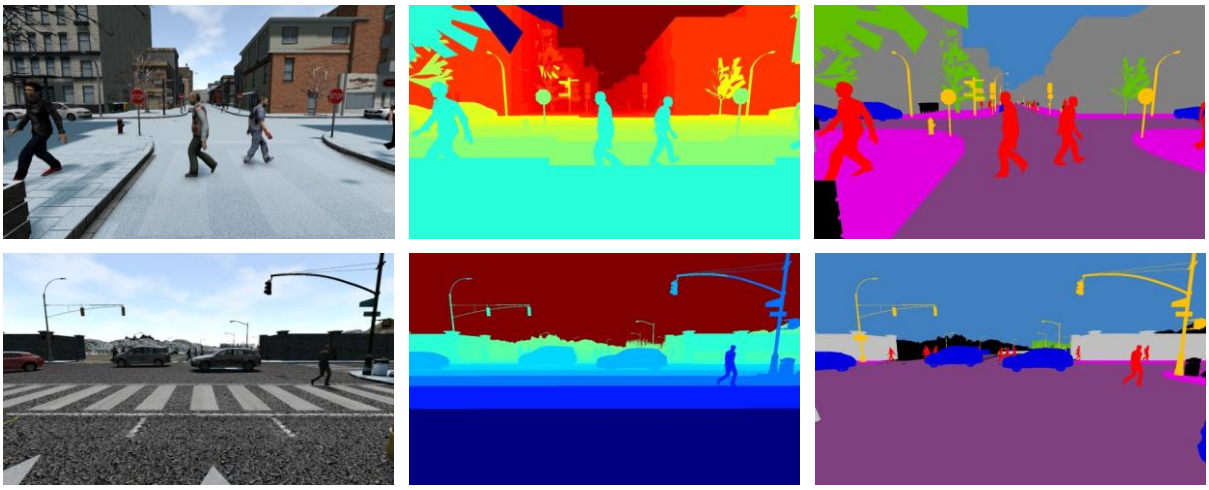
d. Synthia dataset

The Synthia dataset [100] was an outdoor dataset using the Unity Engine [101] to render a virtual city. The dataset was published by Computer Vision Center, Barcelona, and Universitat Autònoma de Barcelona in 2016. It contained 4686 sets of realistic RGB images, Depth maps, and annotated label sets with an image resolution of 1280×760. This experiment used only the winter datasets of all packages, including Highway Seqs-01, New York ish Seqs-02, Old European Town Seqs-04, New York ish Seqs-05, and Highway Seqs-06. The classes of these datasets were reduced to 10 classes to suit our specific goal needs, as shown in Table 4-7. Examples of the Synthia dataset are shown in Figure 4-10 and Figure 4-11 shows examples of each class.

Unlike the Cityscapes dataset, the Synthia dataset contained snowy road scenarios. Nevertheless, this dataset did not have real-world information like other datasets. Therefore, this dataset was used to support the combined performance of the network.

Table 4-7 Synthia dataset

ID	Name	Pixels (pixel)	Pixels (%)	Color
1	Road	6.3E+08	13.7	
2	Sidewalk	9.2E+08	20.1	
3	Building	1.8E+09	40.4	
4	Fence	1.8E+08	3.9	
5	Traffic object	1.1E+08	2.4	
6	Vegetation	1.4E+08	3	
7	Sky	5.6E+07	1.2	
8	Human	1.8E+08	4	
9	Vehicle	1.1E+07	0.2	
10	Unidentified	5.0E+08	10.9	



(a) (b) (c)

Figure 4-10 Examples of Synthia; (a) RGB images, (b) Depth maps, (c) labeled images

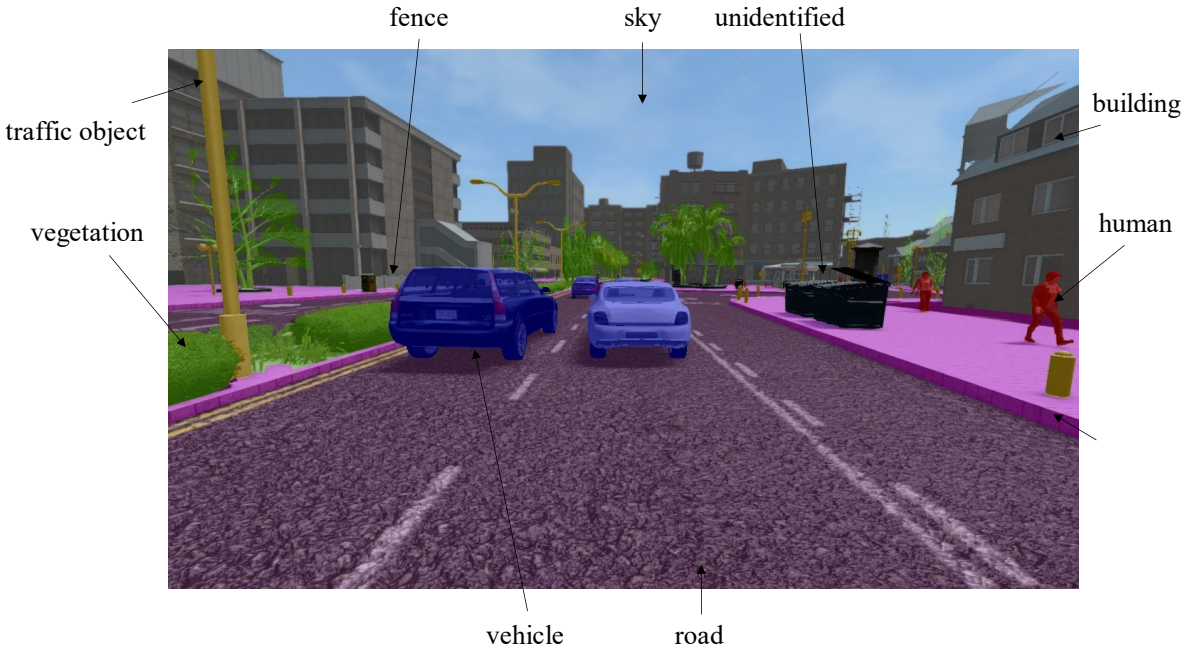


Figure 4-11 Example of each class

e. Dataset calibration

Different types of sensors collected the datasets; thus, the information from each sensor could not be matched directly. The images needed to calibrate together to make each pixel in the image linkable. Each sensor had a different data collecting frequency, making the information not collected simultaneously. However, the frequency rate of the data collecting was around 15–30 Hz, which was fast enough to neglect the different sampling times. Therefore, the nearest timestamp from each sensor information was defined at the same situation shot.

Figure 4-12 shows the raw data from each sensor. Both images had different object positions from each other. In this experiment, the RGB image was used as the reference image. Hence, there was no transformation applied to the RGB image. On the other hand, the Thermal map had to transform the image shape by Eq. (2-7) to match the reference. Lastly, each image was cropped to avoid the data-less pixel, as shown in Figure 4-13.

(a) (b)

Figure 4-12 Example of non-calibrated SRM-1 dataset; (a) RGB image, (b) Thermal map

(a) (b)

Figure 4-13 Example of calibrated SRM-1 dataset; (a) RGB image, (b) Thermal map

f. Feature maps representation

The datasets in this experiment consisted of feature maps such as Depth maps or Thermal maps. These maps were generated to represent only one feature as their name. For example, the Depth map contained the distance value between each object and sensor. The Thermal map illustrated the temperature of each object. As described, both feature maps contained only one value, so the original maps from sensors were monochromic images or grayscale images. The grayscale image was one-channel, unlike the RGB image, as shown in Figure 4-14 (a).

Most of the segmentation network architectures required a three-channel image as an input. Therefore, the one-channel image was converted into a three-channel. Not only was the channel of the image converted, but also the colormap was changed. There were many kinds of the colormap, such as black-white(grayscale), HSV, hot, ocean, winter, autumn, or jet map. The jet colormap was used to represent the feature map in this experiment. This colormap had various colors from blue, the lowest value, to the red, the highest value. Figure 4-14 (b) shows the converted image of Figure 4-14 (a). The white color in Figure 4-14 (a) was changed to red in Figure 4-14 (b), and the black color was blue. The top row images are the Thermal maps from SRM-1, and the bottom ones are Depth maps from Cityscapes.

(a) (b)

Figure 4-14 Colormap conversion; (a) original maps, (b) jet colormap

4.3 Network Architecture

This part proposes a developed neural network for segmentation in the snowy environment. The network was adapted from a famous and influential network structure named DeepLabv3. In this network, a structure called ResNet50 was used as a network backbone for feature extraction. The Atrous Spatial Pyramid Pooling layers (ASPP) of the DeepLabv3 were applied to capture multi-scale information for the segmentation head module. Decoder layers substituted the output map-generating layers. This network also combined the pyramid supervision of RedNet to enhance the training process. Unlike the single input segmentation network, this structure consisted of two encoders working parallelly to support the multiple inputs. In addition to the single input network, this one also contained the fused module, which was introduced to merge the feature maps from two encoders. Figure 4-15 shows the overall structure. Each part of the network will be described as follow.

a. Encoder

The encoder of this network was the ResNet-50 structure, the same as Section 3.2-a. The purpose of the encoder was to increase the number of feature maps from 3 to 2048 channels. It begins with the downsample operation, downsample 0, of a sequence of a 7×7 convolution layer sequence with stride two padding three, normalization, Rectified Linear Unit, and 3×3 max-pooling layer with stride two paddings one. This first layer reduced feature size to 1/4 of the original image and increased the channel to 128 channels. The following four layers of this encoder were processes of bottlenecks of convolution filters. Bottleneck no.1, named down- downsample 1, generated 256 channels of feature map but retained the feature map size at 1/4 of the original image. Bottleneck no.2, named downsample 2, generated 512 channels of feature map and reduced the feature map size to 1/16. Finally, bottlenecks no.3 and 4, named downsample 3 and 4, generated 1024 and 2048 channels of feature map without size reduction. Figure 4-16 shows an example of a bottleneck diagram. The above-explained sequence was done per each input, hence this encoder contained two sequences in parallel. The information of each layer is explained in Table 4-8, where 7×7, 64 represents 64 convolution filters of 7×7 filter size. The original image was converted to the feature map with the shape of 2048×60×80 by using this encoder. Figure 4-17 shows the feature maps during each step in the encoder.

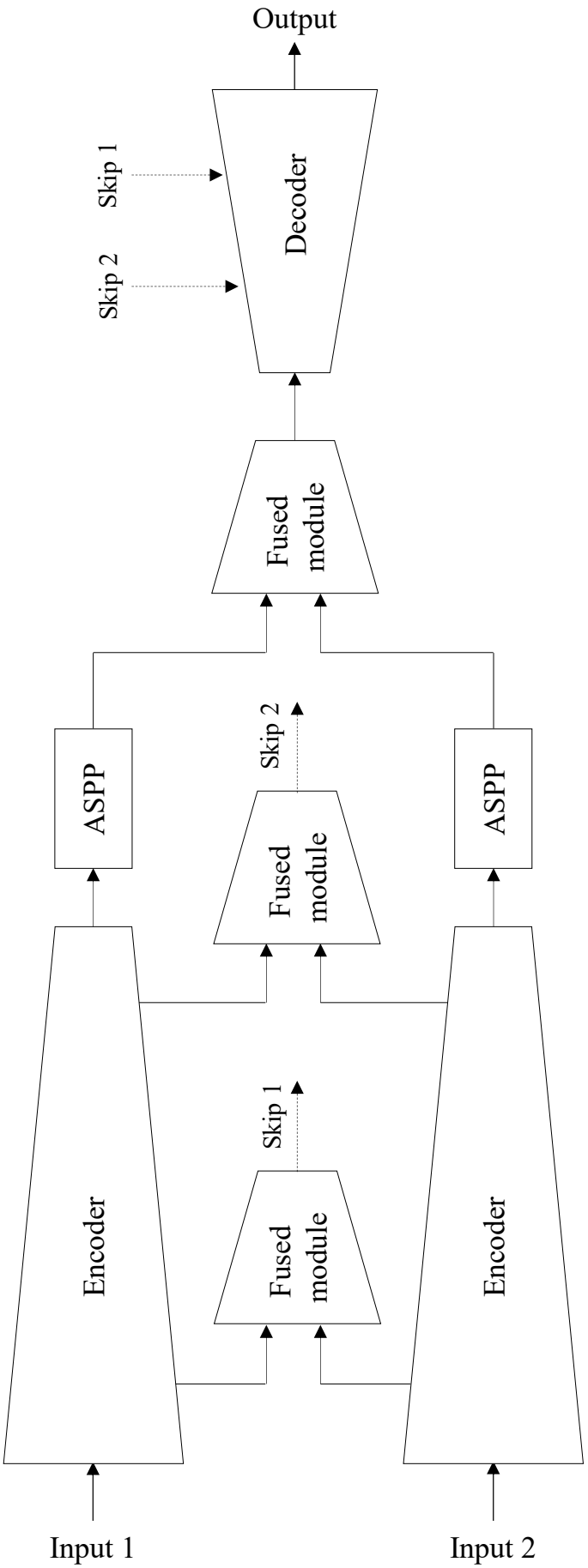


Figure 4-15 Overall network diagram

Table 4-8 Detail of each layer in encoder

Layer name	Output size	Convolution filter
input	$H \times W$	
downsample 0	$\frac{H}{4} \times \frac{W}{4}$	$7 \times 7, 64$
downsample 1	$\frac{H}{4} \times \frac{W}{4}$	$\begin{bmatrix} 1 \times 1, & 64 \\ 3 \times 3, & 64 \\ 1 \times 1, & 256 \end{bmatrix} \times 3$
downsample 2	$\frac{H}{16} \times \frac{W}{16}$	$\begin{bmatrix} 1 \times 1, & 128 \\ 3 \times 3, & 128 \\ 1 \times 1, & 512 \end{bmatrix} \times 4$
downsample 3	$\frac{H}{16} \times \frac{W}{16}$	$\begin{bmatrix} 1 \times 1, & 256 \\ 3 \times 3, & 256 \\ 1 \times 1, & 1024 \end{bmatrix} \times 6$
downsample 4	$\frac{H}{16} \times \frac{W}{16}$	$\begin{bmatrix} 1 \times 1, & 623 \\ 3 \times 3, & 512 \\ 1 \times 1, & 2048 \end{bmatrix} \times 3$

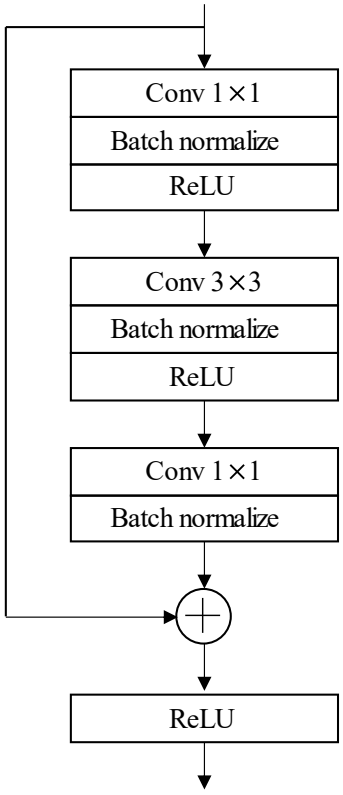


Figure 4-16 Bottleneck block diagram

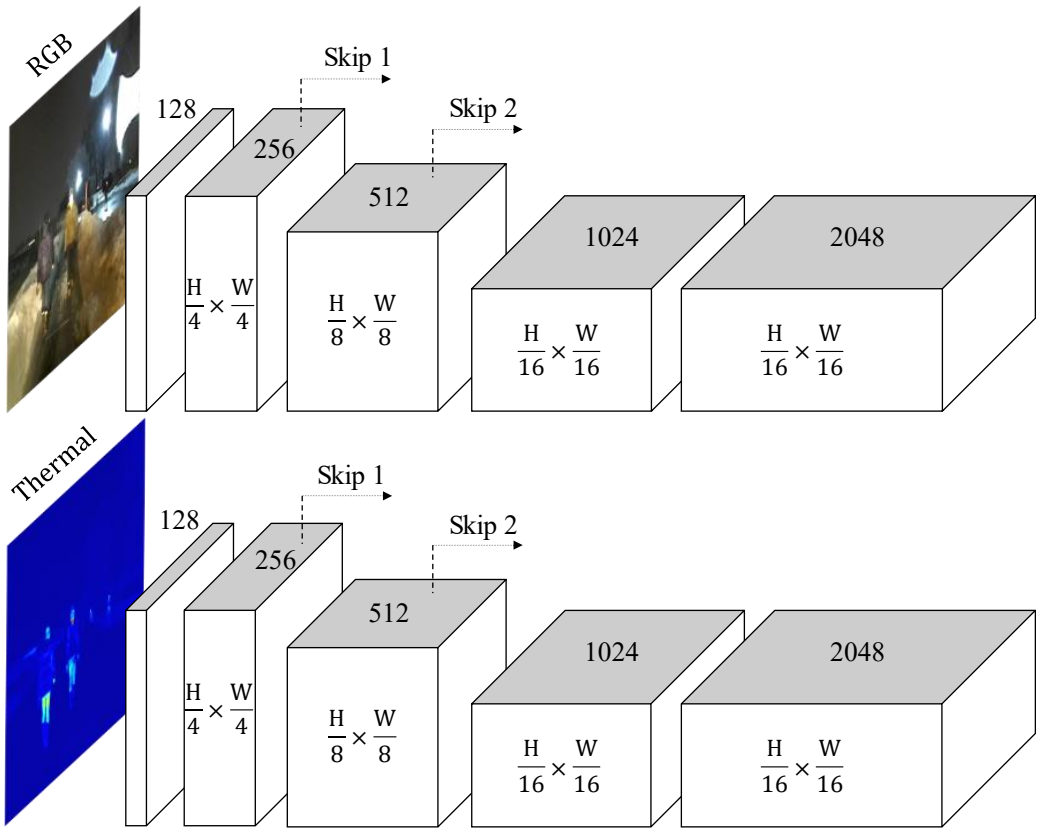


Figure 4-17 Feature map shape in encoders

Table 4-9 Detail of each branch in ASPP

Branch name	Average pooling	Convolution filter	Normalizer and ReLU	Interpolation	Dropout
conv1	-	1×1	✓	-	-
conv3_1	-	$3 \times 3, \text{rate} = 12$	✓	-	-
conv3_2	-	$3 \times 3, \text{rate} = 24$	✓	-	-
conv3_3	-	$3 \times 3, \text{rate} = 36$	✓	-	-
pooling	✓	1×1	✓	✓	-
final branch	-	1×1	✓	-	0.5

b. Atrous Spatial Pyramid Pooling (ASPP)

The Atrous Spatial Pyramid Pooling or ASPP module, introduced in the DeepLabv3 network, was utilized to capture long-range context and multi-scale information. The module consisted of 5 parallel branches of 4 convolution branches (one 1×1-kernel, three 3×3-kernels) and one pooling branch. Figure 4-18 shows the ASPP module. Each 3×3-kernel convolution branch included convolution with rates = {12, 24, 36}, batch normalization, and rectified linear unit (ReLU). The pooling branch includes adaptive average pooling, 1×1 convolution branch, and interpolation. Each branch reduced the feature map channels from 2048 to 256 channels. All five outputs, the same feature size at 256 channels, were combined with the concatenation operator to obtain a feature map with 1280 channels. Then, it was fed into another 1×1 convolution branch with a 0.5 value of dropout. The final output had the same size as the module's input, but the channels were decreased to 1024 channels. The information of each branch is explained in Table 4-9.

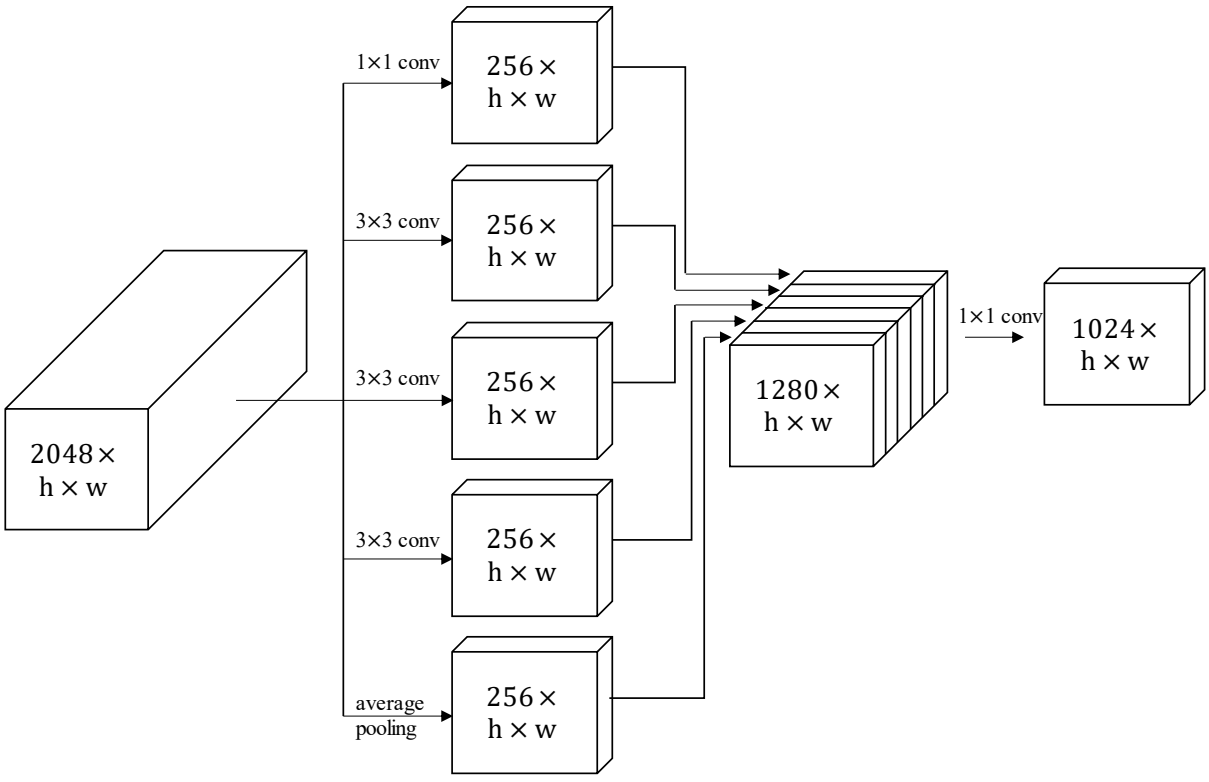


Figure 4-18 Feature map shape in ASPP

c. Fused Module

For segmentation, only RGB images were not enough in the snowy environment because they contained only a feature of color. The color differences could not represent every object in the surrounding area, especially in snowy situations where snow reduced the gradient of different objects' colors, turning most of the area into the same color. Consequently, the Thermal map that contains a temperature feature to support the loss feature from the snow was introduced. Thermal map utilization was more effective than the color image for separating living things from the ambient environment, which did not emit heat. The network consisted of two encoders, one for the RGB image and another for the Thermal map, to improve snow segmentation. An architecture unit that can fuse and select the different maps' essential features was proposed to merge both feature maps from two encoders. The network could learn to intensify the best informative features and minimize the less important information. The fusion module was adapted from the AdapNet++ [92] fusion module, making it more suitable to work with a temperature feature map. The module began with concatenating the two feature maps together and then fed them into three branches: a convolution branch (cb), a convolution with rate branch (cbr), and a skip branch (skip). Figure 4-19 shows the diagram of the fusion module, the symbol ⓘ represents the concatenation operator. Two convolution branches were a sequence of 3×3 convolution layers with a small number of filters, a linearized unit, a 3×3 convolution layer with the same number of filters that was a result of the concatenator, and the softplus function. This module introduced the softplus function as an element-wise function instead of ReLU in each branch's final step. Here, softplus was a smooth approximation to the ReLU function, as defined by Eq. (2-28).

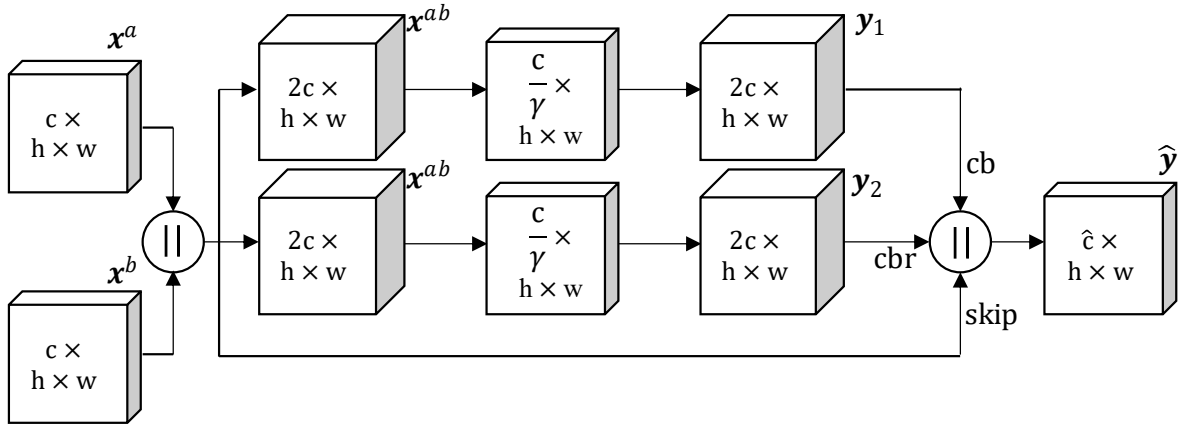


Figure 4-19 Feature map shape in fused module

Given $\boldsymbol{x^a} \in \mathbb{R}^{c\times h\times w}$ and $\boldsymbol{x^b} \in \mathbb{R}^{c\times h\times w}$ denote the feature maps from RGB images and the Thermal map, where c is the channels of the feature and $h \times w$ is the dimension of the feature. The concatenated result of $\boldsymbol{x^a}$ and $\boldsymbol{x^b}$ is represented with $\boldsymbol{x^{ab}} \in \mathbb{R}^{2c\times h\times w}$, and the results from each convolution branch are $\boldsymbol{y_i} \in \mathbb{R}^{2c\times h\times w}$, where $i \in \{1, 2\}$ is a branch number. The 3×3 convolution layers are represented by $\emptyset_\nu^\alpha$, where α denotes the number of filters and ν is the padding and dilation number. Equation (4-1) describes the operation in the convolution branch, where ρ is the Rectified Linear Unit (ReLU). In this case, a compression ratio γ was set to 32, padding ν of 1 for the first branch (cb) and 24 for another (cbr).

$$\boldsymbol{y_i} = \text{Softplus}[\emptyset_\nu^{\alpha_1}\rho(\emptyset_\nu^{\alpha_2}(\boldsymbol{x^{ab}}))] \tag{4-1}$$

Where the number of filters α can be calculated by Eqs. (4-2) – (4-3), and γ denotes a compression ratio.

$$\alpha_1 = \frac{1}{\gamma}c \tag{4-2}$$

$$\alpha_2 = 2c \tag{4-3}$$

The results $\boldsymbol{y_i}$ are concatenated with $\boldsymbol{x^{ab}}$ before passing through the last convolution layer with $\hat{c}$ filters padding of 1 and the normalizer ξ. Equation (4-4) describes the calculation for the final output $\hat{\boldsymbol{y}}$ of this module, where $*$ is a concatenate operation.

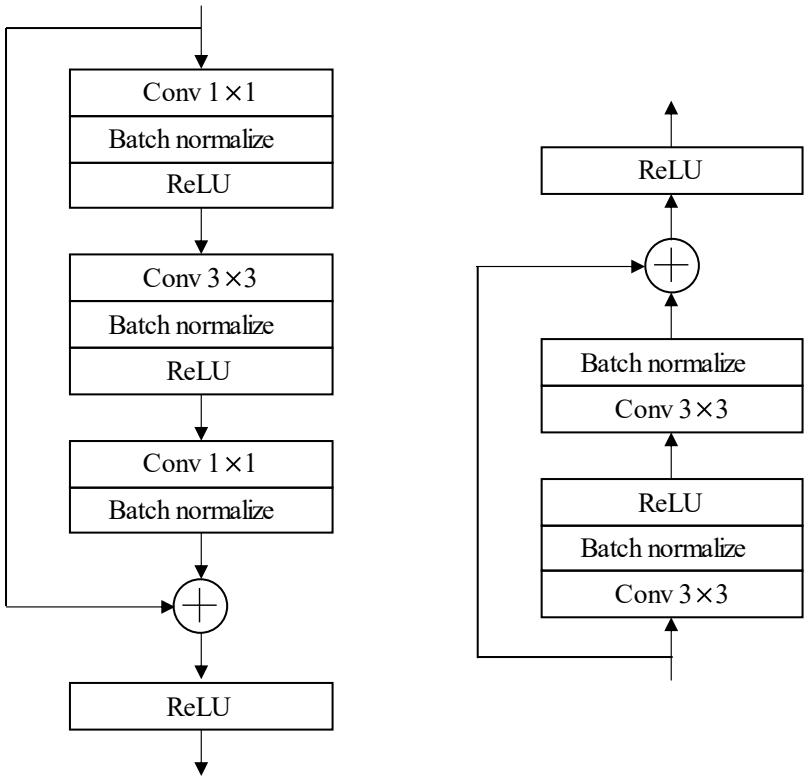
$$\hat{\boldsymbol{y}} = \xi[\emptyset_1^{\hat{c}}(\boldsymbol{y_1} * \boldsymbol{y_2} * \boldsymbol{x^{ab}})] \tag{4-4}$$

In this experiment, $\hat{c}$ value of the fused module used with the outputs of the ASPP module was set to 1024. Therefore, the output of this fusion module had a feature size of 1024 channels with the same feature dimension as the module's input. Using this proposed module could obtain a merged output from two results of two encoder branches. For the fused module of the skip feature in the bypass section, the number of convolution filters in the last layers was set as equal to the module's input channels. Other networks were trained with a concatenator as a fused module at the same position, and the skip feature maps were not operated in these networks.

d. Decoder

The decoder was an inverse direction of the encoder structure. It was developed from the ResNet-34 model but operated in the inverse direction of the original procedure. In this part, the feature map size was expanded by the transpose of the convolution filter.

The decoder begins with the upsample operation, the last block of ResNet-34. Unlike the encoder, a sequence of a 7×7 convolution layer and 3×3 max-pooling layer are removed at the last convolution layer of each layer. A transposed convolution operator substitutes for the convolution filter to double the size of the dimension. The next step was the decoder bottleneck, which consisted of three layers of a reverse bottleneck process. Figure 4-20 shows a comparison of the bottleneck between the encoder and decoder. The number of bottlenecks for each layer was {6, 4, 3}.



(a) (b)

Figure 4-20 Bottleneck diagrams; (a) encoder, (b) decoder

Reverse bottleneck no.1, named upsample 1, generated 512 channels of feature map and increased the feature map size to 1/8 of the original image. Reverse bottleneck no.2, named upsample 2, generated 256 channels of feature map and increased the feature map size to 1/4 of the original image. The last reverse bottleneck, named upsample 3, retained feature map channels but increased the feature map size to half of the original image. The output from the last layer was passed through the final convolution filter to generate the segmentation result image. This final filter reduced feature channels from 64 into the desired class, nine classes. The detail of each layer is explained in Table 4-10, where 3×3, 256 represents 256 convolution filters of 3×3 filter size. This decoder ended with a transposed convolution operator, which reduced the channel number into design classes and doubled the size of the dimension to the same as the input of the network. The fused skip features were added to the output of upsample 1 and upsample 2 in order, as shown in Figure 4-21. This approach computed three side outputs and one final output in the training mode. However, it generated only a final output in prediction mode. All outputs have different resolutions from the groundtruth map. Therefore, each output was resized with a bi-linear interpolation function to the exact resolution as a groundtruth map. Figure 4-22 shows the result feature maps from each decoder step; the bottom color images represent the side outputs. The final and side outputs are passed through the cross-entropy function to create the loss function.

Table 4-10 Detail of each layer in the decoder

Layer name	Output size	Convolution filter
upsample 1	$\frac{H}{8} \times \frac{W}{8}$	$\begin{bmatrix} 3\times 3, & 256 \\ 3\times 3, & 512 \end{bmatrix} \times 6$
upsample 2	$\frac{H}{4} \times \frac{W}{4}$	$\begin{bmatrix} 3\times 3, & 128 \\ 3\times 3, & 256 \end{bmatrix} \times 4$
upsample 3	$\frac{H}{2} \times \frac{W}{2}$	$\begin{bmatrix} 3\times 3, & 64 \\ 3\times 3, & 64 \end{bmatrix} \times 3$
final conv	$H \times W$	$1 \times 1, 9$
side conv		$1 \times 1, 9$

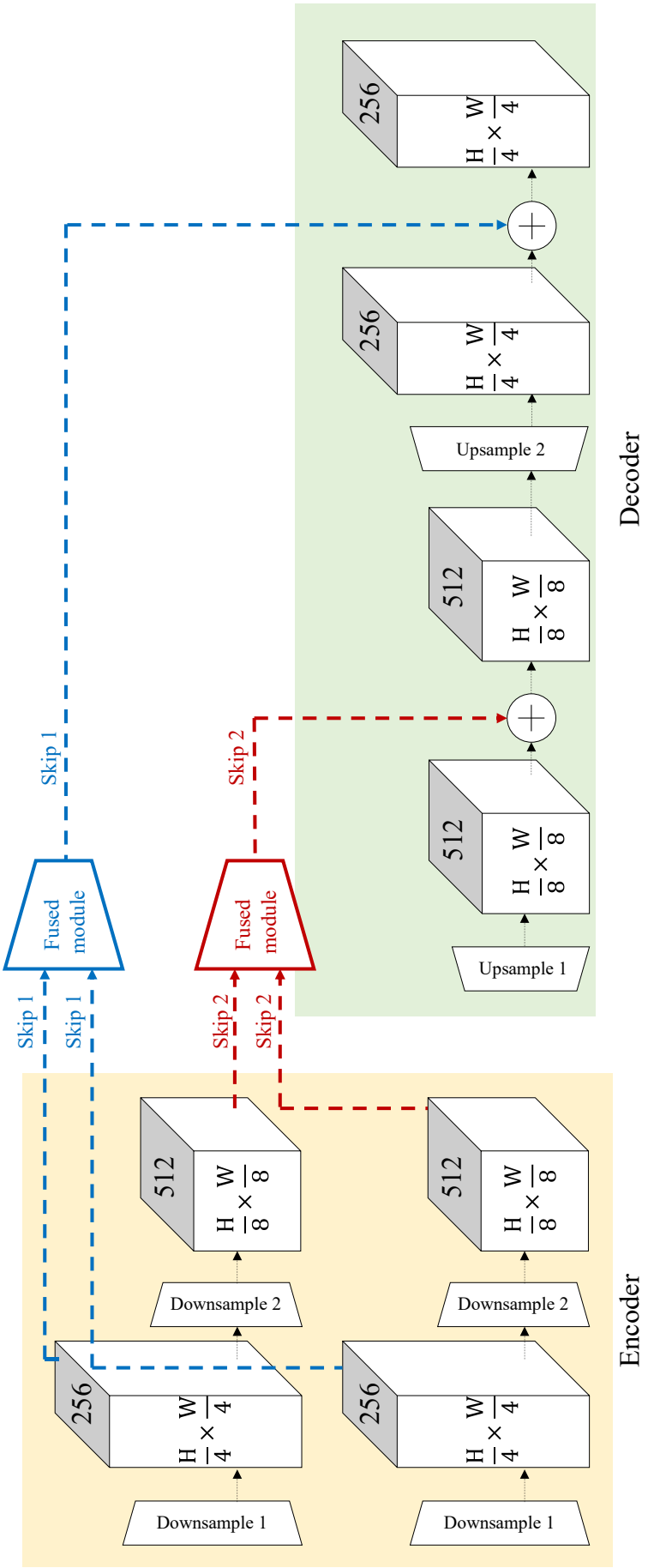


Figure 4-21 Skips flow diagram

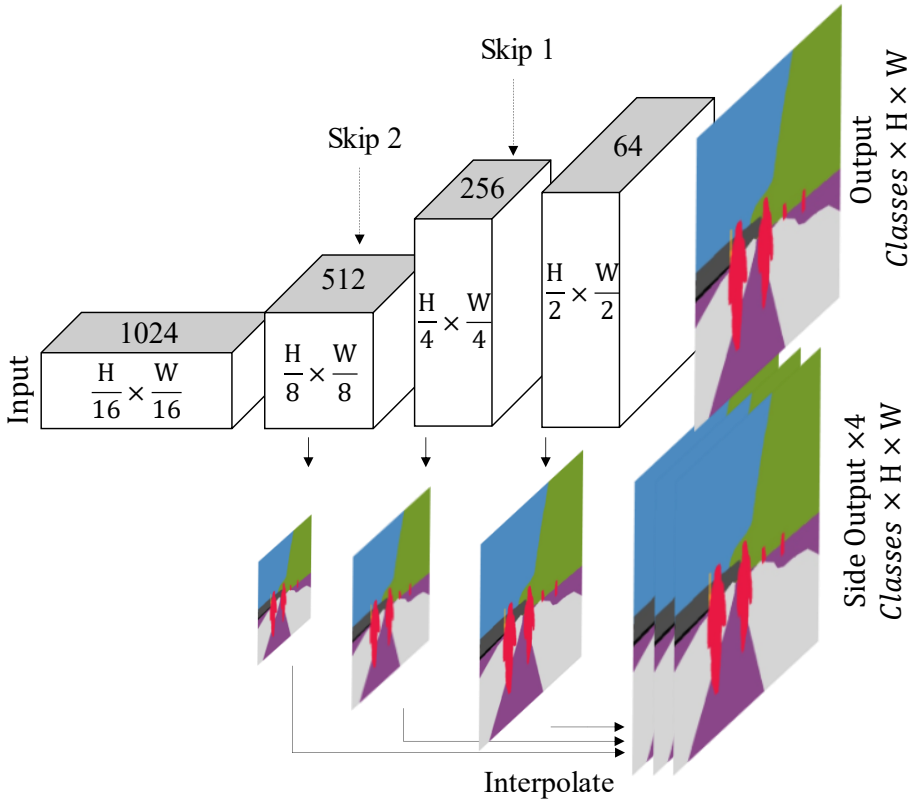


Figure 4-22 Feature map shape in the decoder

4.4 Training and Losses

The dataset was separated into three parts; training, validation, and evaluation datasets. The process was done on two platforms that specifications are shown in Table 4-11 and Table 4-12. The SSW, SRM-1, and SRM-2 were trained and tested on platform-1, and the other datasets were done on platform-2. First, the network structure was trained and compared to the other state-of-the-art networks, DANet, DUpsampling, ICNet, PSANet, PSPNet, DeepLabv3, and DeepLabv3 with SSMA. Then, each network was trained for 287,500 iterations or 200 epochs to decrease the convergent loss. All images in the dataset were resized from their original size into 640×480 pixels and then were fed through the model. The initial parameters for training are described in Table 4-13. The learning rate of this experiment was the same as in Section 3.3. In each training iteration, the learning rate can be plotted as shown in Figure 4-23.

Table 4-11 Training environment of platform-1

Equipment	Specifications
Central processing unit (CPU)	Intel® Core™ i7-4790 @ 3.60GHz
Random access memory (RAM)	16 GB
Graphics processing unit (GPU)	NVIDIA Geforce GTX™1080Ti
Operating System (OS)	Windows 10 Education N
Training framework	Pytorch 1.3

Table 4-12 Training environment of platform-2

Equipment	Specifications
Central processing unit (CPU)	Intel® Core™ i7-8700K @ 3.70GHz
Random access memory (RAM)	32 GB
Graphics processing unit (GPU)	NVIDIA Geforce GTX™1080Ti
Operating System (OS)	Ubuntu 18.04.6 LTS
Training framework	Pytorch 1.3

Table 4-13 Initial parameters

Parameters	Values
Platform	CUDA
Optimizer	Stochastic Gradient Descent (SGD)
Momentum	0.9
Initial learning rate ($rate_{ini}$)	0.001
weight decay	0.0001
$decay$	0.95
Loss function	cross-entropy function

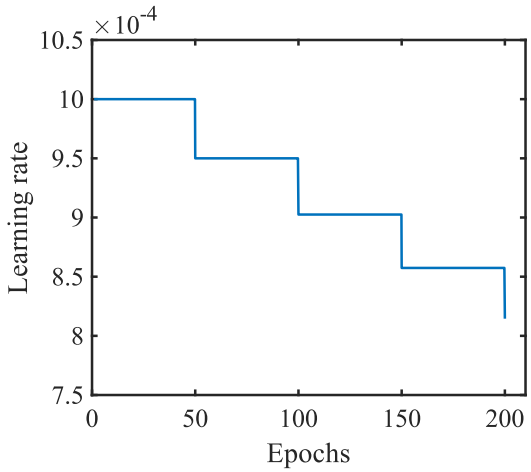


Figure 4-23 Learning rate (lr) at each epoch

Most of the models were calculated for losses by comparing final results and groundtruth, but the model introduced in Section 4.2 had four results in total. Hence, the average of all losses calculated the loss of this model. Figure 4-24 shows the training losses of the first 190,000 iterations from all models on the SRM-1 dataset. The black line in the graph represents the average loss of the introduced network with a pyramid supervision decoder. The loss of DeepLabv3+SSMA was lower than its original structure, DeepLabv3. The proposed network loss without a decoder was higher than both DeepLabv3 and DeepLabv3+SSMA. However, the proposed network with a decoder achieved the lowest loss value during training compared to other networks at the same training iteration. This loss was also lower than the DeepLabv3 loss, its original model structure. The difference between the proposed model with and without a decoder can be clearly shown in Figure 4-24. Therefore, using a decoder with pyramid supervision could enhance loss value during the training process.

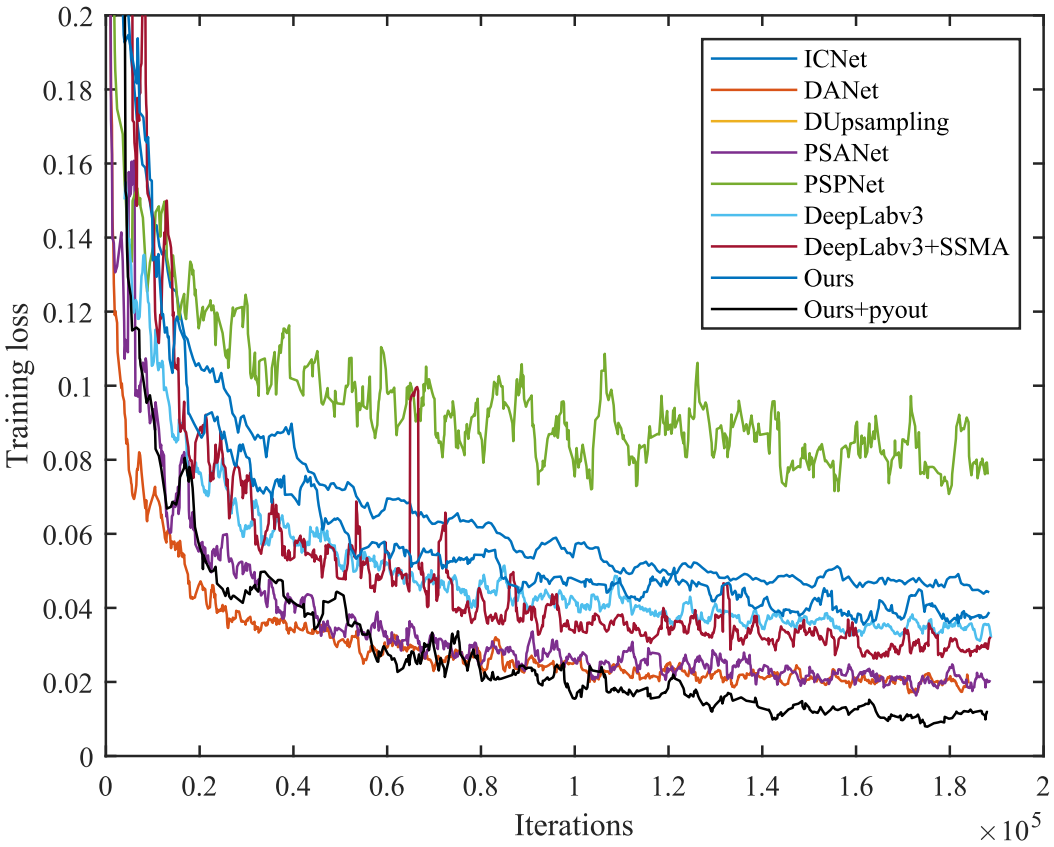


Figure 4-24 Training loss

Figure 4-25 shows the feature map in the encoder part step-by-step. The large top images are input images, the RGB image and the Thermal map. The center column images are the output feature maps of the fused module. The bottom image of the left and right columns is the output of ASPP modules. The most bottom image is the output fed to the decoder. Figure 4-26 shows the feature map in the decoder part step-by-step. The tiny top image is a feature map that is an output of Figure 4-25. The left and right images are the skip feature maps from the center column of Figure 4-25. The large bottom image is a result feature map or segmentation image that contains nine channels of the same size as the original image.

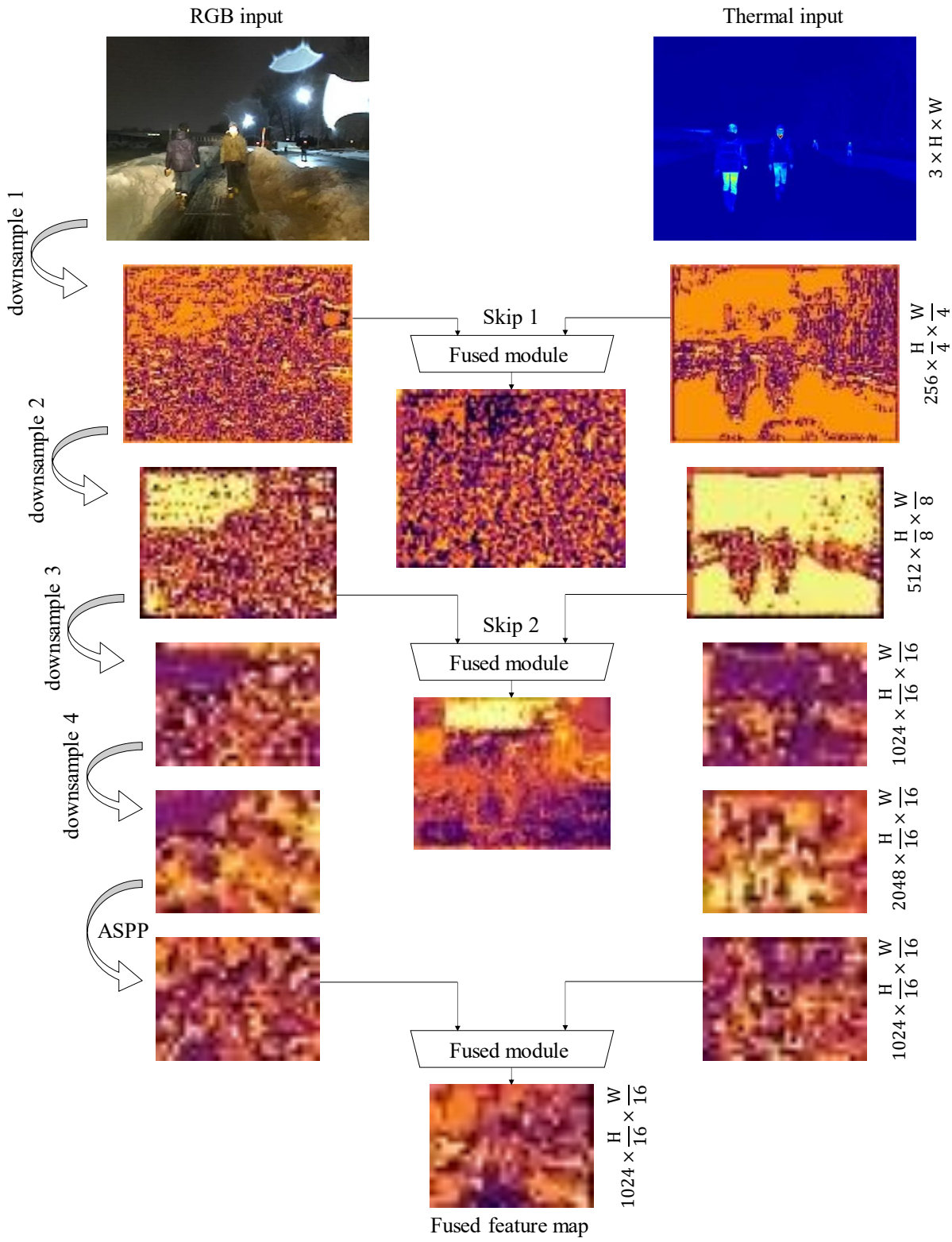


Figure 4-25 Feature map in the encoder, ASPP module, and Fused module

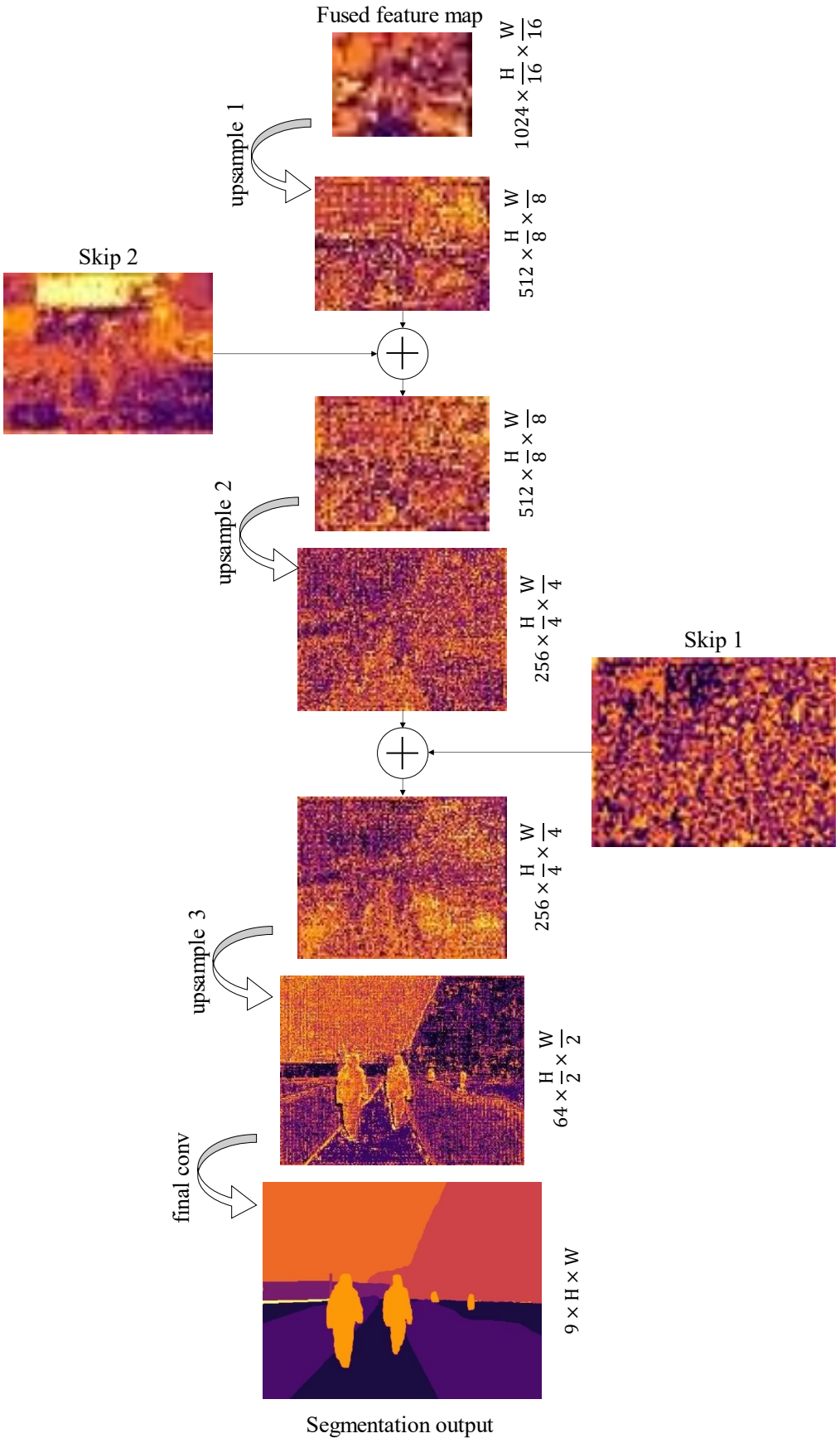


Figure 4-26 Feature map in the decoder

4.5 Results & Evaluation

The semantic segmentation problem based on deep learning could be considered as a pixel classification task. Every model was evaluated by employing the Intersection-over-Union (IoU) and Accuracy (Acc) as indicators to explain the similarity between the segmentation result and groundtruth map. The IoU and Acc were calculated for every class in the result image. Therefore, the mean-IoU (mIoU) and mean-Acc (mAcc) of the image were calculated to represent the overall efficiency of each network. The evaluation was done by feeding a test image into the model one-by-one. Some of the results [2] were published at the 2021 IEEE International Conference on Mechatronics and Automation (ICMA), and some [1] were published in IEEE Sensors Journal.

a. Combination of the compression ratio and the padding number

There were two main parameters in the fused module that could affect the efficiency of the network: the compression ratio (γ) was used to reduce the size of the channels in each branch, and the padding number (ν) was used to capture the multi-scale context in a feature map. The fused module was tested for the most suitable combination of γ and ν in a snowy environment, the SRM-1 dataset. The values of the compression ratio were varied by a power of two and multiple of six for the padding and dilation value. Table 4-14 shows the mIoU of each combination of γ and ν on the SRM-1 dataset. The results showed that the best combination for a snowy environment is γ = 32 and ν = 24.

Table 4-14 The mIoU of each combination on the SRM-1 dataset

Padding number (ν)	Compression ratio (γ)	mIoU (%)
6	32	70.9
12	32	72.6
18	32	71.2
24	**32**	**73.2**
30	32	71.8
36	32	72.0
24	2	72.5
24	4	72.3
24	8	71.1
24	16	70.4
24	**32**	**73.2**

b. The ablation study of the fused module

The ablation study of the proposed fused module was done on the SRM-1 dataset. The network analyzed in this experiment consists of the encoder, ASPP module, and fused module. This study was tested on seven combinations of the fused module: cb, cbr, skip, cb+cbr, cb+skip, cbr+skip, and cb+cbr+skip (the proposal). Where cb denotes a convolution branch, cbr denotes a convolution with a rate branch, and skip is a skip branch. Figure 4-27 shows the structure of each branch. The IoU of the pedestrian class and mean IoU were used as indicators to evaluate the module. Table 4-15 shows the results of the ablation test of the fused module on the SRM-1 dataset. The results showed that using double branches (cb+cbr) was better than only a single branch (cb or cbr) for the mIoU, and the utilization of a skip branch (skip) could improve the pedestrian IoU and mIoU of each convolution branch. The whole combination of each fused module branch reached the highest of every condition in both the pedestrian's IoU and mIoU.

Table 4-15 The ablation test of the fused module on the SRM-1 dataset

Fused module architecture	Human IoU (%)	mIoU (%)
cb + cbr + skip	**75.8**	**73.2**
cb + cbr	73.8	71.9
cb + skip	73.5	72.3
cbr + skip	74.0	72.0
cb	74.0	71.3
cbr	72.7	71.4
skip	73.6	72.3

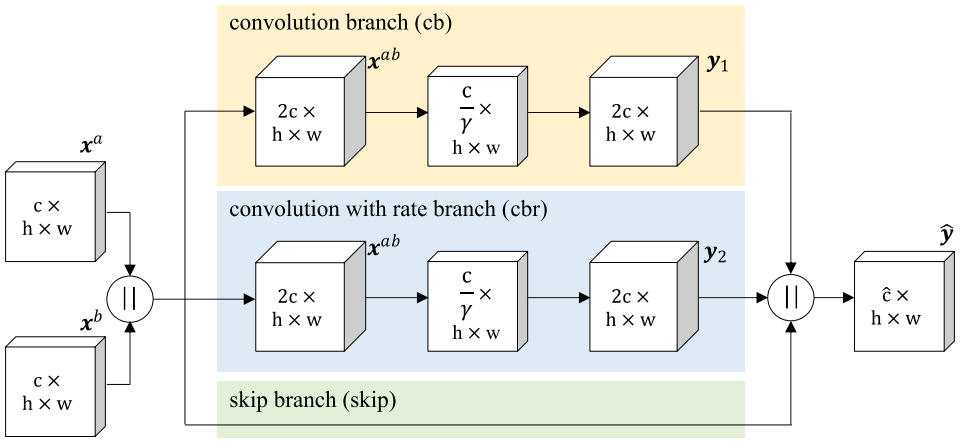


Figure 4-27 Each branch of the fused module

c. The combination of merging operators in the fused module

There are two merging operators used in general. First, the add-operator, symbol as +, is famous for combining two or more feature maps. This operator creates the same shape, c×h×w, as the initial feature maps. Another is the concatenation operator, symbol as ||, working by sticking all feature maps into one feature map. Unlike other operators, the concatenated result has the same feature map size as the initial one but a different channel number. The channel number of this result is from the summation of all initial feature map channels.

The fused module combined two or more feature maps into a new feature map. In the module, there were two locations of the fusing point. The first location was at the input point. The two input feature maps were merged at this point. Another location was after the convolution branch. It was used to merge three feature maps processed in the module. Figure 4-28 shows the merging location in the fused module. The fused module was tested for the most suitable combination of merging operators on the SRM-2 dataset. The IoU of the pedestrian class and mean IoU were used as indicators to evaluate the module. Table 4-16 shows the results of each combination on the SRM-2 dataset. The results showed that the || & + was the best for human recognition. However, the || & || was the best in overall performance. Therefore, the || & || was chosen in the other experiments.

Table 4-16 Operators combination of the fused module on the SRM-2 dataset

Operator 1	Operator 2	Human IoU (%)	mIoU (%)
+	+	80.2	56.6
+	||	81.8	59.9
||	+	83.6	62.8
||	||	**82.8**	**64.6**

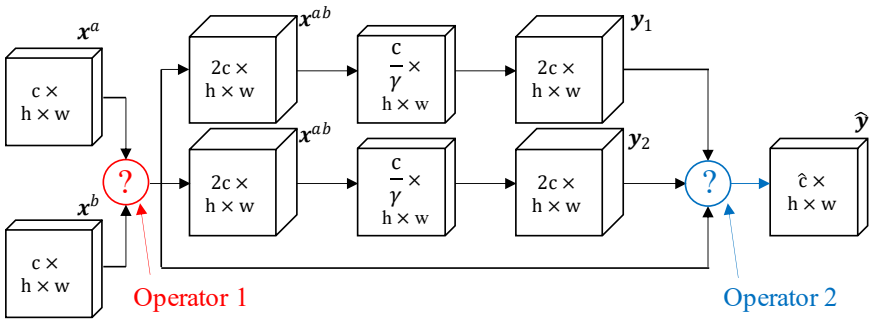


Figure 4-28 Operators in fused module

d. The combination of activation functions in the fused module

Chapter 4.2-c described the process done in the fused module. Each branch of the module, cb and cbr, consisted of the convolution filter and the activation function. This experiment introduced two main activation functions, the ReLU and the Softplus. The ReLU is famous for use in the neural network and is also used in this module. Equation (2-28) described the ReLU equation. Another function, Softplus, was used in only the module part of the overall structure. This activation function characteristic is smoother than the ReLU. Equation (2-30) described the Softplus equation.

Each branch of the fused module consisted of two activation functions, one after the squeeze convolution filter and another after the expanded convolution filter. This experiment was tested to find the best combination of activation functions used in the fused module. This experiment tested four combinations: ReLU & Softplus, ReLU & ReLU, Softplus & Softplus, and Softplus & ReLU. The convolution branches of each combination, except ReLU & Softplus (the proposal), are calculated by Eqs. (4-5), (4-6), and (4-7) in order. Equation (4-1) shows the calculation of the proposal. Table 4-17 shows the results of each combination on the SRM-2 dataset. The results showed that the ReLU&Softplus was the best in all combinations in the fused module. Moreover, using Softplus as the activation function could improve the IoU of the human class.

$$\boldsymbol{y_i} = \rho[\emptyset_v^{\alpha_1}\rho(\emptyset_v^{\alpha_2}(\boldsymbol{x^{ab}}))] \tag{4-5}$$

$$\boldsymbol{y_i} = \mathrm{Softplus}[\emptyset_v^{\alpha_1}\mathrm{Softplus}(\emptyset_v^{\alpha_2}(\boldsymbol{x^{ab}}))] \tag{4-6}$$

$$\boldsymbol{y_i} = \rho[\emptyset_v^{\alpha_1}\mathrm{Softplus}(\emptyset_v^{\alpha_2}(\boldsymbol{x^{ab}}))] \tag{4-7}$$

Table 4-17 Activation function combination of the fused module on the SRM-2 dataset

1st function	2nd function	Human IoU (%)	mIoU (%)
ReLU	**Softplus**	**82.8**	**64.6**
ReLU	ReLU	81.6	63.7
Softplus	Softplus	83.1	63.0
Softplus	ReLU	82.9	63.3

e. The ablation study of Skip in the overall architecture

In the proposed network structure, two skip feature maps were introduced. Skip-1 was created by fusion feature maps resulting from downsample-1 layers. This Skip had 256 channels and the size of 1/4 of the original image. It was combined with the result of the upsample-2 layer. Same as Skip-1, Skip-2 was the result of the downsample-2 layers. It had the shape of 512 channels and 1/8 of the original image size. This feature map was added to the result of the upsample-1 layer. Figure 4-29 shows both Skip-1 and Skip-2 in the overall structure.

The ablation study of the Skip was done on the SRM-1 dataset. The network analyzed in this experiment consists of the encoder, ASPP module, fused module, and decoder. This study was tested on four combinations of the Skip combination: without Skip, Skip-1, Skip-2, and Skip-1 + Skip-2 (the proposal). The IoU of the pedestrian class and mean IoU were used as indicators to evaluate the module. Table 4-18 shows the results of the ablation test of Skip on the SRM-1 dataset. The results showed that Skip-1 + Skip-2 was the best of all combinations.

Table 4-18 The ablation test of the Skip on the SRM-1 dataset

Overall architecture	Human IoU (%)	mIoU (%)
Skip-1 + Skip-2	**82.8**	**79.0**
Skip-1	80.6	77.1
Skip-2	81.4	78.4
no Skip	81.1	78.0

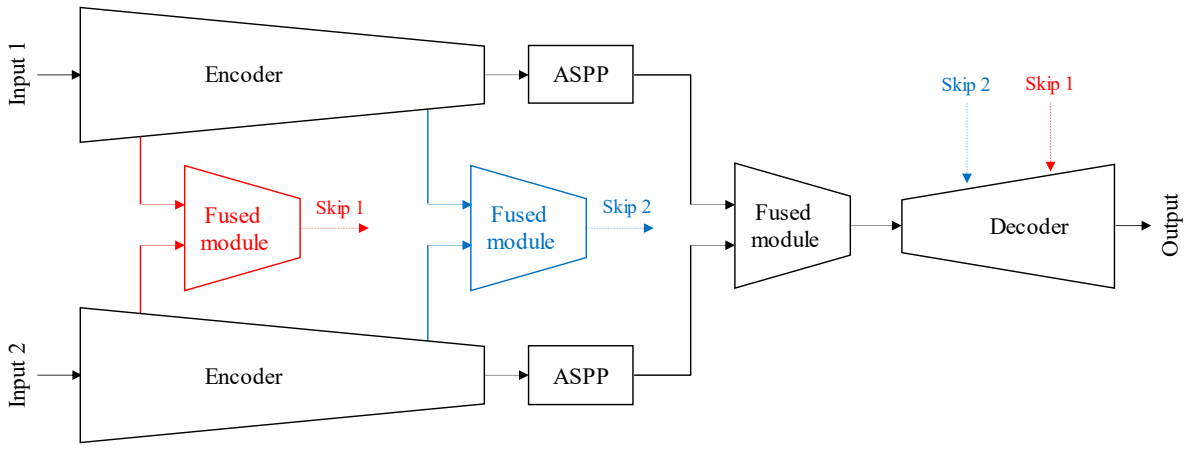


Figure 4-29 Skip in the overall network structure

f. Colormap of mono-information input

The multiple inputs experiment used a Depth map or Thermal map as a second system input. These feature maps were generated from the sensor that measured only a feature, such as distance or temperature. Thus, the original maps were represented as a grayscale image. However, the network encoder needs a feature map with three channels for the input. Therefore, the single-channel feature map was converted into three channels feature map. There were two kinds of feature maps introduced in this experiment. The first feature map was extended from one channel to three channels with the same colormap, grayscale. The three channels grayscale feature map had the same value in all channels. Another was the jet colormap, as described in Section 4.1-f. This colormap contained three channels that could represent the difference in information. Figure 4-30 shows the example input of both types of feature maps.

This experiment compared the efficiency of different types of colormap representing the input feature map. The IoU of the pedestrian class and mean IoU were used to evaluate the input efficiency. The evaluation was done on the original DeepLabv3 model to neglect the effect of the other parameters. Table 4-19 shows the results of each input colormap on the SRM-1 dataset. The results show that the jet colormap result on human detection was obviously higher than the grayscale one.

Table 4-19 Colormap comparison efficiency of DeepLabv3 on SRM-1 dataset

Input 1	Input 2	Human IoU (%)	mIoU (%)
Image (RGB)	**Thermal (jet)**	**72.9**	**70.2**
Image (RGB)	Thermal (grayscale)	29.0	67.4

(a) (b)

Figure 4-30 Thermal map; (a) grayscale colormap, (b) jet colormap

g. Thermal map input segmentation in various light conditions

The various brightness of the light directly affected the performance of segmentation. This experiment used a thermal camera to due with the problem. It was focused on the different light conditions of input images, the daytime and nighttime. The dataset used in this section was SRM-1, which consisted of a snowy road environment during daytime and nighttime. In both single and double inputs, the experiment was tested on the standard networks: DANet, DUpsampling, ICNet, PSANet, PSPNet, and DeepLabv3. All network structures in this experiment could be separated into two parts: the encoder and the head module. Figure 4-31 shows the diagram of the network structure. The networks contained one encoder per one input feature. Therefore, the multiple inputs segmentation networks contained two parallel encoders, as shown in Figure 4-32. The feature map from the encoder was fed into the head module. In this module, the feature maps were computed in various operations due to the different networks. Before the last layer in the head module, the feature maps were merged by the concatenation operator, as shown in Figure 4-33. The ICNet had its encoder different from other networks, but the head module modified the same concept. In this experiment, the operation in both branches was done separately and combined at the end of the network. The dual inputs network combines each processed feature with a concatenation operator; this fusion is done without an additional process for merging. These make the network not recognize the object with slightly different temperatures compared to the outstanding thermal source.

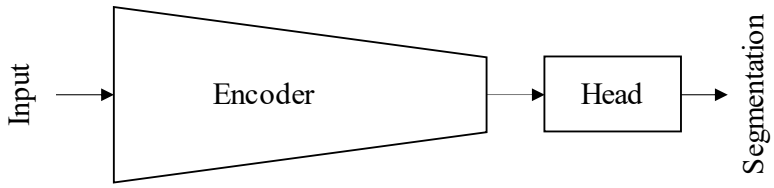


Figure 4-31 The network structure for single input

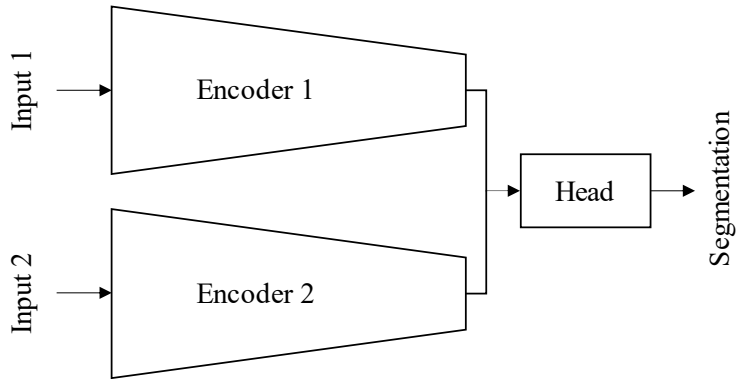


Figure 4-32 The network structure for double inputs

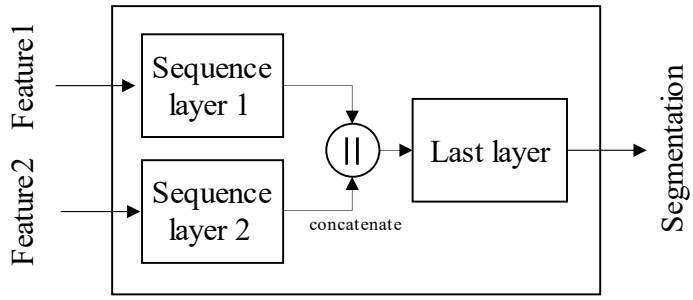


Figure 4-33 Head module of double inputs network

Table 4-20 shows the results in daytime and nighttime on the RGB part, and Table 4-21 shows the results in the RGB-T part. Results showed that the IoU of the pedestrian class was mainly improved by an average of 17.8% in the nighttime when used thermal input was a secondary input. The DUpsampling was also enhanced by a Thermal map in both daytime and nighttime. The human IoU of both RGB and RGB-T parts in the nighttime was higher than daytime; the different average in RGB is 4.1% and 20.6% in RGB-T. Therefore, using a Thermal map could improve human detection from single input segmentation, especially at nighttime. The difference in temperatures between humans and their surroundings was higher during the nighttime. The difference in human segmentation was outstanding in the DUpsampling network. It could recognize humans in detail better than using an only single input. However, there was a slight change in human class in the others described in Table 4-20 and Table 4-21. The vehicle segmentation results on the RGB-T part were lower than the single image dataset because the vehicle in this dataset was an un-started car. The temperature of the vehicle was not too different from the surrounding temperature.

Table 4-20 The segmentation results on RGB part of SRM-1 dataset

ID	Period	IoU (%)									mIoU (%)
		1	2	3	4	5	6	7	8	9	
ICNet	Day	85.1	91.8	85.6	51.3	92.2	94.8	56.4	50.4	66.4	74.9
	Night	86.7	92.0	81.8	5.4	93.3	94.7	57.9	-	53.6	70.7
DUpsampling	Day	81.3	89.3	73.3	0.3	90.9	96.0	0.3	0.7	0.0	48.0
	Night	82.7	90.8	72.7	0.2	93.4	96.5	3.0	-	0.0	54.9
DANet	Day	89.6	94.7	89.1	62.5	95.3	97.2	62.5	48.6	71.6	79.0
	Night	92.0	98.9	87.6	14.7	96.0	97.0	68.5	-	58.6	76.7
PSPNet	Day	88.9	94.2	88.3	61.6	95.1	97.0	60.4	48.5	72.0	78.4
	Night	91.1	94.3	85.9	12.1	95.6	96.8	65.7	-	58.1	75.0
PSANet	Day	89.1	94.4	88.7	65.9	95.3	97.1	64.2	48.6	70.3	79.3
	Night	91.8	94.7	87.9	20.6	96.0	97.0	69.5	-	57.7	76.9
DeepLabv3	Day	88.9	94.1	88.1	60.7	94.9	97.0	59.7	54.4	72.0	78.9
	Night	91.3	94.5	86.0	12.1	95.7	96.9	63.3	-	54.3	74.3

Table 4-21 The segmentation results on RGB-T part of SRM-1 dataset

ID	Period	IoU (%)									mIoU (%)
		1	2	3	4	5	6	7	8	9	
ICNet	Day	82.5	91.4	79.3	44.0	89.4	92.1	34.9	21.3	60.3	66.1
	Night	84.6	66.9	71.4	5.8	84.4	17.6	74.2	-	8.0	51.6
DUpsampling	Day	82.3	89.3	78.7	0.1	92.3	93.0	53.8	1.9	56.8	60.9
	Night	86.6	91.4	83.2	0.2	82.9	94.3	67.1	-	21.6	65.9
DANet	Day	82.2	89.3	85.0	50.0	93.2	95.4	58.3	31.3	64.6	72.1
	Night	89.0	93.2	86.3	13.6	83.4	87.5	76.2	-	15.8	68.1
PSPNet	Day	85.6	91.8	83.1	47.6	92.3	93.6	61.4	29.5	64.5	72.2
	Night	89.5	87.9	84.7	13.5	88.4	84.8	77.6	-	34.3	70.1
PSANet	Day	79.6	86.7	81.1	49.8	91.0	93.6	42.8	22.0	58.5	67.2
	Night	85.5	90.3	79.5	19.8	85.6	90.6	63.1	-	28.1	67.8
DeepLabv3	Day	82.1	88.5	82.3	44.7	92.6	95.4	59.8	25.7	65.9	70.8
	Night	88.3	92.5	82.9	8.8	93.6	95.3	76.6	-	46.2	73.0

h. Comparison of RGB and RGB-T on the Human recognition

The primary purpose of the segmentation work was to use in the snow removal operation. In the snow remover machine operating site, the most important object to detect was the pedestrian or human. However, human detection in the snowy low-light environment was complicated, using only an ordinary sensor or an RGB camera. Therefore, the Thermal map was introduced to improve this issue.

The Thermal map provided excellent information on heat reflected from objects, especially when used for discovering humans. In a snowy environment, the temperature difference between the surrounding environment and a human could be clearly represented in the Thermal map. The further use of this heat information could improve human detection efficiency. The efficiency of using the Thermal map as a second input is described in Table 4-22. This experiment was the comparison between using the RGB input *with* and *without* Thermal map input on the SRM-1 dataset.

Table 4-22 The comparison of using the Thermal map in the SRM-1 dataset

Network	Human IoU (%)	
	RGB	RGB-T
ICNet	57.6	66.1
DANet	67.2	72.4
DUpsmapling	2.3	64.2
PSANet	68.4	58.9
PSPNet	64.5	74.1
DeepLabv3	64.9	72.9
DeepLabv3 + SSMA	-	74.8
Ours	**-**	**75.8**
Ours + pyout	**-**	**78.4**

The test was focused on the IoU of humans, which was called human detection, and tested on many state-of-the-art networks. The results showed that using a Thermal map could improve the efficiency of human detection for most networks. The proposed fused module could reach the highest IoU in the human class, and the application of the decoder could be higher than the previous one. Figure 4-34 shows examples of segmentation errors in human detection. The green area is the pixel difference of the segmentation results from groundtruth (GT). The network using the RGB-T inputs obtained a smaller error for human detection than the best state-of-the-art network using only RGB images as the input.

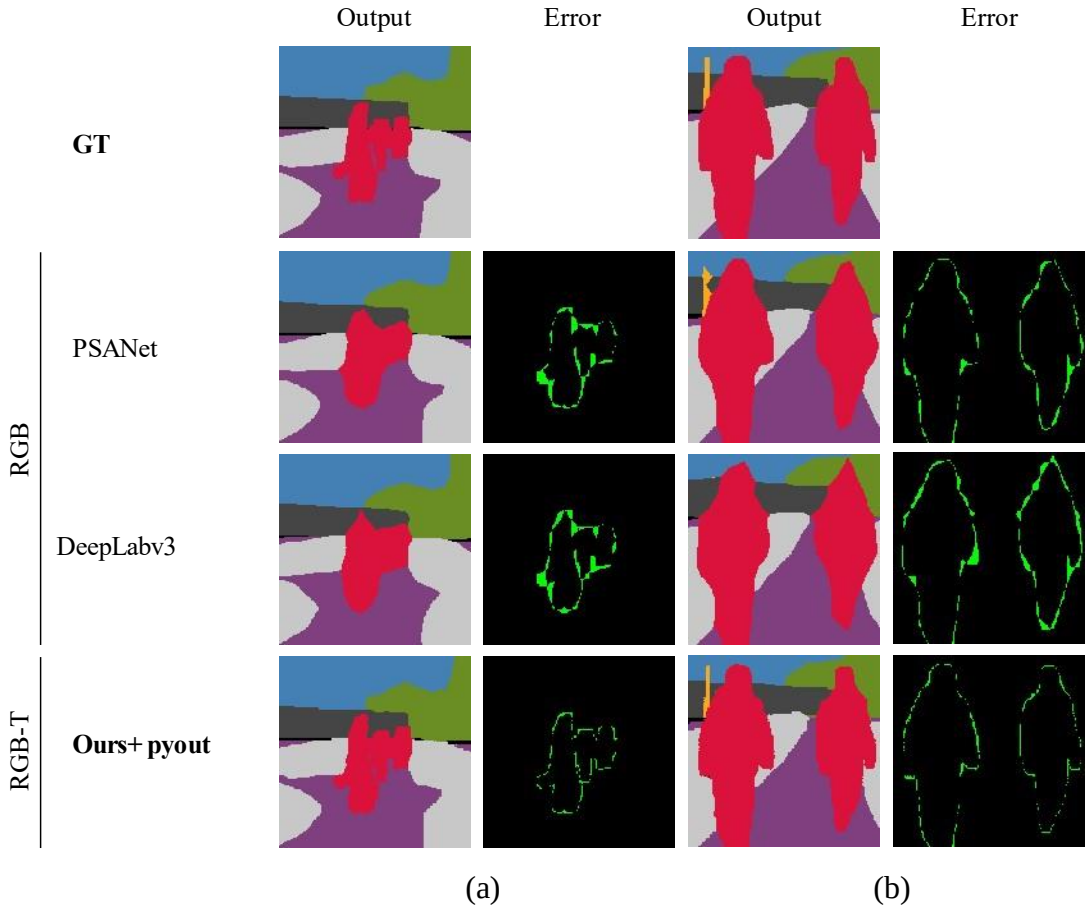


Figure 4-34 Segmentation errors in human detection;
(a) Daytime environment, (b) Nighttime environment

i. The efficiency of the Fused module

The fused module was developed to merge double feature maps into a new feature map. This module could be applied to many network structures by placing the module in the head module, as shown in Figure 4-35. First, the feature map from each branch was fed into the head module, then passed through the sequence layer, different in each network. Next, the fused module merged the output of the sequence layer of each branch. Finally, the merged feature map was passed through the last layer for classification. This experiment compared the performance of the fused module, Figure 4-35, with the concatenation operator, Figure 4-33.

Table 4-23 shows the improved result of each network structure on the SRM-2 dataset. The positive values represent that using the fused module is better than concatenating. The results showed that most of the networks that contained the fused module reached better results in IoU and mIoU. PSANet was the most improved network by fused module. It was enhanced up to about 8.3% IoU on average. Most of the networks, except DUpsampling, were outstandingly improved, up to 5.2% IoU on average, in the vegetation recognition by the fused module. From the overall results could be concluded that the use of a fused module could enhance the segmentation performance.

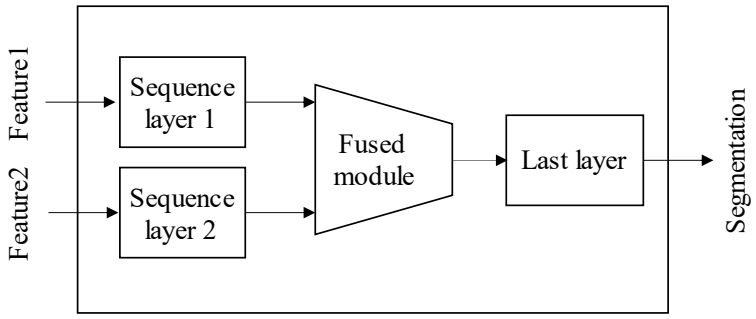


Figure 4-35 Head module of double inputs network with fused module

Table 4-23 The improved results of use of fused module on SRM-2 dataset

ID	IoU (%)									mIoU (%)
	1	2	3	4	5	6	7	8	9	
ICNet	3.0	-1.1	-2.2	-5.6	4.5	5.8	-3.8	2.8	1.0	0.5
DANet	1.0	2.5	1.6	-0.9	4.6	0.1	-2.3	0.3	-2.6	0.5
DUpsampling	0.6	2.7	0.7	3.2	-3.9	-0.2	6.7	0.5	-3.2	0.8
PSANet	5.2	14.0	6.5	13.1	11.0	2.9	12.9	0.5	8.5	8.3
PSPNet	0.4	-1.5	-0.5	0.9	1.1	-0.3	-0.6	0.9	3.0	0.4
DeepLabv3	0.0	0.2	2.2	4.1	4.9	-0.4	0.8	0.0	1.0	1.4

j. The segmentation performance

The main objective of this research is to develop a network that works well with the snow remover machine. A snow remover machine is a truck-like machine used to break the packed snow on the road surface and then throw it elsewhere. The operation of this machine is usually done at the nighttime due to the road traffic. Therefore, the main environment that the system has to process is the snowy road of the city in the nighttime.

All of the networks in this experiment were evaluated on the multiple inputs datasets. The inputs of every network consisted of an RGB image and another feature map, a Thermal map or Depth map. The proposed network was developed as a multiple-input structure that is suitable for the experiment. The other networks in this experiment were managed following the experiment in Section 4.4-g, as shown in Figure 4-32. The experiments would be evaluated on the SSW, SRM-1, and SRM-2 datasets as the primary dataset for the snowy environment. The other two datasets, the Cityscape and the Synthia, were the support experiment due to the information in the dataset. The Cityscape and the Synthia dataset did not contain the natural snowy environment. Furthermore, they did not include the Thermal map, which was the main objective of the research. However, they contained other information, the Depth map, from the RGB image. Therefore, they were used to confirm that the introduced network structure obtained the best efficiency in multiple-input segmentation.

Table 4-24, Table 4-25, and Table 4-26 show the IoU of each class on the RGB-T of the SSW, SRM-1 and SRM-2 datasets. The results show that the proposed network, which used fused modules, could obtain a higher mIoU than other networks, and with the inverse ResNet-34 decoder, it could reach the highest IoU in all classes. The mIoU of the best network was about 12.6% higher than the other networks on the SSW dataset, 12.8% on the SRM-1 dataset, and 9.4% on the SRM-2 dataset. The results show that the proposed network becomes state-of-the-art for segmentation in snowy environments. Figure 4-36, Figure 4-37, and Figure 4-38 show the results of segmentation in every RGB-T dataset. To ensure the efficiency of the approach, the proposed multi-modal fused network on other published datasets and tested for mAcc and mIoU was evaluated. Table 4-27 and Table 4-28 show the results of mAcc and mIoU on the other two datasets, the Synthia and the Cityscapes. The results show that the fused module could improve the efficiency of multiple inputs of the neural network and train using a pyramid supervision path to generate the highest values for every indicator. Figure 4-39 and Figure 4-40 show the segmentation results in the RGB-D dataset.

Table 4-24 The segmentation results on SSW dataset

ID	IoU (%)									mIoU(%)	mAcc(%)
	1	2	3	4	5	6	7	8	9		
ICNet	89.4	87.4	88.3	8.2	89.5	92.7	79.7	79.7	24.9	71.1	98.6
DANet	88.2	86.9	87.2	16.1	86.4	88.1	82.8	78.9	26.2	71.2	98.4
DUpsampling	87.3	83.7	80.2	0.5	86.9	90.5	1.5	22.2	0.0	50.3	97.9
PSANet	89.0	87.4	87.5	18.0	87.7	89.7	85.8	70.2	19.4	70.5	98.5
PSPNet	89.5	87.5	88.5	18.8	86.7	87.8	86.1	83.3	32.1	73.4	98.5
DeepLabv3	88.5	86.6	81.6	17.2	83.6	88.7	85.5	81.8	30.6	71.6	98.3
DeepLabv3 + SSMA	91.9	90.5	89.5	14.1	87.7	89.5	84.5	74.7	21.3	71.5	98.7
Ours	**88.0**	**86.5**	**88.5**	**18.2**	**87.0**	**89.2**	**88.9**	**85.3**	**32.2**	**73.8**	**98.5**
Ours + pyout	**94.1**	**93.5**	**92.2**	**33.8**	**93.8**	**95.3**	**93.7**	**84.5**	**49.1**	**81.1**	**99.2**

1:Road, 2:Snow mountain, 3:Building, 4:Traffic object, 5:Vegetation, 6:Sky, 7:Human, 8:Vehicle, 9:Unidentified

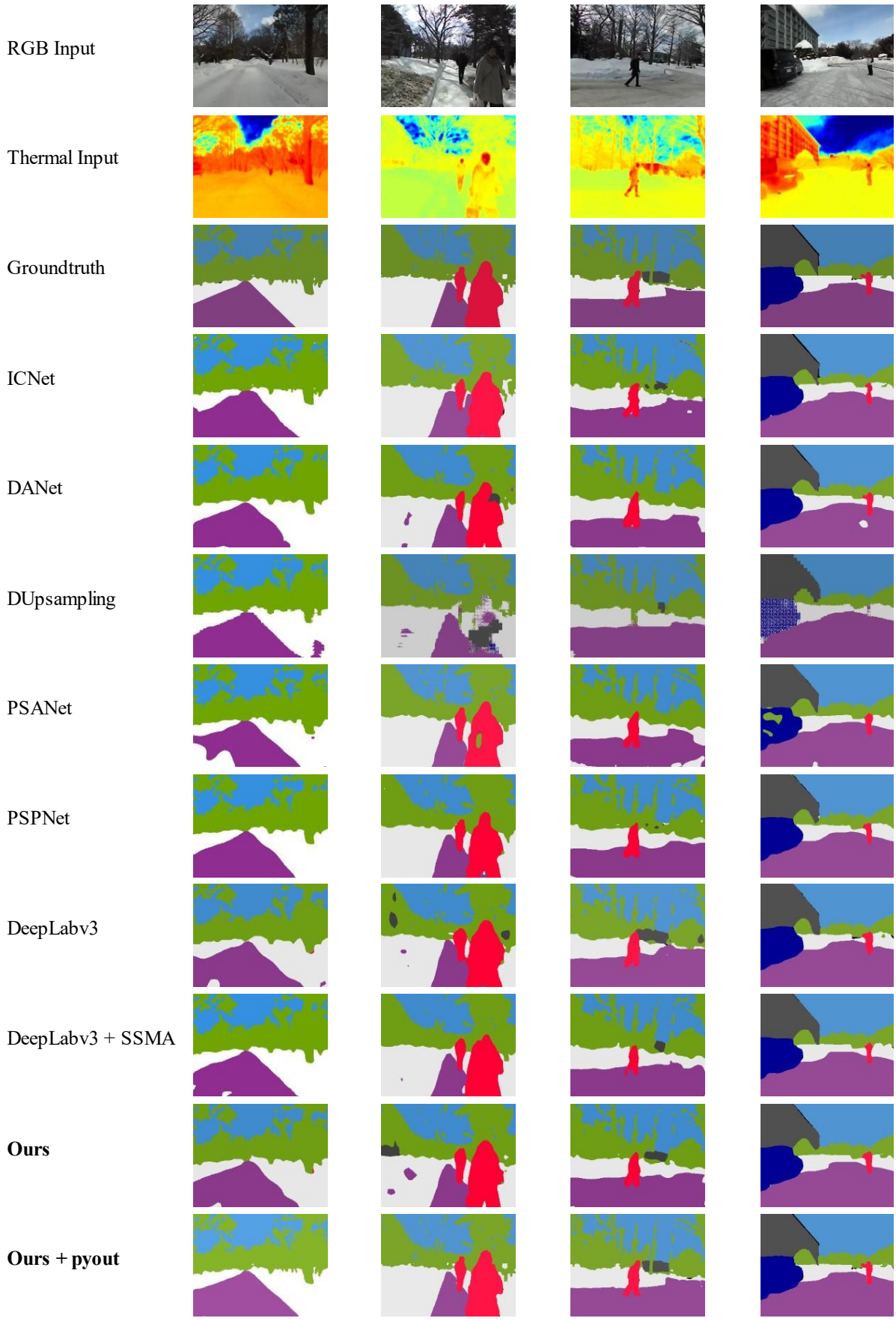


Figure 4-36 Example of segmentation results on SSW dataset

Table 4-25 The segmentation results on SRM-1 dataset

ID	IoU (%)									mIoU(%)	mAcc(%)
	1	2	3	4	5	6	7	8	9		
ICNet	83.2	77.0	74.9	24.1	86.6	54.1	66.1	11.7	22.4	55.6	96.1
DANet	85.4	91.3	85.7	37.7	87.7	91.2	72.4	18.9	35.9	67.4	98.2
DUpsampling	84.2	90.4	81.0	0.1	87.0	93.7	64.2	0.9	39.9	60.2	98.2
PSANet	82.4	88.5	80.3	38.7	87.9	91.8	58.9	16.2	47.2	65.8	98.2
PSPNet	87.3	89.8	83.9	34.6	90.1	88.9	74.1	20.7	52.6	69.1	98.4
DeepLabv3	85.0	90.6	82.6	33.4	93.2	95.0	72.9	19.8	59.4	70.2	98.7
DeepLabv3 + SSMA	85.7	91.4	83.9	36.7	92.5	95.3	74.8	26.6	53.7	71.2	98.7
Ours	**87.5**	**92.7**	**85.4**	**36.4**	**94.4**	**95.9**	**75.8**	**27.5**	**63.3**	**73.2**	**98.9**
Ours + pyout	**91.8**	**95.5**	**91.4**	**53.5**	**95.6**	**96.5**	**81.8**	**31.9**	**67.4**	**78.4**	**99.2**

1:Road, 2:Snow mountain, 3:Building, 4:Traffic object, 5:Vegetation, 6:Sky, 7:Human, 8:Vehicle, 9:Unidentified

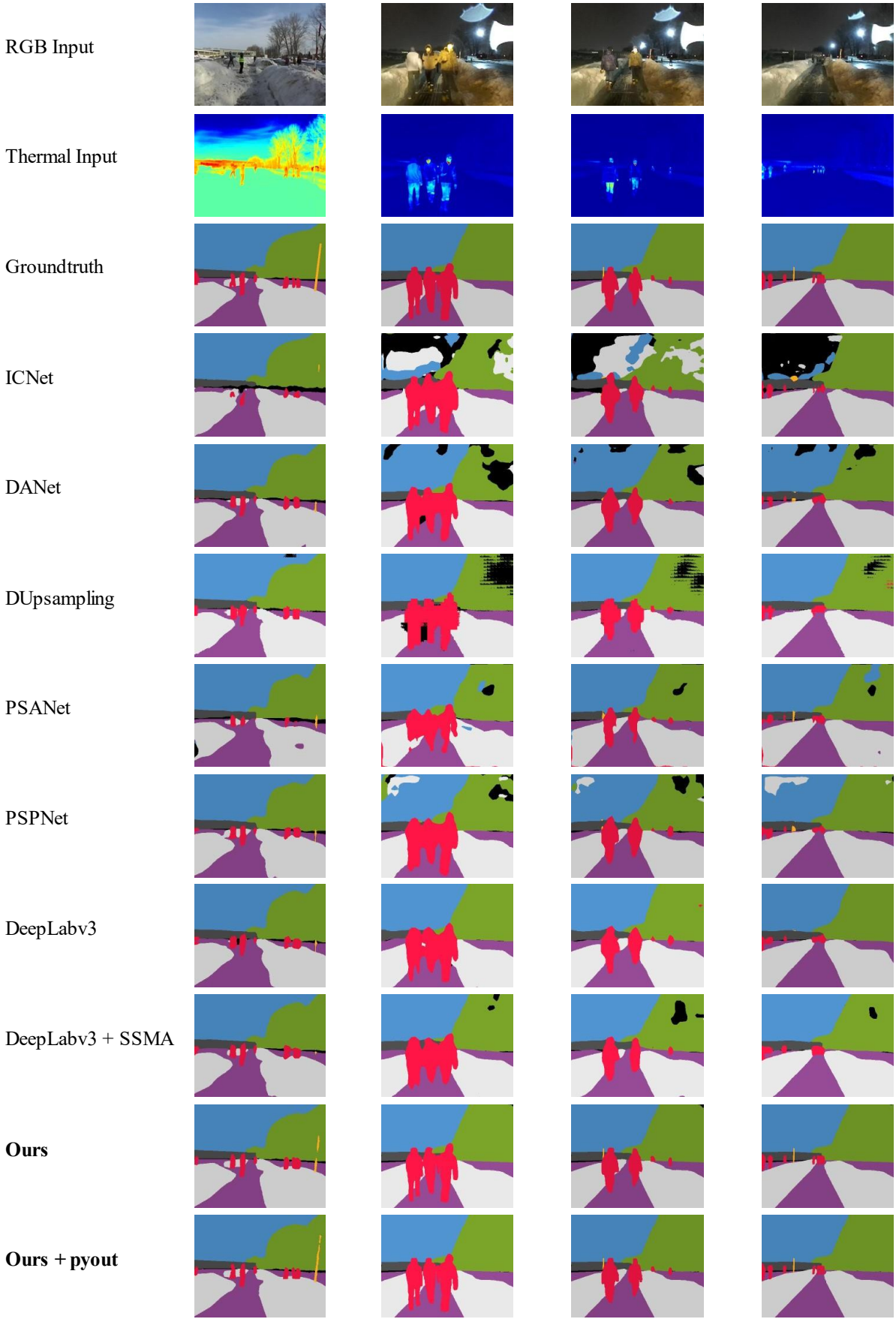


Figure 4-37 Example of segmentation results on SRM-1 dataset

Table 4-26 The segmentation results on SRM-2 dataset

ID	IoU (%)									mIoU(%)	mAcc(%)
	1	2	3	4	5	6	7	8	9		
ICNet	80.1	41.2	82.1	34.9	49.1	74.5	64.7	78.7	33.9	59.9	96.4
DANet	82.8	41.2	79.4	41.0	47.7	84.0	61.7	86.2	41.8	62.9	96.8
DUpsampling	82.0	33.8	78.9	30.1	4.7	83.6	53.0	86.1	40.3	54.7	96.3
PSANet	78.7	37.2	76.1	34.5	41.4	79.3	53.2	86.0	33.9	57.8	96.2
PSPNet	83.5	43.0	81.8	42.8	53.9	84.8	65.5	86.4	39.0	64.5	97.0
DeepLabv3	83.6	42.2	79.8	39.8	48.0	83.2	64.5	85.9	41.8	63.2	96.8
DeepLabv3 + SSMA	84.7	43.7	81.5	40.5	51.5	83.4	66.1	87.6	45.2	64.9	97.1
Ours	**83.3**	**41.4**	**82.2**	**39.7**	**50.6**	**82.7**	**64.2**	**85.5**	**43.2**	**63.6**	**97.0**
Ours + pyout	**88.7**	**59.4**	**87.1**	**51.2**	**51.3**	**88.0**	**67.4**	**91.0**	**50.4**	**70.5**	**97.8**

1:Road, 2:Snow mountain, 3:Building, 4:Traffic object, 5:Vegetation, 6:Sky, 7:Human, 8:Vehicle, 9:Unidentified

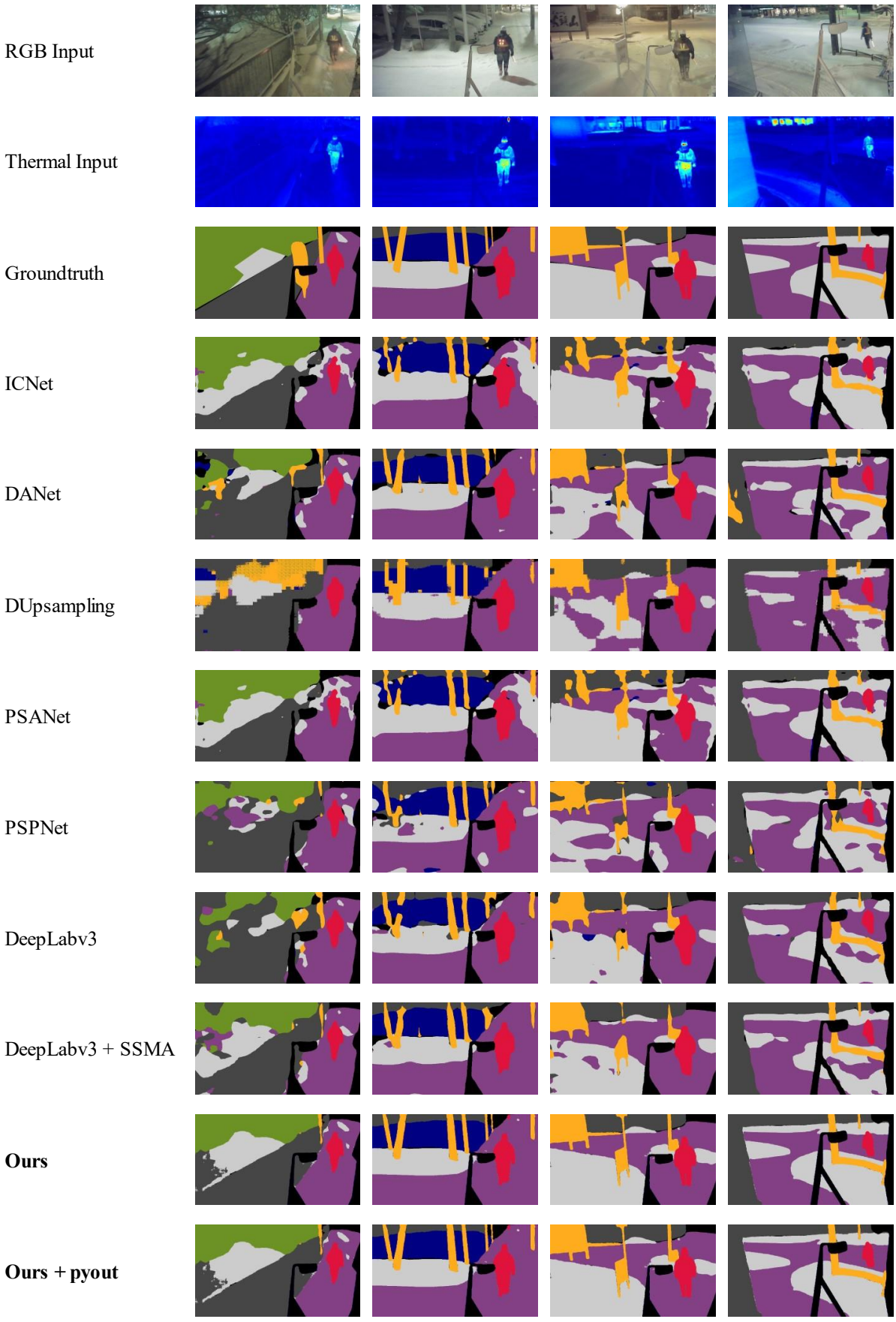


Figure 4-38 Example of segmentation results on SRM-2 dataset

Table 4-27 The segmentation results on Cityscape dataset

ID	IoU (%)											mIoU(%)	mAcc(%)
	1	2	3	4	5	6	7	8	9	10	11		
ICNet	77.5	31.1	65.8	26.3	15.7	69.1	1.1	13.4	69.4	13.1	57.3	40.0	96.0
DANet	92.5	63.0	77.7	42.5	24.8	79.5	52.9	48.3	82.6	15.3	65.6	58.6	97.7
DUpsampling	89.9	59.7	75.7	32.1	24.9	78.0	39.7	38.6	82.9	5.0	58.0	53.1	97.4
PSANet	80.0	53.3	66.5	31.1	26.0	63.3	34.1	45.2	74.5	17.7	39.3	48.3	96.0
PSPNet	92.4	61.1	75.6	44.1	22.5	79.7	38.7	46.3	81.0	14.5	61.4	56.1	97.5
DeepLabv3	93.1	64.7	76.8	50.2	24.8	78.3	47.4	48.7	83.9	24.6	65.0	59.8	97.7
DeepLabv3 + SSMA	91.2	64.7	76.2	44.0	28.6	75.6	33.9	51.9	82.1	32.0	60.5	58.2	97.5
Ours	**92.4**	**68.2**	**79.9**	**48.7**	**25.8**	**81.2**	**58.0**	**47.9**	**84.8**	**20.2**	**65.3**	**61.1**	**97.9**
Ours + pyout	**95.3**	**70.0**	**84.3**	**44.4**	**29.1**	**84.7**	**67.7**	**42.1**	**85.8**	**15.5**	**70.0**	**62.6**	**98.1**

1:Road, 2:Sidewalk, 3:Building, 4:Fence, 5:Traffic object, 6:Vegetation, 7:Sky, 8:Human, 9:Vehicle, 10: Bicycle, 11:Unidentified

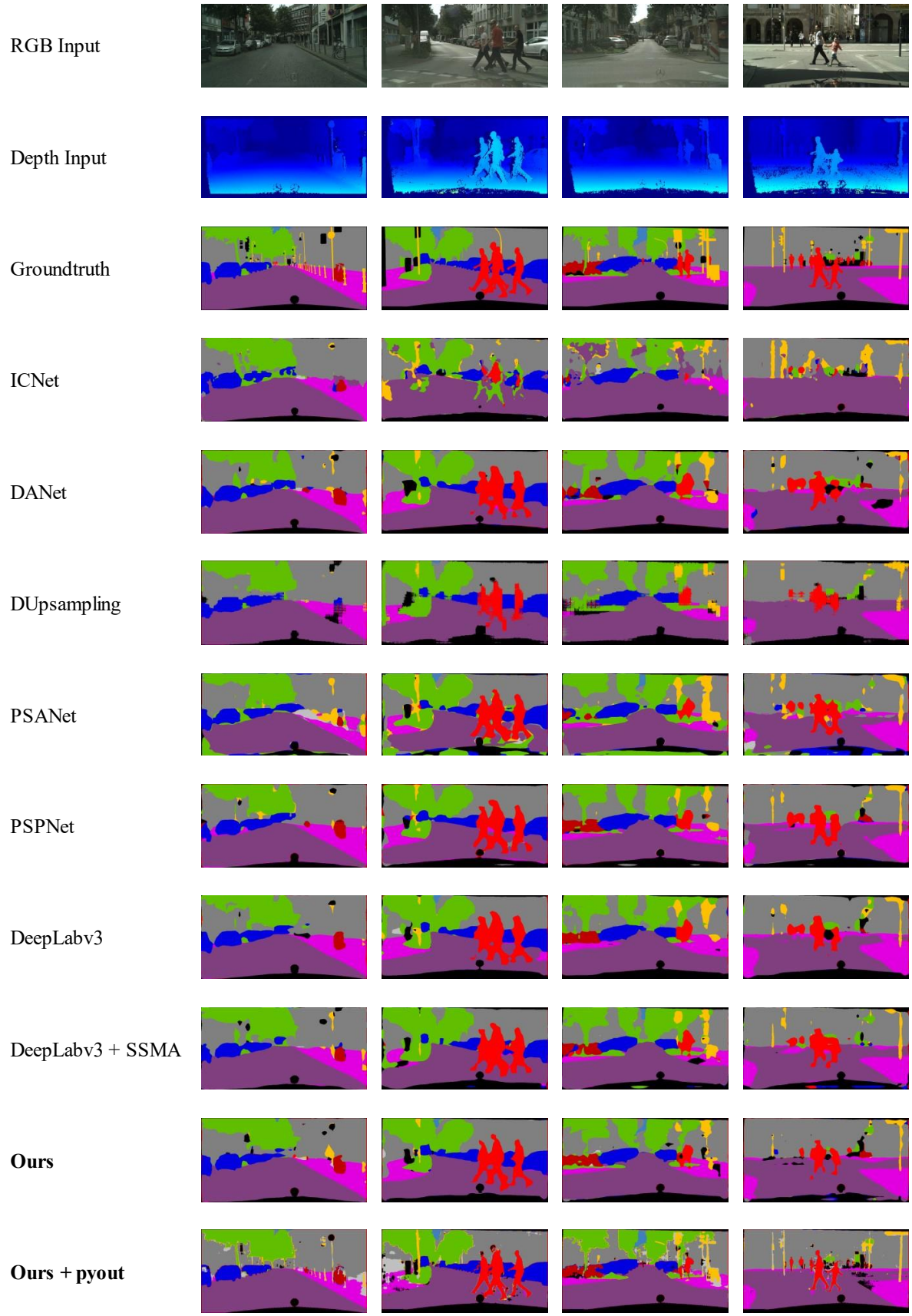


Figure 4-39 Example of segmentation results on Cityscape dataset

Table 4-28 The segmentation results on Synthia dataset

ID	IoU (%)										mIoU(%)	mAcc(%)
	1	2	3	4	5	6	7	8	9	10		
ICNet	4.3	9.5	76.1	15.9	3.2	26.9	93.9	2.8	26.9	19.2	27.9	89.5
DANet	44.5	59.4	72.8	67.7	23.8	37.5	93.7	10.3	70.8	32.7	51.3	94.0
DUpsampling	17.6	15.7	53.7	18.6	3.4	39.1	93.6	3.7	56.9	27.4	33.0	91.0
PSANet	82.6	40.6	83.2	67.9	22.0	35.7	78.0	18.8	55.0	66.5	55.0	97.2
PSPNet	19.5	54.2	60.2	67.8	21.5	43.9	93.4	9.3	73.8	27.8	47.1	92.0
DeepLabv3	79.1	49.2	87.1	77.3	24.7	52.7	93.3	25.4	84.3	50.6	62.4	97.1
DeepLabv3 + SSMA	91.9	47.5	85.3	65.2	17.5	38.2	94.0	25.6	82.1	68.0	61.5	98.0
Ours	**90.0**	**60.4**	**85.9**	**71.9**	**25.2**	**45.1**	**93.9**	**28.4**	**81.8**	**68.8**	**65.1**	**98.1**
Ours + pyout	**95.9**	**60.7**	**92.3**	**83.2**	**35.4**	**54.9**	**97.3**	**14.5**	**80.4**	**79.1**	**69.4**	**98.7**

1:Road, 2:Sidewalk, 3:Building, 4:Fence, 5:Traffic object, 6:Vegetation, 7:Sky, 8:Human, 9:Vehicle, 10:Unidentified

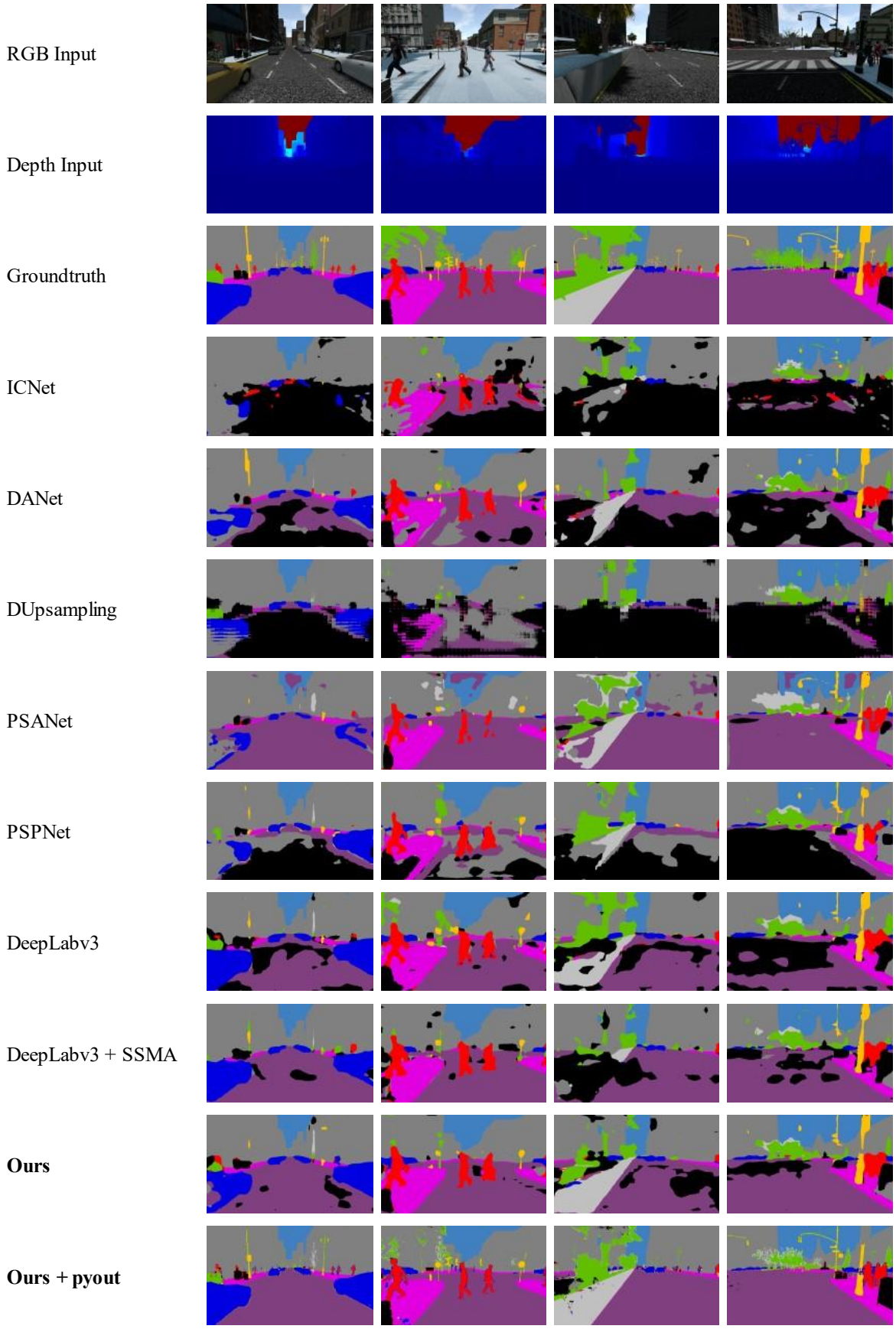


Figure 4-40 Example of segmentation results on SRM-2 dataset

k. The complexity of networks

The complexity of the network was measured and compared with other competitive networks. The increasing number of parameters, GPU memory, MACs(multiply-accumulate), and processing time were used as the indicators of this experiment. To simulate the actual utilization of the system that operated in real-time, this test was done with a single batch size of 3 × 640 × 480 input image. Table 4-29 shows the complexity comparison of the networks. The proposed network with pyramid output had the highest number of parameters but utilized lesser GPU memory than PSANet. The processing speed of the network was also faster than ICNet when feeding a single batch size of the input.

Table 4-29 The complexity comparison of networks

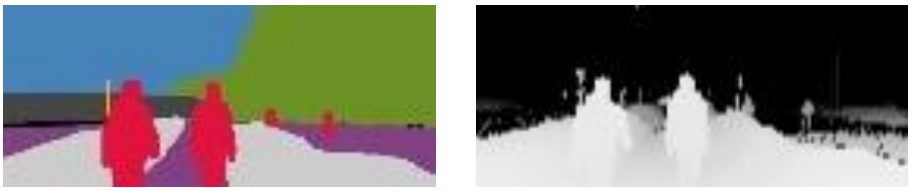
Network	Dataset		Parameters	Memory	MACs	Time
	RGB	T				
ICNet	✓	–	28.29M	0.93G	18.7G	52ms
	✓	✓	30.73M	1.15G	36.65G	112ms
DANet	✓	–	49.62M	1.78G	75.3G	31ms
	✓	✓	49.63M	2.46G	150.6G	63ms
DUpsampling	✓	–	34.53M	1.49G	57.19G	28ms
	✓	✓	39.47M	2.35G	109.78G	58ms
PSANet	✓	–	52.97M	2.19G	144.85G	31ms
	✓	✓	71.85M	3.81G	289.67G	61ms
PSPNet	✓	–	48.76M	1.69G	69.29G	31ms
	✓	✓	67.63M	2.79G	138.58G	59ms
DeepLabv3	✓	–	41.81M	1.46G	65.3G	31ms
	✓	✓	42.4M	2.21G	130.6G	59ms
DeepLabv3 + SSMA	✓	✓	42.54M	2.53G	130.77G	60ms
Ours	✓	✓	**45.5M**	**2.27G**	**134.31G**	**58ms**
Ours + pyout	✓	✓	**109M**	**3.53G**	**393.29G**	**75ms**

M is $\times 10^6$, G is $\times 10^9$

l. 3D visualization

The segmentation result was presented in the RGB image form, as shown in Figure 4-41 (a). It was an obvious form that was easy to display on the screen. In addition, the result could be combined with other information, such as the Depth map, as shown in Figure 4-41 (b). The bright area in Figure 4-41 (b) shows the near position, and the dark shows the far position. The absolute black is an over-measurement limit of the sensor. Finally, the result could be transformed from the RGB image, the 2D model, into the 3D point cloud to represent the 3D model.

The visualization process began with matching the calibration of the Depth map and the RGB image, as done with the Thermal map in Section 4.1-e. As a result, all feature maps, including the result, had the same position in the image. Next, the RGB-D image of the result was created by the Open3D library[102]. Lastly, the RGB-D image was converted to the point cloud form by the pinhold camera model, as described in Section 2.1-e-vi. Figure 4-42 shows the point cloud format, top & front view, of Figure 4-41. The red points in the front represent the human position near the sensor. The green and blue area at the back, shaped like a rectangle, shows the sensor's over-measurement limit point.



(a) (b)

Figure 4-41 RGB image format; (a) Segmentation result, (b) Depth maps

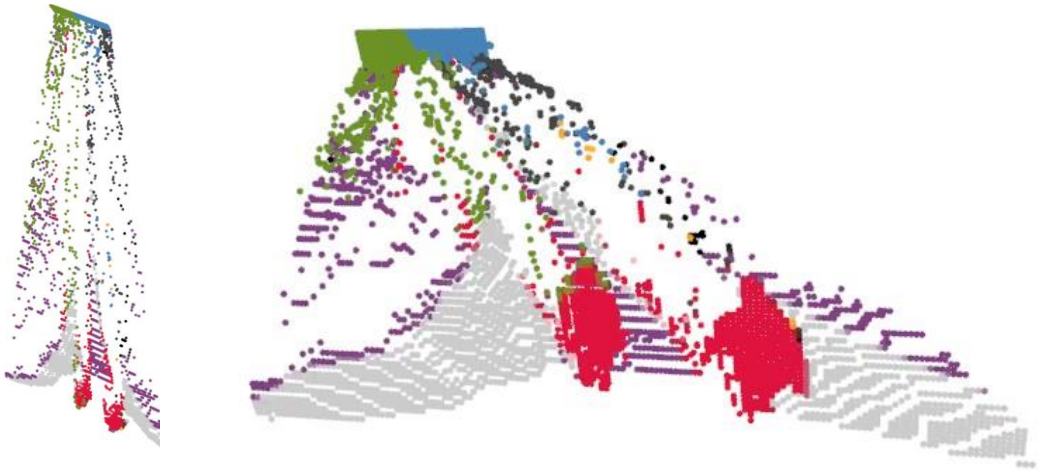


Figure 4-42 Point cloud format

CHAPTER 5

DISCUSSION

5. DISCUSSION

This chapter discusses the advantages and disadvantages of each introduced algorithm, the single input experiment, which used only an RGB image, and the multiple inputs, which used both an RGB image and a Thermal map. In addition, the blind spots of methods are also discussed in this part.

5.1 Single Input Experiment

In this section, a deep learning model for road environment detection in snowy conditions was introduced to apply autonomous driving. The recognition using deep learning shows robust results in the snowy dataset created for the test. The fewer the pixel number of each class, the worse the efficiency of the network. These results were shown as the IoU of the traffic object class in both datasets and the Lane marking-City object in the Mapillary Vistas dataset. The dataset with the various details or scenery made the lower efficiency, which means the networks worked better in the same scenario in a dataset. These may cause by the various detail of each object; for example, the difference in the color of snow could vary from dark gray to perfectly white. The object labeling in dataset creation is also essential. The separation of each class has to be considered based on the objective of the network and dataset. Suppose some classes obtain low performance in recognition. In that case, the network's overall performance or average efficiency will be affected directly.

The pyramid supervision application and obtaining the side output could improve the training and overall segmentation accuracy. Using this decoder, the filter variable can be updated directly from the loss calculation at each layer in the decoder. Therefore the training loss can efficiently improve. The proposed method could reach the mean intersection ratio of 86.4% in the Snowy road dataset and 54.2% in the Mapillary Vistas dataset with tiny changes in processing time compared to other state-of-the-art deep learning networks. The complexity of the decoder part could not make a difference in the processing speed of the network compared to the original one. The proposed method could enhance road surface detection and semantic segmentation in snowy conditions. It became the most robust algorithm for semantic segmentation by using only single input.

5.2 Multiple Inputs Experiment

This section has introduced a fused module for multi-input neural networks to use in snowy road environments. In a snowy environment, the use of a Thermal map as a second input was better than a single input of an RGB image. The whole network was chosen for the double Skip structure, which was demonstrated as the highest performance. The Thermal map has been represented as a jet colormap to improve the efficiency of the segmentation.

The use of the pyramid supervision decoder could improve the training loss during the training process. The side outputs for each layer of the decoder can be calculated for the loss of each decoder step. Then, each loss is used to update the filter variable of the network directly. The mission of the decoder is to expand the resulting scale. Therefore the feature map at each step of the decoder has to be the same image, different in size. Hence, the loss calculation in every step of the decoder, as pyramid supervision, is better than only the final loss calculation in the case of the training process.

Variables of the fused module were tested to tune the network for use in the snowy environment. The padding number (ν), the same value as the dilation number, was selected as a multiple of six based on the work of the DeepLabv3 network. It is enough to merge and capture the multiscale feature map after processing by the head module. The compression ration (γ) was selected as a power of two due to the half-scale number. The feature map in the first branch (cr) was the same size as the module's input, but in the second branch (cbr), the feature map size was reduced by the compression ratio. The best combination of padding number and compression ratio was used as the best setting for segmentation in a snowy environment.

The entire combination of the three branches of fused modules could make the best segmentation and human recognition efficient in a snowy environment. Each module branch is operated to generate different outputs to enhance the information of the feature map. The final output of this module is the merging output from each branch, then converted into the same size as the input.

The merging operators are an essential part of this fused module. The summation is to blend two feature maps together. So it creates a new feature map from the input. But the concatenation is to stack two feature maps. So the output of the concatenation has double in size of the feature map. The results of the merging operator combination told that the

concatenation operator is better to use in the fused module for a snowy environment. This is because it can preserve the most information in the feature map from each branch.

Softplus was used as an activation function in this module instead of ReLU. The difference between these two activation functions is their characteristic. The ReLU eliminates every negative value of the input. On the other hand, Softplus reduces the large negative value and make the output more smooth. Therefore, at each branch of the fused module, the best activation function was using ReLU as the first activation function and Softplus as the second function.

There are three fused modules in the network structure. One is to merge the feature map from the head module output and the others are to merge the feature maps from the encoder. The fused modules that merge for the encoder send the blended output directly to the decoder, called Skip. The Skip is a shortcut path for sending the information to the decoder without passing the head module. It also links the encoder to the decoder when updating the filter value during the training process. The best structure for working in the snowy environment is the network with a main fused module and two additional fused modules for creating the Skip.

The image from the thermal camera is a grayscale image, which contains a single channel of information. The grayscale image in this experiment was enhanced by converting to another colormap, called jet-colormap. The jet-colormap is an RGB image representing the image's range of value. The blue region in the jet-colormap is converted from the dark region in the grayscale image, and the red is converted to white. Therefore, the jet-colormap is a 3-channel image with three times more information than the grayscale image. Using jet-colormap to represent the thermal map can improve the efficiency of human recognition by 40% from using the grayscale image.

The proposed method showed robust human detection and higher mIoU in the snowy datasets, the SSW, the SRM-1, and the SRM-2 dataset. The network with the fused module and pyramid supervision path reached up to 79% mIoU and has become the state-of-the-art network for snowy environments. From the results on the Synthia and Cityscape datasets, it confidently could say that the fused module could enhance the efficiency of multiple input networks. The combination of proposed fused module and pyramid supervision path obtained the best results in both mAcc and mIoU. However, the complexity of the proposed network made the processing speed of the network slower than the original one. Although the complexity had

become 250% of the original, the processing time was increased by only 15%. Using a Thermal map met the blinding point when the heat-emitting objects were put in the ambient temperature for a long time. These made the object's temperature and environment equal, and no difference in the Thermal map. For example, the un-started car was parking outdoors in a snowy environment for a long time. Therefore, it should not obviously appear on the Thermal map. Instead, the Thermal map should be represented in the Jet-colormap because of the quantity of the information. The Jet-colormap contains three different pieces of information, but the Grayscale colormap contains only one piece of information. Therefore, the Jet-colormap is better than Grayscale to represent a Thermal map. The Thermal map could improve human detection from single input segmentation, especially at nighttime. The difference in temperatures between humans and their surroundings was higher at night. Therefore, it could recognize humans in detail better than using an only single input.

Currently, the proposed network is being developed for use with snow removal machine operations where sidewalks and snow road conditions exist. The network was trained with only nine classes of objects. The datasets did not include every object in daily life, for example, other heat-emitting objects like animals or maintenance holes, which can also be represented in the Thermal map. Therefore, the scope of use was limited to the introduced environment.

CHAPTER 6

CONCLUSION

6. CONCLUSION

In cold regions, such as Hokkaido of Japan, traffic problems are usually caused by snow that falls, unmelts, and piled up on the road. Not only does the slipping by snow lead to traffic accidents, but the visibility is also disturbed by the snow. Therefore, a process to reduce traffic problems due to the snow is sweeping and removing the snow from the central part of the road, the frequency use path. In a snowy road environment during the day, visibility is hampered because of reflection from snow on the road. Also, for snow removal machines operating at night, it is very difficult to recognize objects in the vicinity. Traditional neural networks that produce a high accuracy in clear snowless environment do not function well in snowy environments. The only visible light feature for the object detection system is not enough for practical use in a snowy environment. Hence, a thermal camera is utilized to improve the system's accuracy. The network introduced in this dissertation has been developed to work in this inclement weather conditions. During nighttime snow removal machine operations, the most important objective is to detect humans. The proposed network can respond to this class very well because the input of the network includes thermal information, so it is more accurate than other popular networks.

In conclusion, this dissertation has introduced the semantic segmentation systems in both single input and multiple inputs for a snowy environment. In such kind of that environment, the method with only single input segmentation with a pyramid supervision path can improve the original network performance work in a poor environment. Moreover, the use of a Thermal map as a second input is better than a single input of an RGB image. The proposed methods show robust human detection and higher mIoU in the snowy dataset. The network with the fused module and pyramid supervision path has become the state-of-the-art network for snowy environments. From the results on the Synthia and Cityscape datasets, it is confidently said that the introduced fused module can enhance the efficiency of multiple input networks. The combination of fused module and pyramid supervision path obtained the best results in both mAcc and mIoU.

In the future, to improve the proposed network efficiency, the system can be trained for more various classes in the actual environment, including different kinds of heat-emitting objects. In addition, other sensor information can be applied to obtain other unique features, which can enhance the segmentation efficiency.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] S. Vachmanus, A. A. Ravankar, T. Emaru, and Y. Kobayashi, "Multi-Modal Sensor Fusion-Based Semantic Segmentation for Snow Driving Scenarios," *IEEE Sens. J.*, vol. 21, no. 15, pp. 16839–16851, 2021, doi: 10.1109/JSEN.2021.3077029.

[2] S. Vachmanus, A. A. Ravankar, T. Emaru, and Y. Kobayashi, "An Evaluation of RGB-Thermal Image Segmentation for Snowy Road Environment," in *2021 IEEE International Conference on Mechatronics and Automation (ICMA)*, 2021, pp. 224–230, doi: 10.1109/ICMA52036.2021.9512708.

[3] S. Vachmanus, A. A.Ravankar, T. Emaru, and Y. Kobayashi, "Semantic Segmentation for Road Surface Detection in Snowy Environment," in *Proceedings of the SICE Annual Conference 2020 of the Society of Instrument and Control Engineers (SICE)*, 2020, pp. 1381–1386, doi: 10.23919/SICE48898.2020.9240402.

[4] D. Eisenberg and K. E. Warner, "Effects of snowfalls on motor vehicle collisions, injuries, and fatalities," *Am. J. Public Health*, vol. 95, no. 1, pp. 120–124, 2005, doi: 10.2105/AJPH.2004.048926.

[5] J. Weng, L. Liu, and J. Rong, "Impacts of Snowy Weather Conditions on Expressway Traffic Flow Characteristics," *Discret. Dyn. Nat. Soc.*, vol. 2013, p. 791743, 2013, doi: 10.1155/2013/791743.

[6] F. Malin, I. Norros, and S. Innamaa, "Accident risk of road and weather conditions on different road types," *Accid. Anal. Prev.*, vol. 122, no. February 2018, pp. 181–188, 2019, doi: 10.1016/j.aap.2018.10.014.

[7] K. SANO, T. INAGAKI, J. NAKANO, and N. C. Y, "An Analysis on Traffic Accidents on Undivided Expressway in Cold and Snow Area," *J. East. Asia Soc. Transp. Stud.*, vol. 8, pp. 2048–2061, 2010, doi: 10.11175/easts.8.2048.

[8] A. A. Ravankar, Y. Hoshino, A. Ravankar, L. Jixin, T. Emaru, and Y. Kobayashi, "Algorithms and a framework for indoor robot mapping in a noisy environment using clustering in spatial and hough domains," *Int. J. Adv. Robot. Syst.*, vol. 12, no. 3, p. 27, 2015.

[9] A. Ravankar, A. A. Ravankar, Y. Kobayashi, Y. Hoshino, and C.-C. Peng, "Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges," *Sensors*, vol. 18, no. 9, p. 3170, 2018.

[10] Z. Chen, J. Zhang, and D. Tao, "Progressive LiDAR Adaptation for Road Detection," *CoRR*, vol. abs/1904.0, 2019, [Online]. Available: http://arxiv.org/abs/1904.01206.

[11] F. Xu, L. Chen, J. Lou, and M. Ren, "A real-time road detection method based on reorganized lidar data," *PLoS One*, vol. 14, no. 4, pp. 1–17, 2019, doi: 10.1371/journal.pone.0215159.

[12] M. Marchetti, M. Moutton, S. Ludwig, L. Ibos, V. Feuillet, and J. Dumoulin, "Implementation of an infrared camera for road thermal mapping," 2010, doi: http://dx.doi.org/10.21611/qirt.2010.083.

[13] C. Fernández, D. Fernández-Llorca, and M. A. Sotelo, "A Hybrid Vision-Map Method for Urban Road Detection," *J. Adv. Transp.*, vol. 2017, 2017, doi: 10.1155/2017/7090549.

[14] Y. Li, W. Ding, X. Zhang, and Z. Ju, "Road detection algorithm for Autonomous Navigation Systems based on dark channel prior and vanishing point in complex road scenes," *Rob. Auton. Syst.*, vol. 85, pp. 1–11, 2016, doi: https://doi.org/10.1016/j.robot.2016.08.003.

[15] S. Vachmanus, T. Emaru, A. Ravankar, and Y. Kobayashi, "Road Detection in Snowy Forest Environment using RGB Camera," in *Proceedings of the 36 Annual Conference of the RSJ*, 2018, p. ROMBUNNO.1I3–07.

[16] G. Somasundaram, "Lane Change Detection and Tracking for a Safe-Lane Approach in Real Time Vision Based Navigation Systems," pp. 345–361, 2011, doi: 10.5121/csit.2011.1231.

[17] J. Baili *et al.*, "Lane Departure detection using image processing techniques," in *2017 2nd International Conference on Anti-Cyber Crimes (ICACC)*, 2017, pp. 238–241, doi: 10.1109/Anti-Cybercrime.2017.7905298.

[18] X. Jiang, B. Hu, S. Chandra Satapathy, S.-H. Wang, and Y.-D. Zhang, "Fingerspelling Identification for Chinese Sign Language via AlexNet-Based Transfer Learning and Adam Optimizer," *Sci. Program.*, vol. 2020, p. 3291426, 2020, doi: 10.1155/2020/3291426.

[19] X. Jiang, S. C. Satapathy, L. Yang, S.-H. Wang, and Y.-D. Zhang, "A Survey on Artificial Intelligence in Chinese Sign Language Recognition," *Arab. J. Sci. Eng.*, vol. 45, no. 12, pp. 9859–9894, 2020, doi: 10.1007/s13369-020-04758-2.

[20] P. S. Zaki, M. M. William, B. K. Soliman, K. G. Alexsan, K. Khalil, and M. El-Moursy, "Traffic Signs Detection and Recognition System using Deep Learning." 2020.

[21] M. Thoma, "A Survey of Semantic Segmentation," *CoRR*, vol. abs/1602.0, 2016, [Online]. Available: http://arxiv.org/abs/1602.06541.

[22] G. Bradski and A. Kaehler, *Learning OpenCV*. O'Reilly Media, Inc., 2008.

[23] OpenCV team, "OpenCV– Online documentation | OpenCV team," *Open Source Computer Vision*, 2022. https://opencv.org (accessed Jan. 05, 2022).

[24] D. Demolder, “Canon is now selling CMOS image sensors, including a 120MP APS-H beast,” *Digital Photography Review*, 2018. https://www.dpreview.com/news/0671207908/canon-is-now-selling-cmos-image-sensors-including-a-120mp-aps-h-beast (accessed Jan. 05, 2022).
[25] Thermal Imaging Technology, “Thermal Scope,” 2020. thermalscope.com/ (accessed Jan. 05, 2022).
[26] Umicore, “Overview of infrared.” https://eom.umicore.com/en/infrared-solutions/overview-of-infrared/ (accessed Jan. 05, 2022).
[27] I. The MathWorks, “Image Types in the Toolbox,” *MATLAB R2021b*, 2022. https://www.mathworks.com/help/images/image-types-in-the-toolbox.html#f14-33397 (accessed Jan. 06, 2022).
[28] I. The MathWorks, “What Is a Colormap?,” *MATLAB R2021b*, 2021. https://www.mathworks.com/help/matlab/ref/colormap.html#buq1hym (accessed Jan. 06, 2022).
[29] C. Moler, “Origins of Colormaps,” *MATLAB R2014b*, 2015. https://blogs.mathworks.com/cleve/2015/02/02/origins-of-colormaps/.
[30] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition)*. USA: Prentice-Hall, Inc., 2006.
[31] G. Wolberg, *Digital Image Warping*, 1st ed. Washington, DC, USA: IEEE Computer Society Press, 1994.
[32] OpenCV team, “Affine Transformations,” *Open Source Computer Vision*, 2022. https://docs.opencv.org/3.4/d4/d61/tutorial_warp_affine.html (accessed Jan. 07, 2022).
[33] Wikipedia contributors, “Bicubic interpolation --- {Wikipedia}{,} The Free Encyclopedia.” 2021, [Online]. Available: https://en.wikipedia.org/w/index.php?title=Bicubic_interpolation&oldid=1048457243.
[34] A. C. Bovik, *The Essential Guide to Image Processing*. Boston: Academic Press, 2009.
[35] J. Heikkila and O. Silven, “A four-step camera calibration procedure with implicit image correction,” in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997, pp. 1106–1112, doi: 10.1109/CVPR.1997.609468.
[36] O. Stankiewicz, G. Lafruit, and M. Domański, “Chapter 1 - Multiview video: Acquisition, processing, compression, and virtual view rendering,” in *Academic Press Library in Signal Processing, Volume 6*, R. Chellappa and S. Theodoridis, Eds. Academic Press, 2018, pp. 3–74.
[37] Torch Contributors, “PYTORCH DOCUMENTATION,” 2019. https://pytorch.org/docs/stable/index.html (accessed Jan. 18, 2022).
[38] N. Ketkar, “Introduction to PyTorch,” in *Deep Learning with Python: A Hands-on Introduction*, Berkeley, CA: Apress, 2017, pp. 195–208.
[39] Wikipedia contributors, “PyTorch --- {Wikipedia}{,} The Free Encyclopedia.” 2022, [Online]. Available: https://en.wikipedia.org/w/index.php?title=PyTorch&oldid=1063971511.
[40] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural Networks*, vol. 61, pp. 85–117, 2015, doi: https://doi.org/10.1016/j.neunet.2014.09.003.
[41] Y. Bengio, A. Courville, and P. Vincent, “Representation Learning: A Review and New Perspectives,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, 2013, doi: 10.1109/TPAMI.2013.50.
[42] J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin, “Voronoi-Based Multi-Robot Autonomous Exploration in Unknown Environments via Deep Reinforcement Learning,” *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 14413–14423, 2020, doi: 10.1109/TVT.2020.3034800.
[43] D. Ciregan, U. Meier, and J. Schmidhuber, “Multi-column deep neural networks for image classification,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3642–3649, doi: 10.1109/CVPR.2012.6248110.
[44] M. V Valueva, N. N. Nagornov, P. A. Lyakhov, G. V Valuev, and N. I. Chervyakov, “Application of the residue number system to reduce hardware costs of the convolutional neural network implementation,” *Math. Comput. Simul.*, vol. 177, pp. 232–243, 2020, doi: https://doi.org/10.1016/j.matcom.2020.04.031.
[45] K. Fukushima, “Neocognitron,” *Scholarpedia*, vol. 2, no. 1, p. 1717, 2007, doi: 10.4249/scholarpedia.1717.
[46] D. H. Hubel and T. N. Wiesel, “Receptive fields and functional architecture of monkey striate cortex,” *J. Physiol.*, vol. 195, no. 1, pp. 215–243, 1968, doi: https://doi.org/10.1113/jphysiol.1968.sp008455.
[47] M. Matsugu, K. Mori, Y. Mitari, and Y. Kaneda, “Subject independent facial expression recognition with robust face detection using a convolutional neural network,” *Neural Networks*, vol. 16, no. 5, pp. 555–559, 2003, doi: https://doi.org/10.1016/S0893-6080(03)00115-1.
[48] M. Mandal, “Introduction to Convolutional Neural Networks (CNN),” 2021. https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/#h2_1 (accessed Jan. 12, 2022).
[49] J. Jordan, “An overview of semantic image segmentation.,” 2018. https://www.jeremyjordan.me/semantic-segmentation/ (accessed Jan. 13, 2022).
[50] E. Shelhamer, J. Long, and T. Darrell, “Fully Convolutional Networks for Semantic Segmentation,” *IEEE*

Trans. Pattern Anal. Mach. Intell., vol. 39, no. 4, pp. 640–651, 2017, doi: 10.1109/TPAMI.2016.2572683.
[51] IBM Cloud Education, "Convolutional Neural Networks," 2020. https://www.ibm.com/cloud/learn/convolutional-neural-networks (accessed Jan. 13, 2022).
[52] Torch Contributors, "CONV2D," 2019. https://pytorch.org/docs/1.9.1/generated/torch.nn.Conv2d.html (accessed Jan. 13, 2022).
[53] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *CoRR*, vol. abs/1502.0, 2015, [Online]. Available: http://arxiv.org/abs/1502.03167.
[54] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 56, pp. 1929–1958, 2014, [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html.
[55] J. Brownlee, "How to Choose an Activation Function for Deep Learning," *2021*. https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/ (accessed Jan. 17, 2022).
[56] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, 2010, pp. 807–814.
[57] Torch Contributors, "RELU," 2019. https://pytorch.org/docs/1.9.1/generated/torch.nn.ReLU.html#torch.nn.ReLU (accessed Jan. 17, 2022).
[58] J. Han and C. Moraga, "The influence of the sigmoid function parameters on the speed of backpropagation learning," in *From Natural to Artificial Neural Computation*, 1995, pp. 195–201.
[59] Torch Contributors, "SIGMOID," 2019. https://pytorch.org/docs/1.9.1/generated/torch.nn.Sigmoid.html#torch.nn.Sigmoid (accessed Jan. 17, 2022).
[60] J. F. Low, "Softplus and softminus," 2019. https://jiafulow.github.io/blog/2019/07/11/softplus-and-softminus/ (accessed Jan. 17, 2022).
[61] Torch Contributors, "SOFTPLUS," 2019. https://pytorch.org/docs/1.9.1/generated/torch.nn.Softplus.html#torch.nn.Softplus (accessed Jan. 17, 2022).
[62] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol. abs/1207.0, 2012, [Online]. Available: http://arxiv.org/abs/1207.0580.
[63] baeldung, "How ReLU and Dropout Layers Work in CNNs," 2020. https://www.baeldung.com/cs/ml-relu-dropout-layers (accessed Jan. 17, 2022).
[64] F. Chollet, *Deep Learning with Python*. Manning, 2017.
[65] D. Saad, *On-Line Learning in Neural Networks*. Cambridge University Press, 1998.
[66] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of the 30th International Conference on Machine Learning*, 2013, vol. 28, no. 3, pp. 1139–1147, [Online]. Available: https://proceedings.mlr.press/v28/sutskever13.html.
[67] Torch Contributors, "SGD," 2019. https://pytorch.org/docs/stable/generated/torch.optim.SGD.html (accessed Jan. 18, 2022).
[68] A. Dattalo, "ROS Introduction," 2018. http://wiki.ros.org/ROS/Introduction (accessed Jan. 18, 2022).
[69] L. Joseph and J. Cacace, *Mastering ROS for Robotics Programming - Second Edition: Design, Build, and Simulate Complex Robots Using the Robot Operating System*, 2nd ed. Packt Publishing, 2018.
[70] M. Jokela, M. Kutila, and L. Le, "Road condition monitoring system based on a stereo camera," in *2009 IEEE 5th International Conference on Intelligent Computer Communication and Processing*, Aug. 2009, pp. 423–428, doi: 10.1109/ICCP.2009.5284724.
[71] A. Troiano, E. Pasero, and L. Mesin, "New system for detecting road ice formation," *IEEE Trans. Instrum. Meas.*, vol. 60, no. 3, pp. 1091–1101, 2011, doi: 10.1109/TIM.2010.2064910.
[72] R. Omer and L. Fu, "An automatic image recognition system for winter road surface condition classification," in *13th International IEEE Conference on Intelligent Transportation Systems*, 2010, pp. 1375–1379, doi: 10.1109/ITSC.2010.5625290.
[73] P. Jonsson, J. Casselgren, and B. Thornberg, "Road surface status classification using spectral analysis of NIR camera images," *IEEE Sens. J.*, vol. 15, no. 3, pp. 1641–1656, 2015, doi: 10.1109/JSEN.2014.2364854.
[74] S. Kawai, K. Takeuchi, K. Shibata, and Y. Horita, "A smart method to distinguish road surface conditions at night-time using a car-mounted camera," *IEEJ Trans. Electron. Inf. Syst.*, vol. 134, no. 6, pp. 878–884, 2014, doi: 10.1541/ieejeiss.134.878.
[75] N. John, B. Anusha, and K. Kutty, "A Reliable Method for Detecting Road Regions from a Single Image Based on Color Distribution and Vanishing Point Location," *Procedia Comput. Sci.*, vol. 58, pp. 2–9, 2015, doi: 10.1016/j.procs.2015.08.002.

[76] Z. Pan, T. Emaru, A. Ravankar, and Y. Kobayashi, "Applying Semantic Segmentation to Autonomous Cars in the Snowy Environment," *arXiv Prepr. arXiv2007.12869*, 2020.

[77] P.-R. Chen, H.-M. Hang, S.-W. Chan, and J.-J. Lin, "DSNet: An Efficient {CNN} for Road Scene Segmentation," *CoRR*, vol. abs/1904.0, 2019, [Online]. Available: http://arxiv.org/abs/1904.05022.

[78] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "ICNet for Real-Time Semantic Segmentation on High-Resolution Images," *CoRR*, vol. abs/1704.0, 2017, [Online]. Available: http://arxiv.org/abs/1704.08545.

[79] Z. Tian, T. He, C. Shen, and Y. Yan, "Decoders matter for semantic segmentation: Data-dependent decoding enables flexible feature aggregation," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2019-June, pp. 3121–3130, 2019, doi: 10.1109/CVPR.2019.00324.

[80] C. Liang, J. Ge, W. Zhang, K. Gui, F. A. Cheikh, and L. Ye, "Winter Road Surface Status Recognition Using Deep Semantic Segmentation Network," in *International Workshop on Atmospheric Icing of Structures*, 2019, pp. 1–6.

[81] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *CoRR*, vol. abs/1505.0, 2015, [Online]. Available: http://arxiv.org/abs/1505.04597.

[82] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 6230–6239, 2017, doi: 10.1109/CVPR.2017.660.

[83] X. Gao *et al.*, "An End-to-End Neural Network for Road Extraction from Remote Sensing Imagery by Multiple Feature Pyramid Network," *IEEE Access*, vol. 6, pp. 39401–39414, 2018, doi: 10.1109/ACCESS.2018.2856088.

[84] H. Zhao *et al.*, "PSANet: Point-wise Spatial Attention Network for Scene Parsing," in *Computer Vision -- ECCV 2018*, 2018, pp. 270–286.

[85] J. Fu, J. Liu, H. Tian, Z. Fang, and H. Lu, "Dual Attention Network for Scene Segmentation," *CoRR*, vol. abs/1809.0, 2018, [Online]. Available: http://arxiv.org/abs/1809.02983.

[86] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking Atrous Convolution for Semantic Image Segmentation," *CoRR*, vol. abs/1706.0, 2017, [Online]. Available: http://arxiv.org/abs/1706.05587.

[87] J. Jiang, L. Zheng, F. Luo, and Z. Zhang, "RedNet: Residual Encoder-Decoder Network for indoor {RGB-D} Semantic Segmentation," *CoRR*, vol. abs/1806.0, 2018, [Online]. Available: http://arxiv.org/abs/1806.01054.

[88] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *CoRR*, vol. abs/1512.0, 2015, [Online]. Available: http://arxiv.org/abs/1512.03385.

[89] N. Silberman and R. Fergus, "Indoor scene segmentation using a structured light sensor," *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 601–608, 2011, doi: 10.1109/ICCVW.2011.6130298.

[90] S. Gupta, R. B. Girshick, P. Arbelaez, and J. Malik, "Learning Rich Features from {RGB-D} Images for Object Detection and Segmentation," *CoRR*, vol. abs/1407.5, 2014, [Online]. Available: http://arxiv.org/abs/1407.5736.

[91] A. Valada, G. L. Oliveira, T. Brox, and W. Burgard, "Deep Multispectral Semantic Scene Understanding of Forested Environments Using Multimodal Fusion," in *International Symposium on Experimental Robotics (ISER 2016)*, pp. 465–477, doi: 10.1007/978-3-319-50115-4_41.

[92] A. Valada, R. Mohan, and W. Burgard, "Self-Supervised Model Adaptation for Multimodal Semantic Segmentation," *Int. J. Comput. Vis.*, vol. 128, no. 5, pp. 1239–1285, 2020, doi: 10.1007/s11263-019-01188-y.

[93] C. Hazırbaş, L. Ma, C. Domokos, and D. Cremers, "FuseNet: Incorporating Depth into Semantic Segmentation via Fusion-Based CNN Architecture," 2016, doi: 10.1007/978-3-319-54181-5_14.

[94] W. Wang and U. Neumann, "Depth-aware CNN for RGB-D Segmentation," *CoRR*, vol. abs/1803.0, 2018, [Online]. Available: http://arxiv.org/abs/1803.06791.

[95] H. Qishen, W. Kohei, K. Takumi, U. Yoshitaka, and H. Tatsuya, "MFNet: Towards real-time semantic segmentation for autonomous vehicles with multi-spectral scenes," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 5108–5115, doi: 10.1109/IROS.2017.8206396.

[96] S. Yuxiang, Z. Weixun, and M. Liu, "RTFNet: RGB-Thermal Fusion Network for Semantic Segmentation of Urban Scenes," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2576–2583, 2019, doi: 10.1109/LRA.2019.2904733.

[97] G. Neuhold, T. Ollmann, S. R. Bulò, and P. Kontschieder, "The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 5000–5009, doi: 10.1109/ICCV.2017.534.

[98] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "BiSeNet: Bilateral Segmentation Network for Real-time Semantic Segmentation," *CoRR*, vol. abs/1808.0, 2018, [Online]. Available: http://arxiv.org/abs/1808.00897.

[99] M. Cordts *et al.*, "The Cityscapes Dataset for Semantic Urban Scene Understanding," *CoRR*, vol.

abs/1604.0, 2016, [Online]. Available: http://arxiv.org/abs/1604.01685.

[100] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, no. 600388, pp. 3234–3243, 2016, doi: 10.1109/CVPR.2016.352.

[101] J. K. Haas, "A history of the unity game engine," 2014.

[102] Q.-Y. Zhou, J. Park, and V. Koltun, "{Open3D}: {A} Modern Library for {3D} Data Processing," *arXiv:1801.09847*, 2018.

APPENDIX

Sensors Technical Specifications

ZED Mini

Video Output

Output Resolution	Side by Side 2× (2208×1242) @15fps 2× (1920×1080) @30fps 2× (1280×720) @60fps 2× (672×76) @100fps
Output Format	YUV 4:2:2
Field of View	Max. 90° (H) × 60° (V) × 100° (D)
RGB Sensor Type	1/3” 4MP CMOS
Active Array Size	2688x1520 pixels per sensor (4MP)
Focal Length	2.8mm (0.11”) - f/2.0
Shutter	Electronic synchronized rolling shutter
Interface	USB 3.0 Type-C port

Depth Sensing

Baseline	63 mm (2.4”)
Depth Range	0.10 m to 15 m (0.3 to 49 ft)
Depth Map Resolution	Native video resolution (in Ultra mode)
Depth Accuracy	< 1.5% up to 3m < 7% up to 15m

Motion

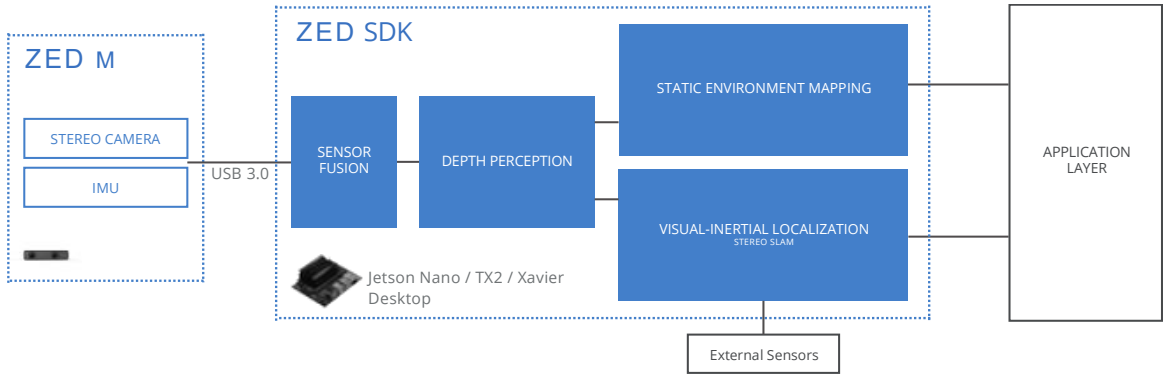
Motion Sensors	Gyroscope, Accelerometer Sampling Rate 800Hz
Technology	Visual-inertial stereo SLAM
6-axis Pose Accuracy	Position: +/- 1mm , Orientation: 0.1 deg.
Pose Update Rate	Up to 100 Hz

Physical

Dimensions	124.5 × 30.5 × 26.5 mm (4.9 × 1.2 × 1.0”)
Weight	62.9g - 0.14 lb
Operating Temperature	0°C to +45°C (32°F to 113°F)
Power	380mA / 5V USB Powered

System Requirements	Win 10, Win 8 Win 7 Ubuntu 18.0/16.04 CentOS, Debian (via Docker) USB3.0 Interface
SDK Requirements	Dual-core 2.3GHz or faster Minimum 4GB RAM Memory Nvidia GPU* Compute capability ≥ 3.0 * Compatible with Nvidia Jetson Nano, TX2, Xavier

Functional SDK Diagram



Reference

"ZED Mini - Mixed-Reality Camera", *Stereolabs.com*, 2022. [Online]. Available: https://www.stereolabs.com/zed-mini/. [Accessed: 20- Jan- 2022].

ZED 2

Sensors

Sensor Type	1/3” 4MP CMOS	
Array Size	2688 × 1520 pixels	
Pixel Size	2µm × 2µm	
Shutter	Electronic synchronized rolling shutter	
Output Resolution	(Side by side)	
	2×(2208×1242) @15fps	cropping mode
	2×(1920×1080) @15/30fps	cropping mode
	2×(1280×720) @15/30/60fps	binning 2×2 mode
	2×(672×376) @15/30/60/100fps	binning 4×4 mode
Output Format	YUV 4:2:2 - UYVY (8bits)	
Max S/N Ratio	38.3 dB	
Dynamic Range	64.6 dB	
Sensitivity	1900 mV/Lux-sec	
Depth Range	0.3 m to 20 m (1 to 65.6 ft)	
Depth Accuracy	< 1% up to 3m	
	< 5% up to 15m	

Lenses

Baseline	120 mm (4.7”)
Field of View	Max. 110°(H) × 70°(V) × 120°(D)
Aperture	f/1.8
TV Distortion	5.07%

Physical

Dimensions	175 × 30 × 33 mm (6.89 × 1.18 × 1.3”)
Weight	166g (0.36 lb
Operating Temp.	-10°C to +45°C (14°F to 113°F)
Power	380mA / 5V USB Powered

Inertial Measurement Unit

Accelerometer Range	+/- 8G
Accelerometer Resolution	0.244 mg
Accelerometer Noise Density	3.2 mg
Gyroscope Range	+/- 1000 dps
Gyroscope Resolution	0.03 dps
Gyroscope Noise Density	0.16 dps
Sensitivity Error	+/- 0.4%
Output Data Rate	400 Hz

Magnetometer

Magnetic Field Range	+/- 2500 µT (z) +/- 1300 µT (x,y)
Magnetic Field Resolution	0.3 µT
Output Data Rate	50 Hz

Barometer

Pressure Range	300 to 1100 hPa
Pressure Resolution	0.18 Pa
Relative Pressure Accuracy	0.12 hPa
RMS Noise	0.2 Pa
Output Data Rate	25 Hz

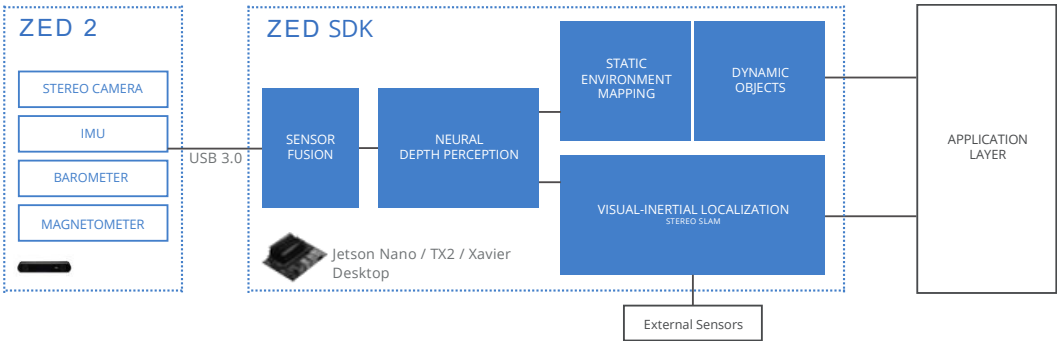
Temperature Sensors

Temperature Range	-40 to 125 °C
Abs. Temperature Accuracy	+/- 0.5 °C
Output Data Rate	25 Hz

System Requirements

Supported OS	Windows 10 - 64 bit Ubuntu 16.04/18.04 - 64 bit Debian, CentOS (via Docker) Jetson L4T
CPU	Dual-core ≥ 2.4GHz processor Minimum 4GB RAM
GPU	NVIDIA GPU ≥ 2GB Memory NVIDIA Compute capability ≥ 3.0 Compatible with: – NVIDIA Jetson Nano – NVIDIA Jetson TX2 – NVIDIA Jetson Xavier

Functional SDK Diagram



Reference

"ZED 2i - Industrial AI Stereo Camera", *Stereolabs.com*, 2022. [Online]. Available: https://www.stereolabs.com/zed-2i/. [Accessed: 20- Jan- 2022].

Optris PI 640i

Technical specifications

Optical resolution	640 × 480 pixels
Detector	FPA, uncooled (17 µm × 17 µm)
Spectral range	8 – 14 µm
Temperature ranges	–20 to 100 °C, 0 to 250 °C, (20) 150 to 900 °C
Frame rate	32 Hz / 125 Hz @ 640 × 120 pixels
Optics (FOV)	15° × 11° FOV / f = 41.5 mm
	33° × 25° FOV / f = 18.7 mm
	60° × 45° FOV / f = 10.5 mm
	90° × 66° FOV / f = 7.7 mm
Thermal sensitivity (NETD)	40 mK
Accuracy	±2 °C or ±2 %, whichever is greate
PC interface	USB 2.0 / optional USB GigE (PoE) interface
Process interface (PIF)	0 – 10 V input, digital input (max. 24 V)
Cable length (USB)	1 m (standard), 5 m, 10 m, 20 m
Ambient temperature	0 to 50 °C
Storage temperature	– 40 to 85 °C
Relative humidity	20 – 80 %, non-condensing
Enclosure (size / rating)	46 × 56 × 76 mm / IP 67 (NEMA)
Weight	269 g
Shock / Vibration	IEC 60068-2-27 (25G and 50G)
Tripod mount	¼- 20 UNC
Power supply	via USB

Reference

"optris PI 640i The smallest measuring VGA thermal imager worldwide", *Optris.global*. [Online]. Available: https://www.optris.global/thermal-imager-optris-pi-640. [Accessed: 20-Jan- 2022].

OS-1

Optical Performance

Range (80% Lambertian reflectivity, 1024 @ 10 Hz)	100 m @ >90% detection probability 120 m @ >50% detection probability
Range (10% Lambertian reflectivity, 1024 @ 10 Hz)	45 m @ >90% detection probability 55 m @ >50% detection probability
Minimum Range	0.3 m for point cloud data
Range Accuracy	±3 cm for lambertian targets ±10 cm for retroreflectors
Precision (10% Lambertian reflectivity, 1024 @10 Hz, 1 standard deviation)	0.3 - 1 m: ± 0.7 cm 1 - 20 m: ± 1 cm 20 - 50 m ± 2 cm >50 m: ± 5 cm
Range Resolution	0.3 cm
Vertical Resolution	32, 64, or 128 channels
Horizontal Resolution	512, 1024, or 2048 (configurable)
Field of View	Vertical: 45° (+22.5° to -22.5°) Horizontal: 360°
Angular Sampling Accuracy	Vertical: ±0.01° / Horizontal: ±0.01°
False Positive Rate	1/10,000
Rotation Rate	10 or 20 Hz (configurable)
No. of Returns	2 (strongest, second strongest)

Laser

Laser Product	Class 1 eye-safe per IEC/EN 60825-1: 2014
Laser Wavelength	865 nm
Beam Diameter Exiting Sensor	9.5 mm
Beam Divergence	0.18° (FWHM)

Lidar Output

Connection	UDP over gigabit Ethernet
Points Per Second	up to 655,360 (32 channels) up to 1,310,720 (64 channels) up to 2,621,440 (128 channels)
Data Rate (Megabits Per Second) (Legacy Mode)	up to 66 Mbps (32 channels) up to 129 Mbps (64 channels) up to 254 Mbps (128 channels)
Data Rate* (Megabits per second) (Dual Return Mode) * Not applicable for 1024x20 & 2048x10	up to 43.6 Mbps (32 channels) up to 85.6 Mbps (64 channels) up to 169.4 Mbps (128 channels)
Data Per Point	Range, signal, reflectivity, near-infrared, channel, azimuth angle, timestamp
Timestamp Resolution	< 1 µs
Data Latency	< 10 ms

IMU Output

Connection	UDP over gigabit Ethernet
Samples Per Second	100
Data Per Sample	3 axes gyro, 3 axes accelerometer
Timestamp Resolution	< 1 µs
Data Latency	< 10 ms
Model	InvenSense ICM-20948

Control Interface

Connection	TCP and HTTP APIs
Time Synchronization	Input sources: - IEEE1588 Precision Time Protocol (PTP); Accuracy: <1 ms error - gPTP; Accuracy: <1 ms error - NMEA $GPRMC UART message support - External PPS; Accuracy: <1 ms error - Internal 10 ppm drift clock; Accuracy: <20 ppm error Output sources: - Configurable 1 - 60 Hz output pulse
Lidar Operating Modes	×512 @ 10 Hz or 20 Hz ×1024 @ 10 Hz or 20 Hz ×2048 @ 10 Hz
Additional Programmability	Multi-sensor Phase Lock Azimuth Masking Low-power Standby Mode Queryable intrinsic calibration information: - Beam angles - IMU pose correction matrix

Mechanical/Electrical

Power Consumption	14 - 20 W (23 W peak at startup, 28 W peak if operating below -40 °C)
Operating Voltage	9V - 34 V, 12 V or 24 V nominal
Connector	Proprietary pluggable connector (Power + data + DIO)
Dimensions	Diameter: 85 mm (3.34 in) Height without cap: 58.35 mm (2.3 in) Height with cap:73.5 mm (2.9 in)
Weight	Without cap: 377 g (13.3 oz) With radial cap: 447 g (15.8 oz)
Mounting	Bottom: 4×M3 screws, 2×locating 2 mm pin holes Top: 4×M3 screws, 4×locating 2 mm pin holes, 1×M6 screw

Operational

Operating Temperature	-40 °C to +60 °C (with mount) Between +53 °C and +60 °C, sensor automatically reduces range (max 20% range reduction)
Storage Temperature	-40 °C to +75 °C
Ingress Protection	IP68 (1m submersion for 1 hour, with I/O cable attached) IP69K (with I/O cable attached)
Shock	IEC 60068-2-27 (Amplitude: 100 g, Shape: 11 ms half-sine, 3 shocks × 6 directions)
Vibration	IEC 60068-2-64 (Amplitude: 3 G-rms, Shape: 10 - 1000 Hz, Mounting: sprung masses, 3 axes w/ 8 hr duration each)

Software

Sample Drivers	ROS, C++, Python

Reference

"OS1", *Ouster.com*, 2021. [Online]. Available: https://ouster.com/products/scanning-lidar/os1-sensor/. [Accessed: 20- Jan- 2022].