國立臺灣大學電機資訊學院資訊工程研究所

碩士論文

Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

適合自動駕駛車輛之結合邊緣資訊 即時影像語意分割系統

Real-Time Semantic Segmentation with Edge Information for Autonomous Vehicles

韓翔宇

SHIANG-YU HAN

指導教授：傅立成 博士　　共同指導：蕭培墉 博士

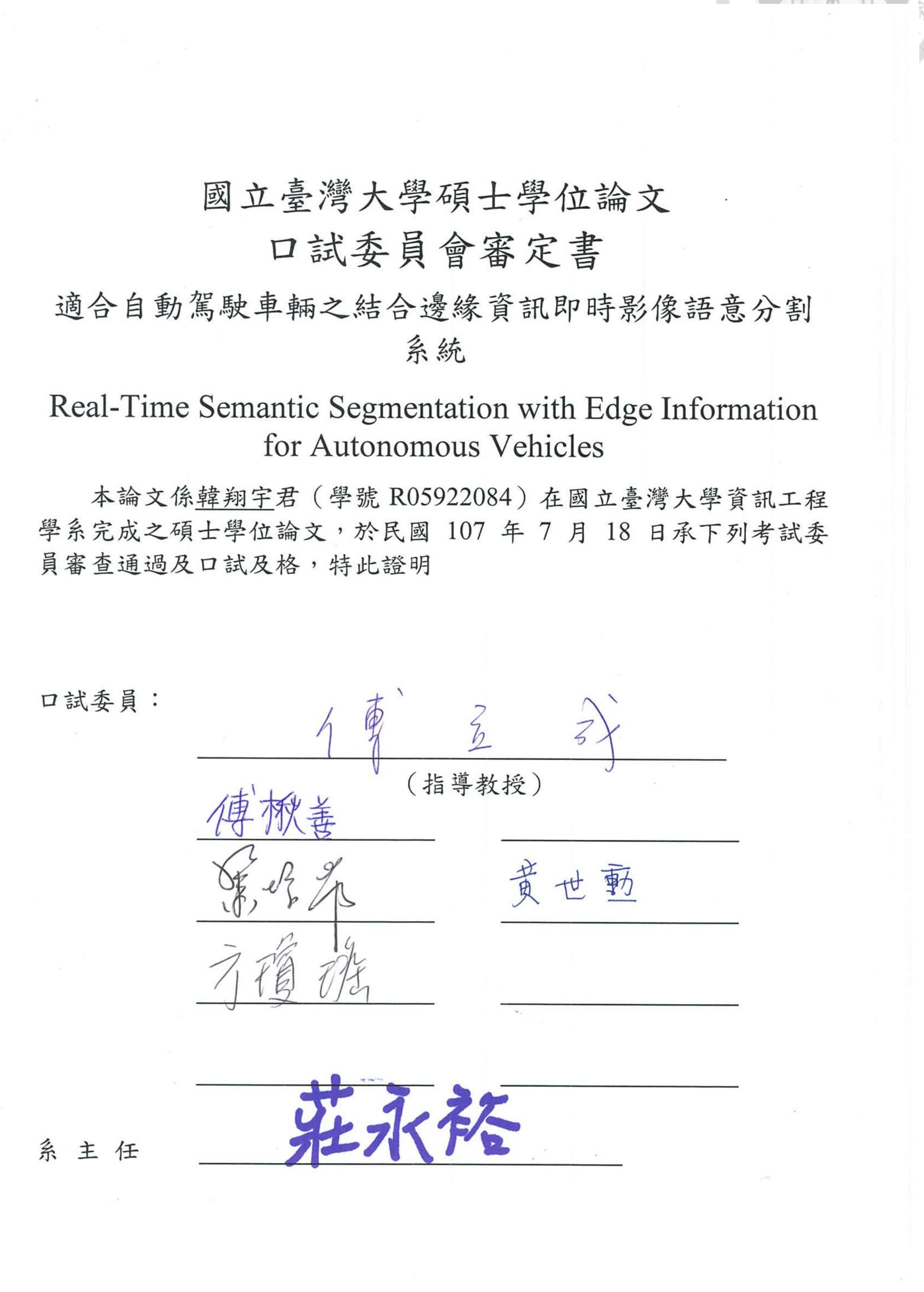
Advisor: Li-Chen Fu, Ph.D., Pei-Yung Hsiao, Ph.D.

中華民國 107 年 7 月

July, 2018

doi:10.6342/NTU201803607

口試委員會審定書



國立臺灣大學碩士學位論文

口試委員會審定書

適合自動駕駛車輛之結合邊緣資訊即時影像語意分割系統

Real-Time Semantic Segmentation with Edge Information for Autonomous Vehicles

本論文係韓翔宇君（學號 R05922084）在國立臺灣大學資訊工程學系完成之碩士學位論文，於民國 107 年 7 月 18 日承下列考試委員審查通過及口試及格，特此證明

口試委員：

傅立成

（指導教授）

傅楸善	
[illegible]	黃世勳
方瓊瑤	

系 主 任 莊永裕

doi:10.6342/NTU201803607

中文摘要

先進駕駛輔助系統 (ADAS) 包含兩項基本功能需求。首先是物件偵測功能，用於車輛行駛中避免碰撞障礙物或是路上行人。另外一項需求則是藉由影像切割功能找到車輛可以行駛的區域。有別於傳統影像切割方法，採用語意分割的深度學習網路架構，可以更正確辨識不規則的道路區域，指引自駕車行駛在更複雜的道路環境中。

近年來隨著卷積神經網路(CNNs) 的普及化，其功能已超越傳統以人工找出特徵的影像分割方法。 但是，卷積神經網路(CNNs)架構複雜，需要更多的處理時間與硬體效能需求，對於實作於車載處理系統的即時運用，尚有困難需要克服。 目前有一些方法被提出，例如 Enet，藉由刪除一些卷積層，達到更快執行速度，但卻犧牲影像切割的正確性。

本研究中，首先分析最先進的即時影像語意分割系統的輸出。 由這些輸出結果顯示，大多數被錯誤分類的像素，都是位於兩個相鄰物件的邊界上。基於此觀察，本研究提出一種新穎的即時影像語意分割網路系統，它包含一個類感知邊緣損失函數模塊與一個通道關注機制，旨在提高系統準確性而不損害運行速度。本研究以 Cityscapes 數據集評估所提出的方法，該資料集是目前公認最具挑戰性和權威性的道路語意分割數據集。評估結果顯示，在即時運作條件下，本研究的平均準確度超過 70%。

關鍵字: 深度學習, 卷積神經網路, 即時影像語意分割, 邊緣資訊

doi:10.6342/NTU201803607

ABSTRACT

Advanced Driver Assistance Systems (ADAS) consists of two basic functions. One is the Object detection for preventing vehicles from hitting pedestrians or other obstacles. The other is image segmentation for recognizing drivable areas and guiding the vehicle forward. For the latter, unlike those traditional image segmentation methods, image semantic segmentation based on deep learning architecture can handle the road areas better, guiding a vehicle to drive in a more complex environment.

With the popularity of Convolution Neural Networks (CNNs) in recent year, the traditional hand-crafted features methods have shown to be outperformed. However, deep CNN models are difficult to implement on vehicle application because the severe cost of time for complex processing. Although some proposed methods, such as Efficient neural network (Enet), achieved higher speed by removing some layers, it also led to the decrease of segmentation accuracy.

In this research work, we first analyze the output of state-of-the-art real-time semantic segmentation networks. The result shows that most of the misclassified pixels are located on the edge between two classes. Based on this observation, we propose a novel semantic segmentation network which contains a class-aware edge loss module and a channel-wise attention mechanism, aiming to improve the accuracy with no harm to inference speed. We evaluate the proposed method on cityscapes dataset, which is the most challenging and authoritative on-road semantic segmentation dataset. The results show that our proposed method can achieve over 70% mean IOU on Cityscapes test set under real-time requirements.

Keywords: Deep Learning, Convolution Neural Networks, Real-time Semantic segmentation, Edge Information.

doi:10.6342/NTU201803607

CONTENTS

doi:10.6342/NTU201803607

doi:10.6342/NTU201803607

doi:10.6342/NTU201803607

LIST OF FIGURES

doi:10.6342/NTU201803607

doi:10.6342/NTU201803607

LIST OF TABLES

doi:10.6342/NTU201803607

Chapter 1 Introduction

1.1 Motivation

Although the field of computer vision has existed for more than 50 years, it did not begin to thrive until the 1990s, the time in which the digital cameras were popularized. Since then, many researchers have devoted themselves to giving computers a visual understanding of the world. For feature description, researchers have proposed many different algorithms including the Scale-invariant feature transform (SIFT)[1] and the Histogram of oriented gradient (Hog)[2]. However, these methods highly rely on hand-crafted feature sets which are designed by researchers based on their own domain knowledge. With this limitation, methods based on hand-crafted feature are not only time-consuming but also tend to achieve unsatisfactory performance due to the content complexity of real environment. In recent years, thanks to the rapid development of graphics processing unit (GPU) technology, convolution neural networks (CNNs) have been applied to the field of computer vision. With many datasets which contain large amount of image materials published online, such as ImageNet[3], PASCAL VOC[4], and Cityscapes[5], CNNs can be trained to learn the representative feature from those images. Led by their independent learning ability, CNNs are able to surpass all of the hand-crafted feature based methods in performance.

doi:10.6342/NTU201803607

Despite that computer vision has been applied to vast kinds of fields, it plays an indispensable role in Advance Driver Assistance system (ADAS). The ultimate goal of ADAS is development of autonomous driving systems, which usually consist of several sub-modules to handle different tasks. In general, an autonomous driving system can be divided into three parts: the perception module, the decision module, and the control module[6].

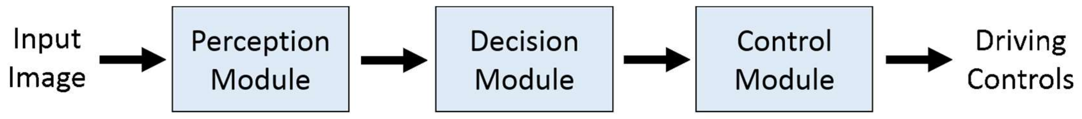


Fig. 1-1 Three main modules for an autonomous driving system

With the continuous improvement of the performance of autonomous driving system, the decision module requires more environmental information provided by the perception module to assist in decision making. However, ADAS equipped only with object detection system fail to provide accurate information. While detecting the drivable area, objet detection system output bounding boxes to frame those areas. However, because the drivable area is often in irregular shapes and cannot be perfectly fit into those rectangular bounding boxes, there can also be some obstacle existing in the boxes. The only way to overcome this problem is to precisely classify each pixel by incorporating semantic segmentation network into the perception module. That is why we can see that most of the large self-driving company in the market today have integrated the Semantic segmentation network in their products.

Unlike other fields of application, ADAS systems especially required high processing speed. If the system cannot process road information immediately in emergency situation and pass it to the decision-making module for judgment, it will be considered useless even if the accuracy of output is high. With this requirement in mind,

doi:10.6342/NTU201803607

we design our system to achieve Real-time speeds (30FPS[7]) with the GTX TitanX (Maxwell) GPU. There are two main reasons why we choose TitanX as a platform to evaluate the inference speed. The first reason is that because this GPU is widely used in other semantic segmentation as a platform for speed evaluation. The second reason is that the architecture of TitanX is the same as the widely used autopilot platform “Driver PX”.

Besides speed, accuracy is also very important for autonomous driving applications. To the best of our knowledge, the most authoritative on-road semantic segmentation record based on the data-set, Cityscapes, shows that the state-of-the-art result can achieve no higher than 70% mean Intersection-Over-Union (mIOU) under real-time speed. However, we believe that such performance can be improved by redesigning the network architecture.

Running under real-time environment means that the maximum computational resource which can be assigned to each input image is fixed. Under such limitation, how to design a network that can effectively allocate computational resources becomes an important factor to overcome the barrier of 70% mean IOU or to outperform the state-of-the-art.

Because the Deep CNN has a large amount of parameters which need to be trained, and the amount of high quality semantic segmentation labels are relevantly small. To deal with this problem, we usually pre-train our model on a big classification dataset (e.g. ImageNet[8]) to learn adaptive low level features, and then fine-tune the model on the target data. However, feature representations learned from classification data usually focus on main objects of image, so we may need to assist the network focus on other areas in fine-tuning stage to improve overall accuracy.

In this thesis, we propose a novel real-time semantic segmentation network which

doi:10.6342/NTU201803607

combines class-aware edge loss module and residual channel-wise attention mechanism, achieving both correct rate and real-time requirements. To validate our work, we evaluate it on Cityscapes[5] and KITTI on-road datasets[9] which contain urban road scenes. The experimental results show that our proposed class-aware edge loss and attention mechanism can improve the overall performance without harming inference time.

1.2 Related work

In this thesis, we propose a novel semantic segmentation network architecture which combine class-aware edge loss module and channel-wise attention mechanism, to improve the detection performance. In the following section, we will describe the evolution of CNNs and some state-of-the-art CNN based semantic segmentation network.

1.2.1 Convolutional Neural Networks for semantic segmentation.

In recent years, deep convolutional neural networks have become popular solution tools in almost all areas of computer vision[10-12]. All this started with Alexnet proposed by Krizhevsky *et al.* [13]. Alexnet won the first place of the ILSVRC competition in 2012. They added many new ideas in Alexnet, such as using Rectified Linear Units (ReLU) instead of sigmoid as an activation function. This new activation function would increase training speed and allow the network to converge faster. Since the traditional image datasets are too small, the data simply cannot reflect the real world. Deep learning method contains large number of trainable parameters, and the trained network will tend to be overfitted if the training set is too small. Fortunately, Li *et al.*

doi:10.6342/NTU201803607

published their large annotated images dataset, called ImageNet[3], openly, and the dataset contains more than 15 million manually-labeled high-resolution pictures. The total number of categories exceeds 20,000. With the rapid development of GPU technology at that time, CNNs based method quickly defeated traditional classifier, and become the best image classification method.

Because of the success of Alexnet in image classification task, people began to focus on CNN. Soon, researcher discovered that CNNs are robust feature extractor which can be widely applied to various tasks. By connecting different subnets, CNN can accomplish multiple image processing tasks. For example, in RCNN [14] an object detection network proposed by Girshick *et al.* use support vector machine (SVM) to classify the region features extracted by Alexnet to achieve object detection. Not only object detection, CNNs can also be used in semantic segmentation task. Fully Convolutional Network (FCN) [15] proposed by Long *et al.* is the first network which achieves semantic segmentation based on CNN. By replacing the fully connected layers with convolutional layers, FCN can provide spatial maps instead of 1D classification array. Those maps are upsampled using fractionally strided convolutions to produce dense pixel-wise prediction. Because FCN didn't contain any fully connected layers, the input image can be arbitrary sized. Some other methods try to refine the segmentation result by post-processing, such as DeepLab network[16] using fully connected factor graph to model the relation between each pair of pixels.

However, the aforementioned networks are running too slow. They cannot be used for tasks that have constraints on computational time. Enet[8] was proposed to address this problem. By reducing the size of feature maps at the beginning of the network, the total amount of computation is reduced. Enet can achieve 46.8 FPS at 1280x720 resolution. This greatly increases the range of application of semantic segmentation and

doi:10.6342/NTU201803607

leads a number of semantic segmentation networks design to achieve even higher inference speed.

1.3 Contribution

In this thesis, we focus on the output of the state-of-the-art semantic segmentation method and analyze the distribution of misclassified pixel. We also propose the Edgenet, which effectively reduces the number of misclassified pixels.

The major contributions of this thesis are listed as follows:

(a) The main contribution is to design a CNN based Semantic segmentation network. Combined with class-aware edge loss in training stage, our net is able to learn the edge information of instance without affect inference speed.

(b) The second contribution is that we propose a novel channel-wise attention mechanism, called residual squeeze-and-excitation(SE) module, to learn the importance of different channels without influencing network convergence.

(c) Combined with class-aware edge loss and residual SE module, our proposed Edgenet achieves over 70% mean IOU on Cityscapes semantic segmentation dataset under real-time speed.

1.4 Thesis organization

In Chapter 1, we have introduced the motivation of our work which aims to propose a novel real-time semantic segmentation architecture and achieve better result on Cityscapes dataset. We also discuss the problem to be improved we found. Finally, the contributions of this thesis are introduced. To summarize our research, the rest of thesis are organized as follows:

doi:10.6342/NTU201803607

In Chapter 2, we will present some background knowledge of deep CNNs. We will first introduce some basic modules of CNNs. Then, some CNNs architectures which are widely used as the encoder network for semantic segmentation will be explained. Finally, we will illustrate some state-of-the-art semantic segmentation network and training strategy which can reduce the amount of high quality segmentation data for training.

In Chapter 3, we will explain our methods in detail. We will first analyze the distribution of those misclassified pixels. Then, the overview of the novel semantic segmentation network proposed in this thesis, called Edgenet, will be introduced. After that, we will go deeper to the details of how to implement our proposed method.

In Chapter 4, we will introduce the public datasets first, which includes KITTI road detection dataset and Cityscapes semantic segmentation dataset. Then, we will present the experimental results of our proposed method. The visualization of the detection results of each dataset will also been shown.

In Chapter 5, we will conclude the contributions of this thesis and give a discussion on the experiments result. Finally, we will discuss the future work about how to further improve semantic segmentation accuracy.

doi:10.6342/NTU201803607

Chapter 2

Preliminaries

This study proposes a novel Edgenet architecture which combines edge information and channel-wise attention mechanism. Our proposed architecture is inspired by the Efficient Residual Factorized net (ERFnet)[17] which is based on factorized convolutional layers. In this chapter, we will first describe the convolutional neural network and other semantic segmentation network briefly.

2.1 Convolutional Neural Network

In recent years, CNN has been making a big splash in the field of image processing. Whether it is image classification[13, 18, 19], object detection[14, 20-22] or image segmentation[15, 23-25], CNN can be used to achieve the accuracy that cannot be achieved by traditional hand engraving methods. Unlike the traditional hand carving method, e.g. HOG [2], which requires researchers to design feature extractor based on their domain knowledge, CNNs can extract enough representative features from a large amount of training materials with its strong learning ability. Based on different purpose, each neural network will be designed with different loss function as a criterion for

doi:10.6342/NTU201803607

judging the performance of the network. The predict result and ground truth will be input to the loss function and the loss will be calculated. To continually minimize be the ultimate goal of training. When designing a neural network, the operation of each layer must conform the requirement of differentiable. This is because training the neural network requires the gradient[26] of each parameters,

All of these structures require considerable amount of computing resources. Thanks to the rapid development of GPUs in recent years. Computations in a convolution neural network can be practice on the GPU for parallelization. To further optimize the computing process and reduce the running time, series of methods are proposed, such as Kernel decomposition[18]. We would like to explain some of the layers and acceleration methods that are commonly used in convolutional neural network.

2.1.1 Convolutional Layers

Convolution as a feature extraction method has long been widely used in traditional image processing method. By performing a dot product with an input patch on the convolutional kernel, the intensity of the feature can be obtained. In CNN architecture, convolutional layers consist of a set of convolutional kernels, whose width and height are relatively small, but extend through the full depth of the input volume. Each layer may consist different number of kernels to generate different output dimension. During the forward pass, a two-dimensional feature map will be produced. We sometimes add zero-padding around the border of the image to make sure that the input will be consistent with the output in length and width dimension. By stacking the output map of all kernels, a three-dimensional feature maps (height, width, channel) will be produced. Each convolutional layer takes the extracts from the upper layer as the

doi:10.6342/NTU201803607

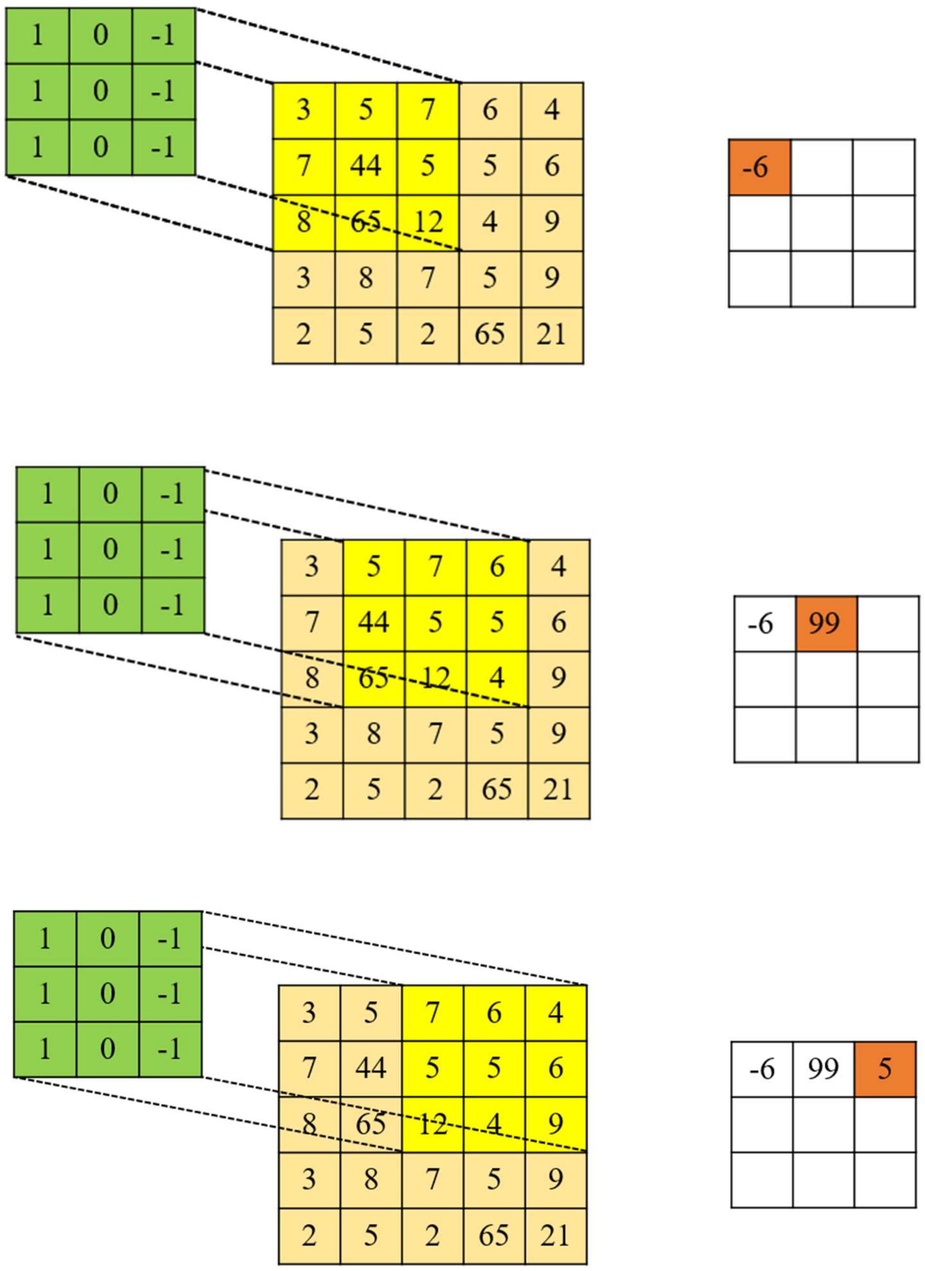
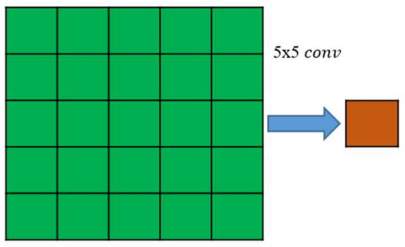


Fig. 2-1 Convolution operation.

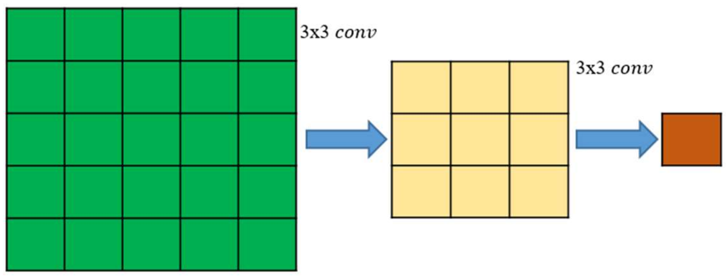
input to further obtain more abstract information. In general, the bottom convolutional layer will extract low-level features, such as color block or edge, while the top convolutional layer will extract high-level features, such as human face or vehicle.

doi:10.6342/NTU201803607

Although the learning ability of the convolution layer is powerful, its most criticized point is that it requires a lot of computing resources. This will be a fatal problem to applications that the time-limited requirement is very important to them. In order to solve the problem of high computational complexity, [18] proposes replacing the large kernel with a continuous small kernel volume base layer. From Fig. 2-2, it can be seen that after two layers of 3x3 convolutional layers are stacked, their receptive field will be able to reach the same size of a single 5x5 convolution layer. Table 2-1 shows the number of operations required for both when input size = 224x224x3, padding = 1, and output channel = 96. Obviously, the 5x5 convolution layer requires more computing resources than two consecutive 3x3 convolution layer.



(a) Single 5x5 convolutional layers.



(b) A stack of two 3×3 convolutional layers.

Fig. 2-2 The receptive field of convolutional kernels.

doi:10.6342/NTU201803607

Table 2-1 Calculation amount of convolutional layers

5x5conv	3x3conv
$5 \times 5 \times 3 \times (224 - 3 + 2 + 1)^2 \times 96 \times 2 = 7.1\text{M}$	$3 \times 3 \times 3 \times (224 - 3 + 2 + 1)^2 \times 96 \times 2 = 2.6\text{M}$

2.1.2 Pooling Layers

In the architecture of the neural network, since the deeper convolution layer is responsible for extracting more abstract and diverse features, the deep convolutional layers needs more kernels. In general, the number of kernels will grow exponentially with depth. The number of kernels directly affects the number of channels of output feature maps, so the amount of memory and computation required will also grow exponentially. This limits the depth of the network we can design, thus limiting the learning ability of networks. With the pooling layer, we can aggregate the information in a certain area to achieve the reduction of feature maps size. Max-pooling is the most commonly used pooling operation in a convolutional neural network. It keeps the maximum value in the region. Fig. 2-3 shows the result of Max-pooling acting on one slice of the channels. In addition to Max-pooling, Average-pooling is another commonly used pooling operator. Average pooling uses the average value in the region as the output. As shown in Fig. 2-4. the advantage of Average-pooling is that it can deliver messages more completely, but it also lost Max-pooling's ability to extract strong features.

doi:10.6342/NTU201803607

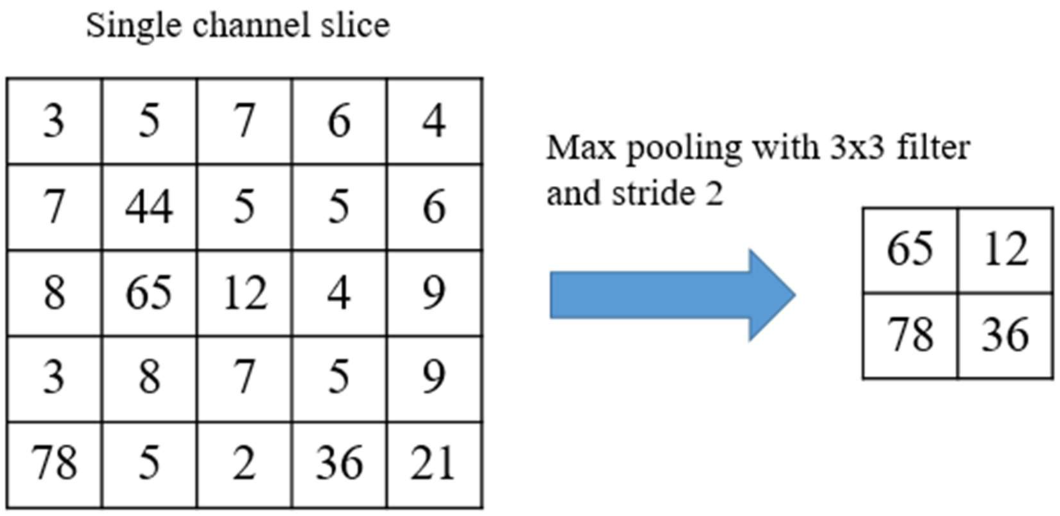


Fig. 2-3 Calculation of Max-pooling.

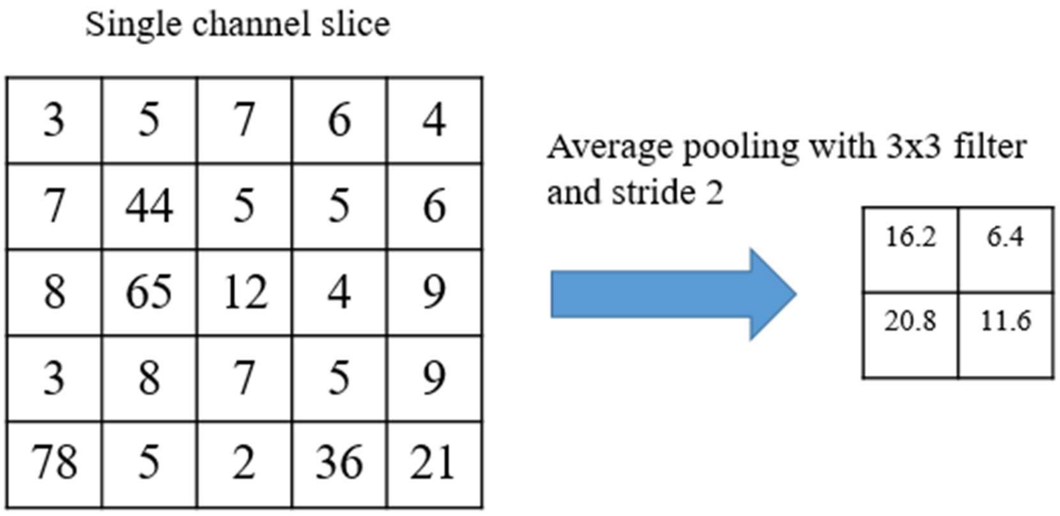


Fig. 2-4 Calculation of average-pooling.

2.1.3 Rectified Linear Unit (ReLU)

In the biological nervous system, the signal will be output after neurons have received stimuli that exceed the threshold value. In the artificial neural network, we use the activation function to achieve non-linear effects. Traditional networks usually use sigmoid (2-1) as activation function.

doi:10.6342/NTU201803607

$$\sigma(x) = 1/(1 + e^{-x}) \qquad (2\text{-}1)$$

To accelerate the training process with gradient descent base optimizer, Rectified Linear Units (ReLUs)[13] are applied to the deep convolutional neural networks, which is define as (2-2). When the input increase, the derivative of sigmoid function become small. This lead to gradually decrease of the gradient during back-propagating process, making the entire network slower to converge. On the contrary, ReLUs can run more efficiently. Because the differential value of ReLUs always maintains as one when input value greater than one, the network will converge quickly during training. By replacing sigmoid with ReLUs, a faster and more effective training of deep neural architectures can be achieved.

$$f(x) = max(0, x) \qquad (2\text{-}2)$$

2.1.4 Up-Sampling Layers

Unlike other images processing task, semantic segmentation needs to produce a high resolution output. Neural networks use the pooling to reduce the size of the feature maps when extracting feature. To output the same size as the input image, the semantic segmentation networks have to enlarge the low-resolution feature maps. Up-sampling layers are responsible for this work. We will introduce two commonly used up-sampling layers below.

doi:10.6342/NTU201803607

● **Indices Unpooling Layer**

Indices Unpooling[27] is a reverse operation of pooling which enlarges the information back to a patch area. By recording the locations of the maxima within each pooling region, unpooling layers will back-fills the value to this position, other position will be filled with zero. All unpooling layers has a corresponding pooling layer to provide pooling indices.

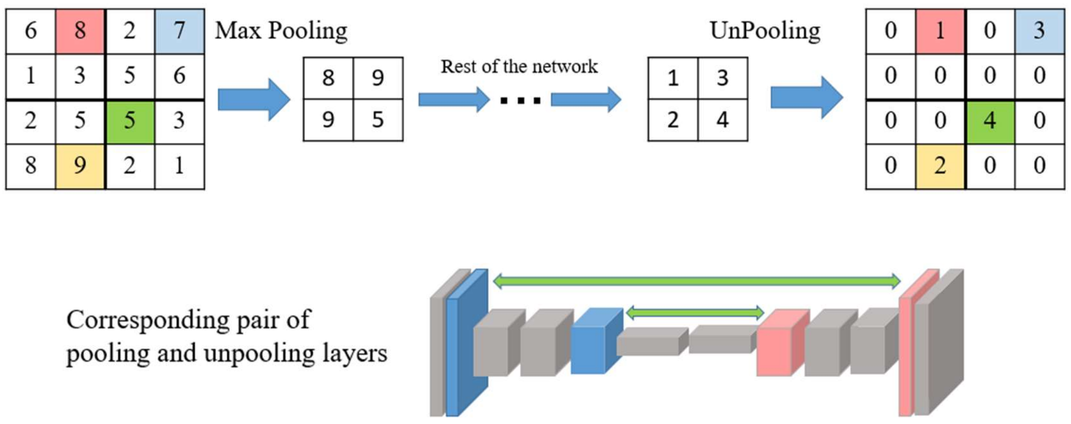


Fig. 2-5 Unpooling Layers

● **Fractionally Strided Convolutional layer**

Unlike unpooling layers, Fractionally Strided Convolutional layers contain trainable parameter, which means that it has stronger learning ability. In general, before regular convolution, it will insert zeros between the feature maps to magnify its size. Fig. 2-6 shows the process of feature maps enlarge by fractionally strided convolution layer.

doi:10.6342/NTU201803607

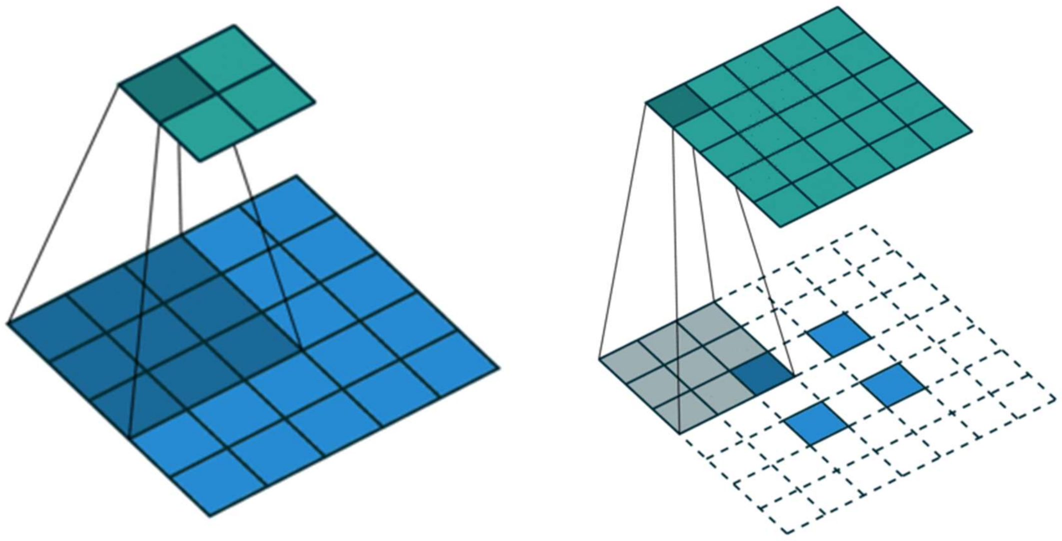

(a)Regular convolution (b) Fractionally Strided Convolution

Fig. 2-6 Fractionally Strided Convolution layer

2.1.5 Adam Optimizer

Adam optimizer[28] is usually used to optimize the CNN. Adam renew momentum m_t and velocity v_t with gradient g_t. Then optimize network parameter θ by $\mathrm{m_t}$ and v_t. The update values v_t, m_t, and the updated parameter θ_{t+1} are computed by following formulas,

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{2-3}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) {g_t}^2 \tag{2-4}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t} + \varepsilon} m_t \tag{2-5}$$

Where η is the learning rate and β_1, β_2 are decay rate of previous momentum and velocity. For training a CNN network, we follow the polynomial learning rate decay policy, which set an initial learning rate and decrease it as a polynomial curve define by ourselves.

doi:10.6342/NTU201803607

2.1.6 Alex-Net

Alex-Net[13] proposed by Krizhevsky *et al.* is the first CNN architecture which achieved an outstanding performance in ILSVRC-2012. Alex-Net contains five convolutional layers and three fully connected layers. Since the GPU had only 3G of memory in that time, the authors had to train the network on two GPUs separately. The structure of Alex-Net is shown in Fig. 2-7.

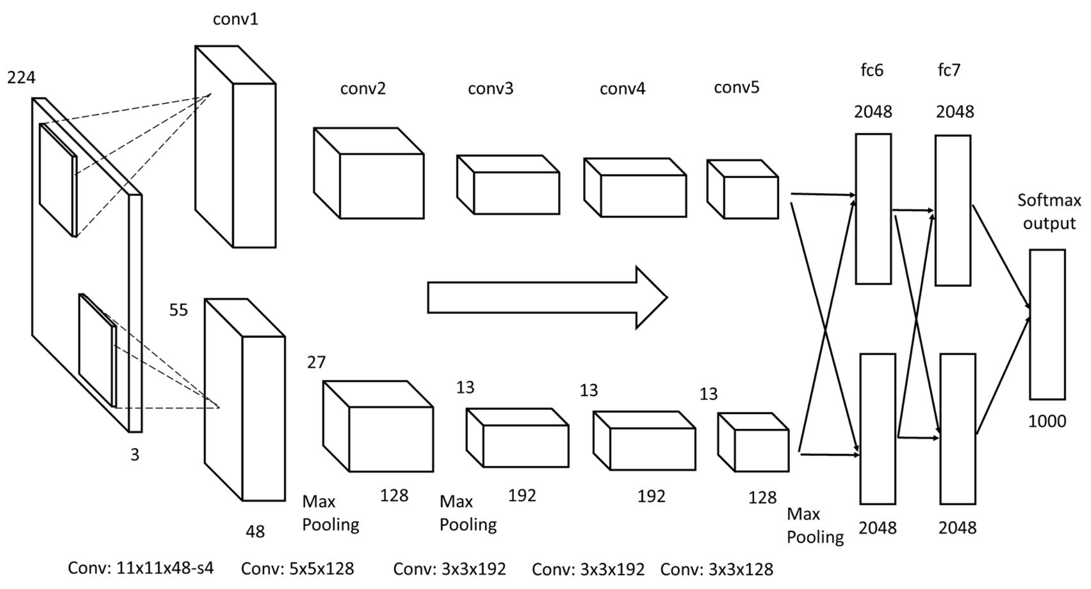


Fig. 2-7 Alex-Net [13]

2.1.7 Residual-Net

In recent years, the CNNs have gone deeper with developing direction. However, as the depth increases, the performance of the CNNs decrease gradually. At first, people thought that it was overfitting caused by having too much learning parameter. But that soon realized that the CNNs could not even converge on the training set, which shown to be a new problem. To deal with it, He *et al.* [19] introduced a new CNN architecture, called Residual network (Res-net), which make the deeper network to be trainable. They

doi:10.6342/NTU201803607

applied residual block to simplify the optimization process. The residual block is defined as:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x} \tag{2-6}$$

where $\mathbf{x}$ and $\mathbf{y}$ is the input and output of the layers. $\mathcal{F}(\mathbf{x}, \{W_i\})$ is the residual mapping to be learned. The residual learning is done by bypassing input feature maps and merge it with the output of convolution, which is formulated as $\mathcal{F} + \mathbf{x}$ operation. If an identity mapping were optimal, it would be easier to push the residual to zero than to fit an identity mapping by a stack of nonlinear layers. The residual block is shown as Fig. 2-8. As the result, they proposed several networks with different number of layers (e.g., Resnet-18, Resnet-34, Resnet-50 and Resnet-152) and showed that the performance of network can be improved by simply increasing the depth of the network.

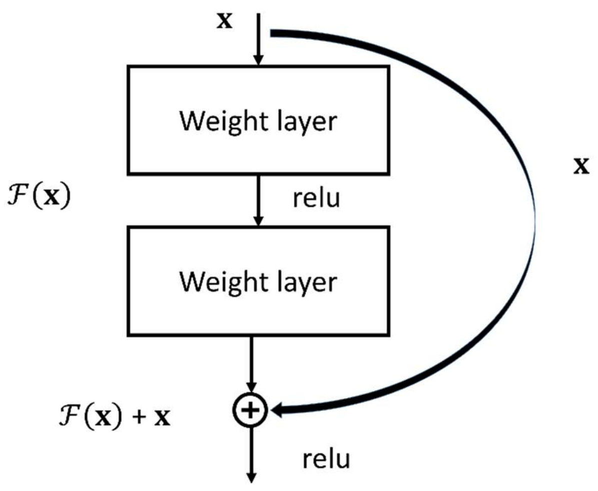

x
Weight layer
x
$\mathcal{F}(\mathbf{x})$
relu
Weight layer
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Fig. 2-8 Residual block.

2.1.8 Squeeze-and-Excitation(SE) Net

SE-net[29] proposed by Jie Hu, Li Shen and Gang Sun is the winner of ILSVRC [8] 2017 competition. They apply channel-wise attention mechanism in their network. With the attention mechanism, CNNs can effectively direct the computing resources to most

doi:10.6342/NTU201803607

informative regions. In convolution process, the importance of features filtered out by each kernel are different. To use this characteristic, they proposed Squeeze-and-Excitation module weighting each channel with their importance. The SE-module is shown as Fig. 2-9. After regular convolution, they use global average pooling to form a vector containing global information, each channel will be average into a value. After global average pooling, a HxWxC feature maps will become a 1x1xC vector. Then they apply two fully connected layers around the nonlinear activation function to fully capture channel-wise dependencies, they set reduction ratio r as 16 to achieve an effective trade-off between model complexity and performance. Before channel-wise multiply back to feature maps, they apply a sigmoid activation function to increase nonlinearity.

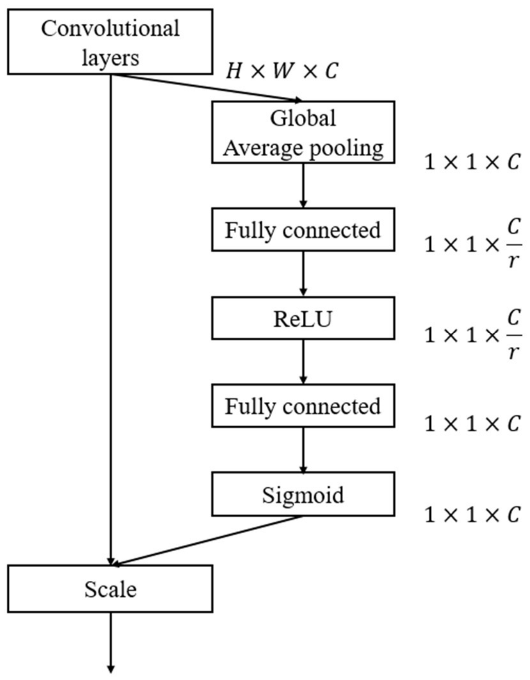


Fig. 2-9 Squeeze-and-Excitation module

doi:10.6342/NTU201803607

2.1.9 Fine-tuning

Training deep CNN from scratch requires a large amount of data, it is hard for researchers to collect enough labeled images for each specific tasks. Especially for jobs that require large pixel-wise labels such as semantic segmentation, it is even more difficult to collect a large number of labels. Fortunately, according to [30], no matter what the task is, the features captured by shallow convolutional layers are shareable. Combined with the concept of transfer learning Fig. 2-10. We can first pre-train on the image classification dataset, e.g. ImageNet, and replace the fully connected layer for classification with the decoding network of semantic segmentation. Fine-tune segmentation network requires only a relatively small amount of data.

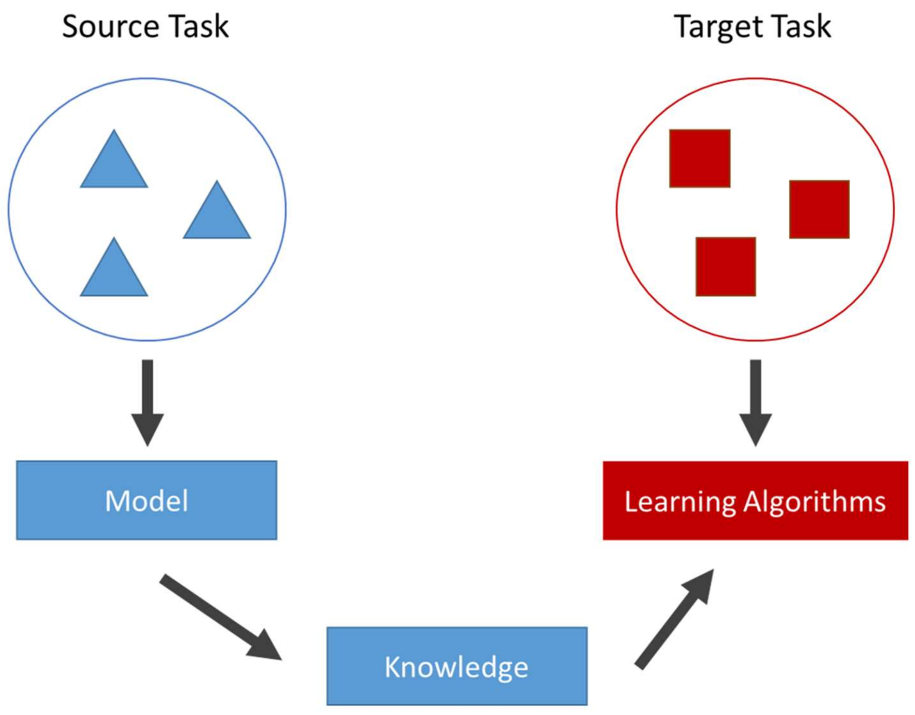


Fig. 2-10 Paradigm of transfer learning

doi:10.6342/NTU201803607

2.2 Semantic segmentation Network

Before we introduce our proposed novel semantic segmentation network, we will present other semantic segmentation architectures first. Fcn[15] is the first approach trying to solve semantic segmentation problem with fully convolutional network, and then Segnet [23] is a Symmetrical Encoder-decoder CNN can run at near real-time speed (about 16 FPS).

2.2.1 Fully Convolutional Network

In order to achieve pixel-wise classification, CNNs have to keep the spatial information of specific features. However, fully connected layers are usually added at the end of the network in traditional classification CNNs. This will remove spatial information and restrict the size of the input image. Long et.al. resolve this problem by replacing fully connected layers with convolutional layers containing the same number of channels. Fig. 2-11 shows the architecture of FCN.

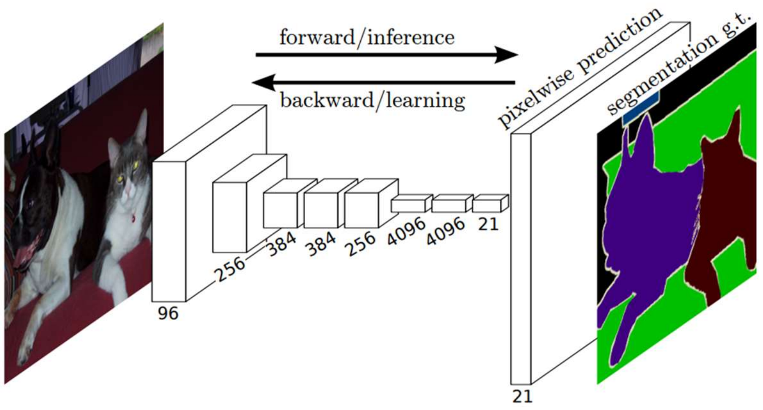


Fig. 2-11 Architecture of FCN[15]

To enlarge feature maps to the size of input image, they apply fractionally strided convolutional layer, also known as deconvolution layer, to the end of their network. In this study, they found that segmentation detail will be improved by fusing information

doi:10.6342/NTU201803607

from lower layers. It’s because feature maps from lower layer will keep more fine-grained information. By replacing fully connected layer with convolutional layers, their architecture can be applied to almost every classification CNNs.

2.2.2 Segnet

Segnet[23], as a network for semantic segmentation, uses a symmetric encoder decoder architecture. The first half of the network are similar to the front 13 layers of VGG-16 net. As a encoder, They are responsible to collect enough representative information. The decoder network architecture in the latter half is exactly the same as the encoder network. Only the pooling layers which used for downsampling are replaced with the unpooling layers. The pooling indices needed during unpooling are provided by the corresponding pooling layer in the encoder. Compared with Fully Convolutional Net, Segnet does not have lots of channel within the convolutional layers, it can achieve 16FPS on GTX TitanX GPU with Maxwell architecture.

The speed which can be considered almost real-time allows Segnet to be widely applied to many real world application. However, still, its inference speed does not meet the requirement of autonomous driving application. Fig. 2-12 shows the architecture of Segnet.

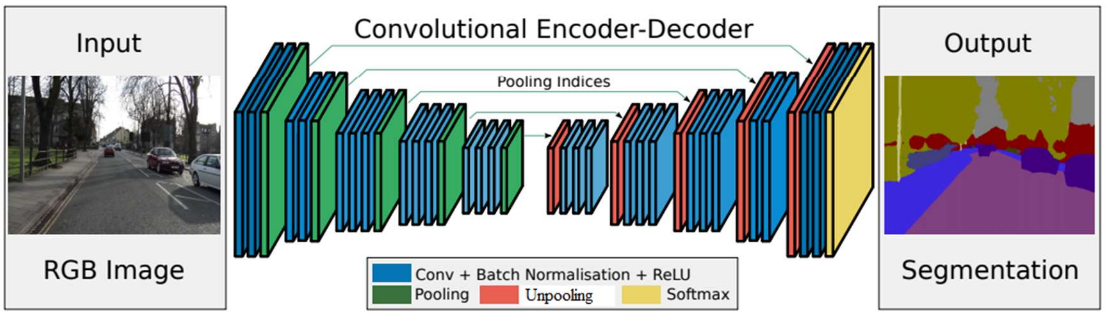


Fig. 2-12 Architecture of Segnet[23]

doi:10.6342/NTU201803607

Chapter 3

Edge Net

In this chapter, we propose a novel real-time semantic segmentation network, which combines class-aware edge loss module and residual channel-wise attention mechanism, achieving both high accuracy and real-time requirements. First, we analyze the distribution of pixels which are misclassified by state-of-the-art real-time semantic segmentation network to identify potential weakness. Next, we introduce our proposed CNN-based Semantic Separation Network, called Edgenet, which aims to improve accuracy using Residual Squeeze-and-Excitation (SE) block and class-aware edge loss modules by allocating computing resources to areas that contain rich information.

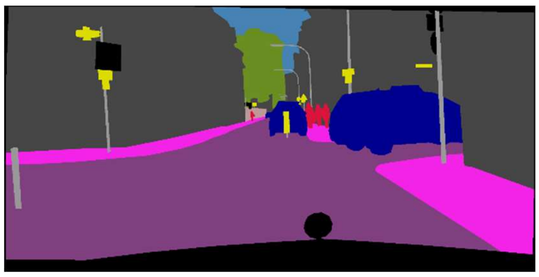
3.1 Observation

In order to achieve higher mean IOU without jeopardizing the inference speed, the strategy is to force our network concentrating on those difficult areas. From the former argument, we here choose ERFnet proposed by Romera *et al.* [17] as means for analysis because it is the best semantic segmentation algorithm which can run under real-time speed to the best of our knowledge. First, we evaluate ERFnet with Cityscapes dataset[5]

doi:10.6342/NTU201803607

which contains numerous high resolution semantically segmented urban scenes. Following their training policy, we first pre-train the encoder with ImageNet dataset, and then fine-tune the whole net on Cityscapes dataset. Finally, the mean IOU we achieve is as high as 72% on Cityscape validation set which is close to their 71.5%. The visualization result of ERFnet are shown in Fig. 3-1. By observing Fig. 3-1 (c), we can see that most of the misclassified pixels fall on the boundaries between adjacent categories, which is what we call the edges. Based on this observation, we decide to analyze in detail the distribution between misclassified pixels and the edge.

(a) Input image


(b) Ground truth

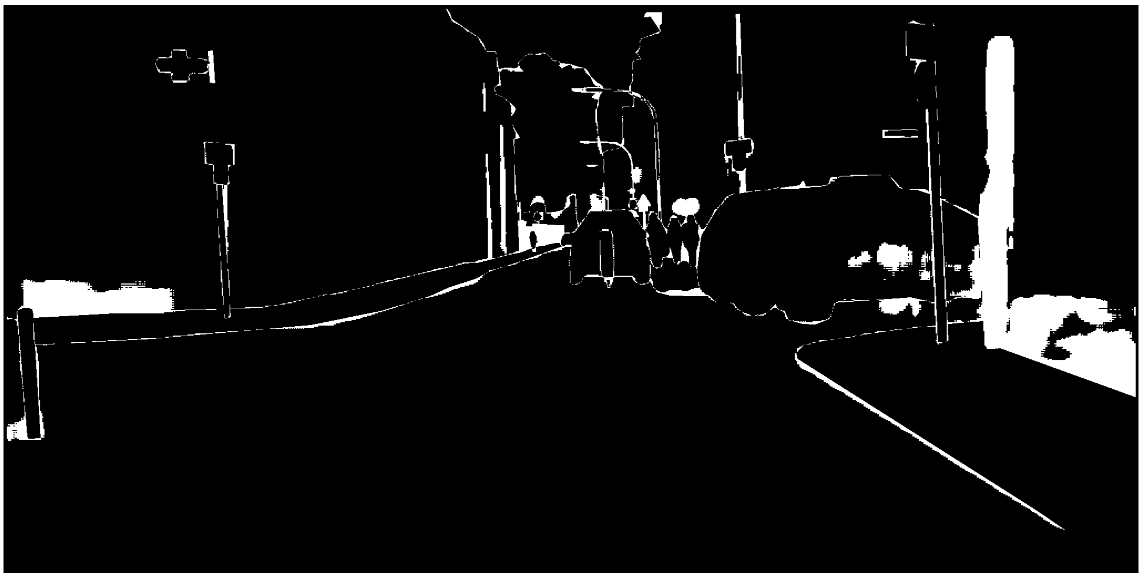

(c) Pixels been misclassified by ERFnet

Fig. 3-1 Visualization of ERFnet result

The statistical results we presented in Fig. 3-2 show that nearly 60% of misclassified pixels are located within 10 pixels from the edge. We believe that there are

doi:10.6342/NTU201803607

two reasons for this phenomenon. The first is that there may be some slight errors during the image marking process conducted by human manually. The second reason is that the edges usually occupy too few pixels in the image. While in a typical semantic segmentation, the loss function doesn't discriminate pixels on edge between different classes. This makes the network focus on a large area of classification but ignores the correctness of the pixel near the edge.

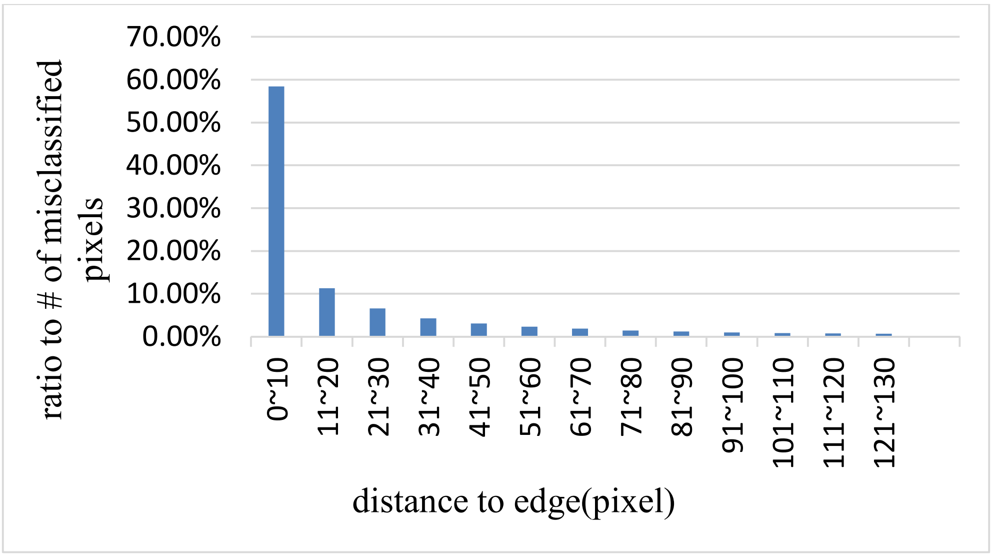


Fig. 3-2 Histogram of misclassified pixels

3.2 Edgenet Overview

Edgenet is a network that can be trained end-to-end. We add our proposed residual SE-block after each Convolutional layer to reweight each channel with their importance. At the end of the network, we combine the class-aware edge loss module to improve the segmentation result near to edge. The proposed Edgenet is shown in Fig. 3-3.

doi:10.6342/NTU201803607

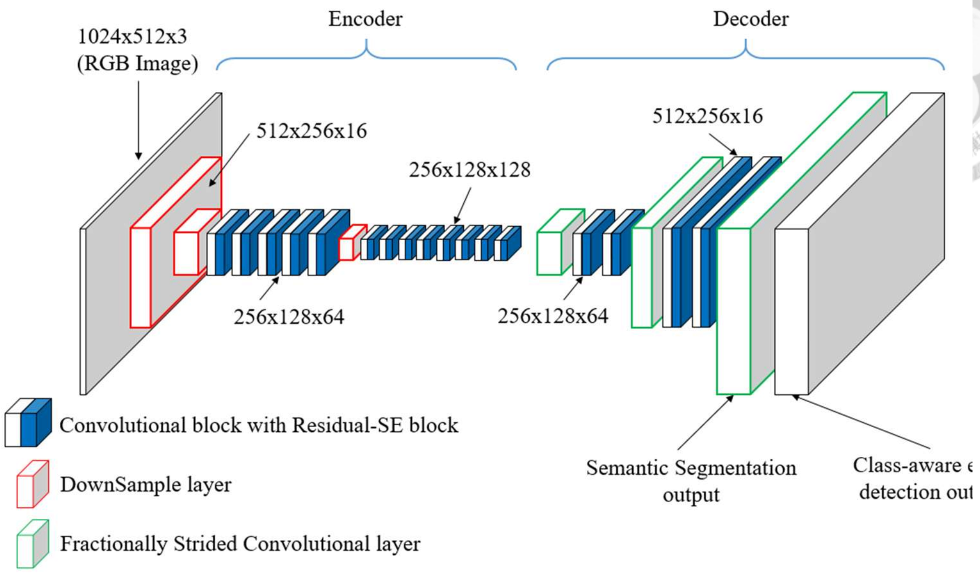


Fig. 3-3 Overview of our proposed Edgenet.

3.3 Edgenet Design

In this section, we will go into details about the design of our proposed Edgenet architecture. Edgenet contains a set of asymmetric subnetworks with encoder and decoder. We will first explain the structure of encoder and decoder subnetworks, and then the class-aware edge loss module we proposed will be introduced to decrease the segmentation error on the edge area. Finally, we will explain the residual SE-block applied after every convolution layer.

3.3.1 Encoder subnetwork

Fig. 3-3 Overview of our proposed Edgenet.Fig. 3-3 shows the overview of Edgenet. Our net includes three sets of downsamplers. The first downsampler performs max pooling and convolution with stride 2. The image size is reduced from 1024 x 512 to 512 x 256. The second set of downsampler also executes max pooling and strided

doi:10.6342/NTU201803607

convolution, continuously reducing the size of the feature maps from 512 x 256 to 256 x 128. The feature maps are quickly downsampled twice in the initial stage of the encoder network, results in only a quarter of height and width length compared to the original image. With this process, the computational complexity of subsequent five residual convolutional building blocks will be reduced. The last group of downsamplers also performs max pooling and convolution, reducing the feature maps size from 256 x 128 to 128 x 64, followed by 8 consecutive convolutions which have different dilated ratio.

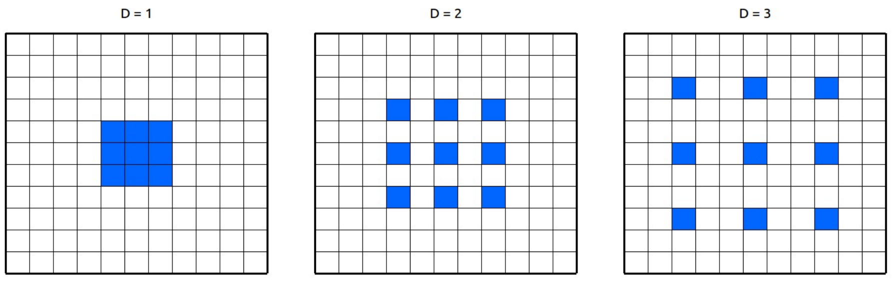


Fig. 3-4 Kernel of dilated convolution

The dilated convoluted [16] kernel is shown in Fig. 3-4. The use of dilated convolution can bring the benefits of increasing the reception field.

In the past, in order to increase the reception field, pooling[31] is used to summarize the information in the area. However, the pooling leads to the loss of spatial information. For network like semantic segmentation whose output is pixel-wise prediction, using too much pooling layers to enlarge receptive field will result in the impossibility to reproduce fine-grained details. The size of feature maps will shrink too much after pooling layers and the spatial information will be lost. By using dilated convolution, we can keep the spatial information while enlarging the receptive field without increasing the amount of computation complexity and the amount of parameters.

doi:10.6342/NTU201803607

Overall, the feature maps output by the encoder is only one-eighth of size compared to the original image. The downsampler, as shown in Fig. 3-5, uses a parallel architecture with stride convolution and pooling, then concatenates output featuremaps along channel dimension. This concept was first proposed in the inception v2[32]. By applying this method, the two problems, the loss of detail information caused by pooling before convolution, as well as the high computational cost caused by convolution before pooling, can both be solved. Besides, the Stride Convolution in parallel with pooling also help to increase reception field while reducing the amount of computation in the encoder sub-network as a whole.

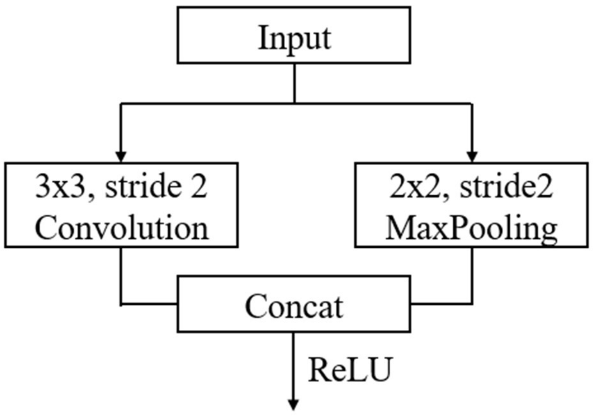


Fig. 3-5 Downsampler

Convolutional residual building block in our Edgenet is based on the non-bottleneck design[19], which is composed by two 3x3 convolutions. To reduce computational complexity, the 3x3 convolution is factorized into two continuously one-dimensional convolutions using the method proposed by Romera *et al.*[33]. An addition ReLU is placed in the middle to increase the nonlinearity.

doi:10.6342/NTU201803607

3.3.2 Decoder subnetwork

The second part of our Edgenet is the decoder network. The goal is to enlarge the low-resolution feature maps extracted by encoder network back to high-resolution multi-dimensional pixel-wise classification results. Because the resolution of input feature maps is one-eighth of input picture, decoder also contains three upsampler layers. We use fractionally strided convolutional layer to implement upsampler layer. The advantage is that compared to unpooling layer, there will be more trainable parameters in fractionally strided convolutional layer. To give the network better learning ability, we applied two convolution layers between two adjacent upsampler.

3.3.3 Class-aware edge loss module

In Section 3-1, we analyze the distribution of those misclassified pixels. Most of them are located near the edge. In order to properly classify these pixels, we need to let our encoder focus on these areas. There are two ways to achieve this. The first way is adding additional extractor of edge information into encoder network, such as[34]. This may bring up two problems. The first one is the increased computational complexity caused by additional edge extractor. The second is the extra training time of an edge extractor we may need to spend. However, the entire network cannot be trained end-to-end. The second approach is to design an extra loss function to drive our network converge in a direction that put more attention on edges. Since one of the main goals of our network design is to maintain the speed of real-time operation. We prefer to choose the latter approach. The general segmentation loss usually implemented with 2D cross entropy as shown in equation (3-1). Where W, H are the width and height of output feature map and $G(x,y)$, $P(x,y)$ are one-hot vector of ground-truth label at position (x,y) and prediction of segmentation at position (x,y). This loss function does not

doi:10.6342/NTU201803607

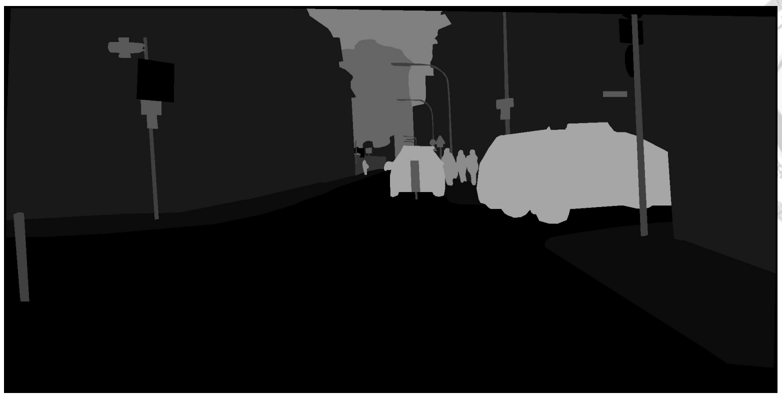
discriminate pixels on edge between different classes, it treats every pixel equivalently as dealing with classification problem, and then adds up all cross-entropy loss of every pixel.

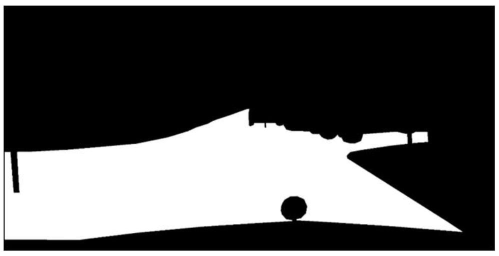
$$Loss_{seg} = -\frac{1}{N}\sum_{x=1}^{W}\sum_{y=1}^{H} G(x,y)\log(P(x,y)) \qquad (3\text{-}1)$$

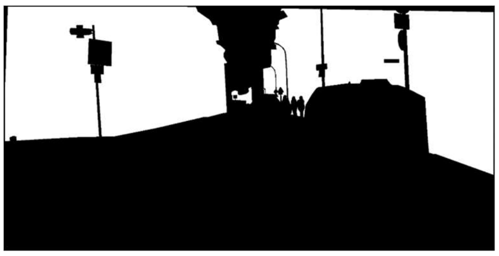
In order to increase the weight of the edge in the loss, we need to first find the location and category of the edge from the ground truth. Preprocessing steps are as follows.

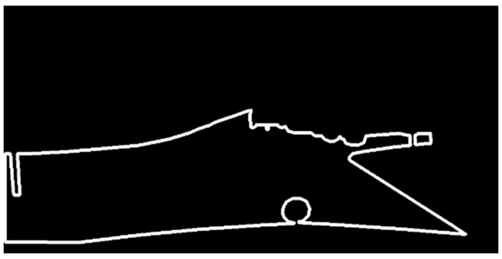
- generating binary images of each class from ground truth image. Fig. 3-6 (b, c) show the binary image of road and building classes;
- finding edge in each binary image. Fig. 3-6 (d, e) show the edge imag of road and building classes;
- merging all edge-relevant binary images into a class-aware edge ground truth, where each pixel is labeled with their original class id, as shown in Fig. 3-6 (f).

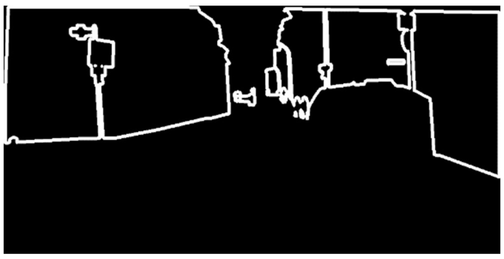
doi:10.6342/NTU201803607


(a) Ground truth image


(b) Binary image of road


(c) Binary image of building


(d) Edge of road


(e) Edge of building

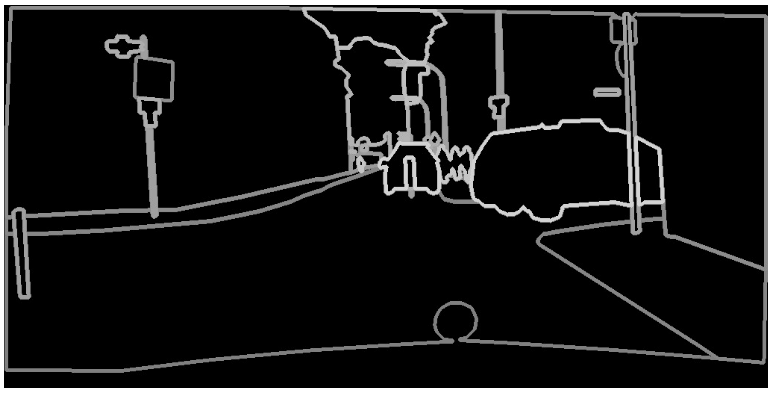

(f) Class-aware edge ground truth

Fig. 3-6 Preprocessing edge ground truth

doi:10.6342/NTU201803607

In order not to affect the overall inference speed, we place the additional layer related to the edge after the semantic segmentation result. Fig. 3-7 illustrates our design.

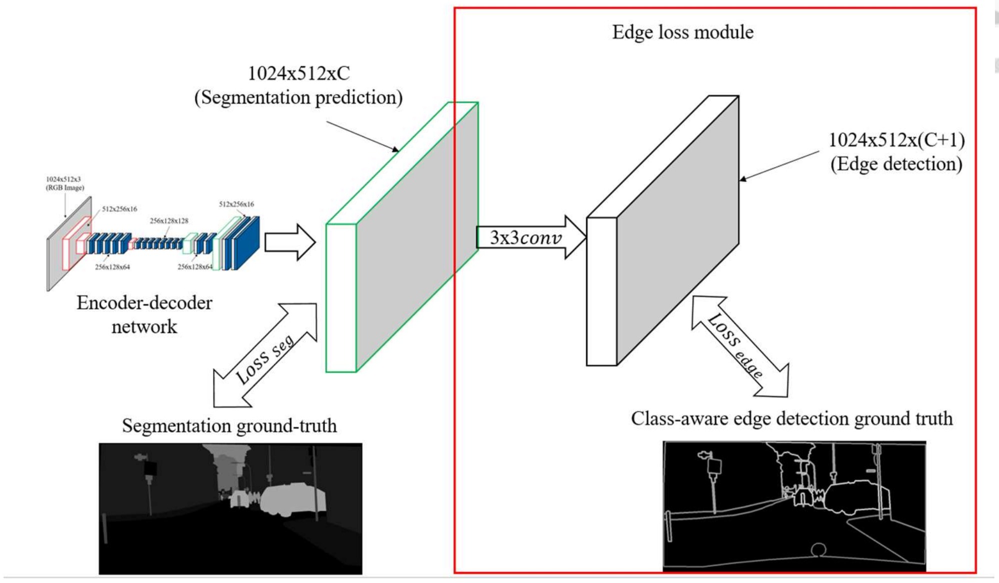


Fig. 3-7 Edge Loss module

A 3x3 convolutional layer is located after semantic segmentation to generate edge detection from segmentation result. Then, the edge loss will be calculated based on edge detection result and edge ground truth.

If the segmentation result is good, the associated 3x3 convolutional layer will be sufficient to generate good edge prediction similar to edge ground truth. If the edge detection result is very different from edge ground truth, which indicates that the semantic segmentation does not contain correct edge information, the gradient of the $loss_{edge}$ will also be directly backpropagated to semantic segmentation network, forcing it to extract more edge information to reduce $loss_{edge}$.

$G_{seg}(x,y)$ and $P_{seg}(x,y)$ are ground truth of segmentation and edge detection at position (x,y), where $G_{edge}(x,y), P_{edge}(x,y)$ are prediction of segmentation and edge detection at position (x,y). Overall, we sum up the loss of original segmentation

doi:10.6342/NTU201803607

described by equation (3-3) and the edge loss by equation (3-4), and take it to be the total loss function shown by equation (3-2) of our network, where λ is the hyper parameter, used to balance the importance between original segmentation loss and edge loss.

$$Loss_{total} = Loss_{seg} + \lambda * Loss_{edge} \tag{3-2}$$

$$Loss_{seg} = -\frac{1}{N}\sum_{x=1}^{W}\sum_{y=1}^{H} G_{seg}(x,y)\log\big(P_{seg}(x,y)\big) \tag{3-3}$$

$$Loss_{edge} = -\frac{1}{N}\sum_{x=1}^{W}\sum_{y=1}^{H} G_{edge}(x,y)\log\big(P_{edge}(x,y)\big) \tag{3-4}$$

We also apply the class weighting technique proposed in [24] to deal with class imbalance problem, where the weight of each class is formulated as in equation (3-5).

$$\omega_{class} = \frac{1}{ln(1.2 + P_{class})} \tag{3-5}$$

With P_{class} referring to that, among all pixels, what proportion does pixels in this class take up. Our edge loss module does not participate in the segmentation prediction process. It only helps the network focus on the classification correctness of the pixel near the edge. Since our edge loss module will not be involved in the testing stage, the overall inference will not be influenced.

3.3.4 Residual SE-block

To properly allocate the computing resources is an important strategy for us to improve accuracy without affecting inference speed. In a broad sense, attention can be viewed as a factor to bias the allocation of available processing resources toward the informative area of the input. We design the attention mechanism in our network for

doi:10.6342/NTU201803607

better resource allocation. The feature maps of semantic segmentation are three-dimensional. We propose an edge loss module to force our network focus on the edge in width-wise and height-wise spatial dimensions. For the third dimension, channel, the importance of each feature maps are different. By strengthening the important channels and suppress less useful ones, the representational power of the network will be enhanced. Different from the spatial dimension, there is no strong information, such as "edge", to indicate the importance. The Squeeze-and-excitation net[29] mentioned in preliminary cleverly uses the extra fully connected layer to learn the importance of each channel. After we have implemented the SE-block in our network, the performance of the network decreased. The rise of network learning ability has resulted in the performance decrease, where the same situation has also occurred in residual net proposed by He *et al*. [19]. Inspired by [29] and [19], we have modified SE-block as a Residual SE-block structure as show in Fig. 3-8.

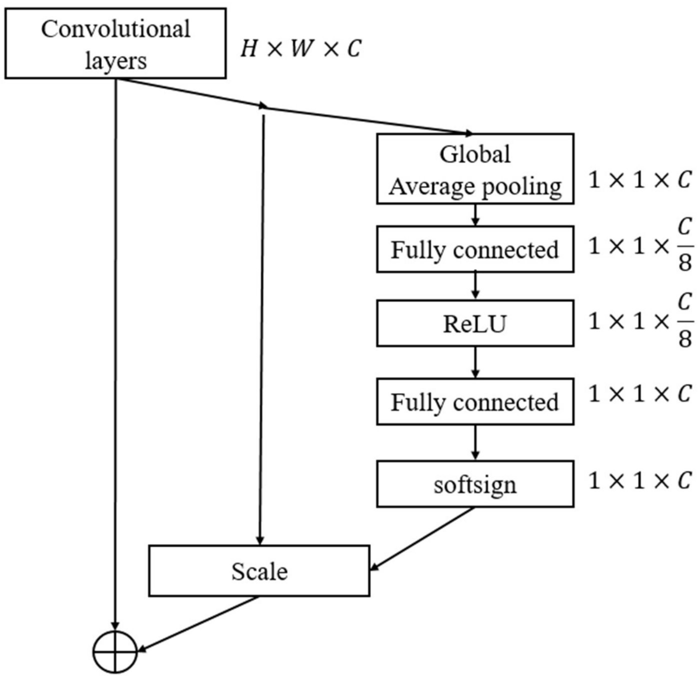


Fig. 3-8 Residual SE-block

doi:10.6342/NTU201803607

We perform shortcut connection from input of SE-block to the output, and apply element-wise addition on two feature maps channel by channel. Because the element-wise operation is simple without need to training any parameters, the network can gain the feature representations from previous layers without extra computation complexity. If these channels do not need to be weighted, it would be easier to push the residual branch to zero than to fit an identity mapping by a stack of fully connected layers. Based on the concept that residuals may contain negative values, we replace sigmoid activation function in residual SE-block with SoftSign as shown in equation (3-6).

$$f(x) = \frac{x}{1 + |x|} \tag{3-6}$$

We apply residual SE-block after each convolution block to perform feature recalibration.

doi:10.6342/NTU201803607

Chapter 4 Experiments

Our proposed method has been introduced in chapter 3. In this chapter, we will evaluate our method by conducting several experiments.

First, we will describe each dataset and how we evaluate the performance. Then, we will show the experimental results on each dataset, including the comparison with other state-of-the-art methods.

4.1 The Datasets

4.1.1 Cityscapes Dataset

Cityscapes [5] is the most challenge and authoritative dataset for on-road semantic segmentation nowadays, which is our main experimental target. Cityscapes dataset has 30 different classes which belong to the eight major category.

In order to ensure the diversity of the dataset, the images in Cityscapes were taken from 50 cities in Germany, in all seasons except winter. All the photos are manually selected and marked. All of the images meet the conditions of containing a large number of moving objects and various backgrounds.

The image of Cityscapes dataset is divided into two parts. The first part is the fine annotations, and the second part is the coarse annotation. Fine annotations are images

doi:10.6342/NTU201803607

with careful labels. There are 5,000 fine annotations, including 2,975 training images, 500 validation images, and 1,525 test images. The training and validation sets are provided with ground truth image while the testing set only contains the original image.

Coarse annotations are images with approximate outline of the object. Cityscapes provide a set of 20,000 images. The class and the categories of image from Cityscape are listed in Table 4-1.

Table 4-1 Class of Cityscape dataset. (**Bold** words indicate Categories of the column. Red words indicate those classes be ignored in evaluation stage.)

Flat	**Construction**	**Nature**	**Vehicle**	**Sky**	**Object**	**Human**	**Void**
Road	Build	Veget	Car	Sky	Pole	Person	Static
Sidewalk	Fence	Terrain	Bicycle		Traffic sign	Rider	Ground
Parking	Wall		Bus		Traffic light		Dynamic
Rail track	Bridge		Truck		Pole ground		
	Tunnel		Train				
			Motorcycle				
			Caravan				
			Trailer				

The resolution of each images in Cityscapes dataset is about 2048x1024. In addition to the images, Cityscapes also provides other metadata such as GPS coordinates, corresponding right stereo views, Ego-motion data from vehicle odometry, and outside temperature from car sensor. Some sample images and ground truth images are shown in Fig. 4-1.

doi:10.6342/NTU201803607

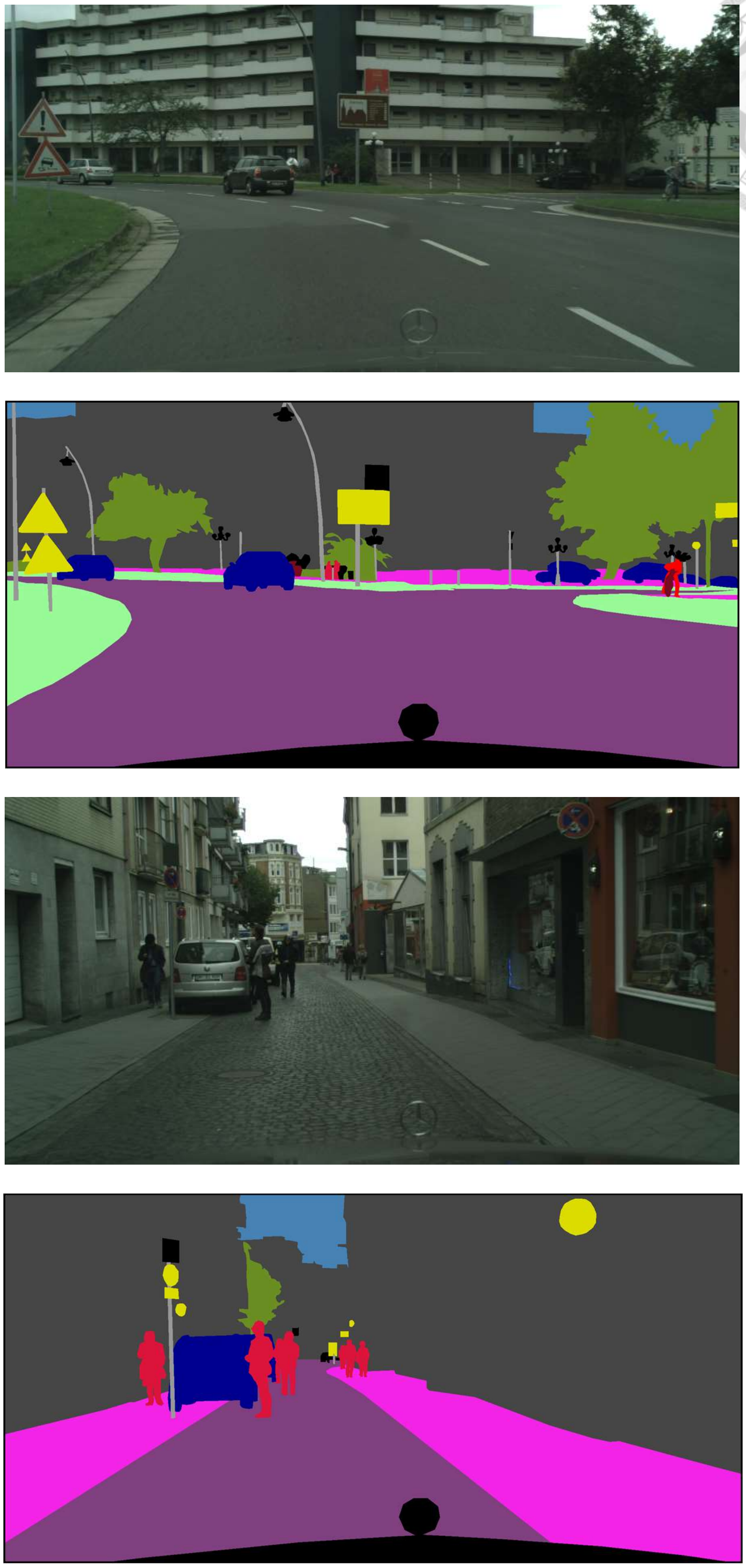


Fig. 4-1 Example images and annotations from the Cityscapes dataset.

doi:10.6342/NTU201803607

4.1.2 KITTI On-road Dataset

KITTI [9] dataset is an Vision benchmark which focus on on-road scene scenario. There are many different tasks in this dataset, e.g., optical flow, object detection, road detection. As we have claimed, our system can be used in the ADAS as a module to detect the drivable area. For verification, we use KITTI road detection task in the dataset to evaluate our system. The road detection benchmark contains 289 training images and 290 testing images. The image contains three scenes, urban unmarked, urban marked and urban multiple marked lane. The image resolution in KITTI dataset is 1242x375. We will describe it with more detail in the following section. Some sample images and ground truth images are shown in Fig. 4-2.

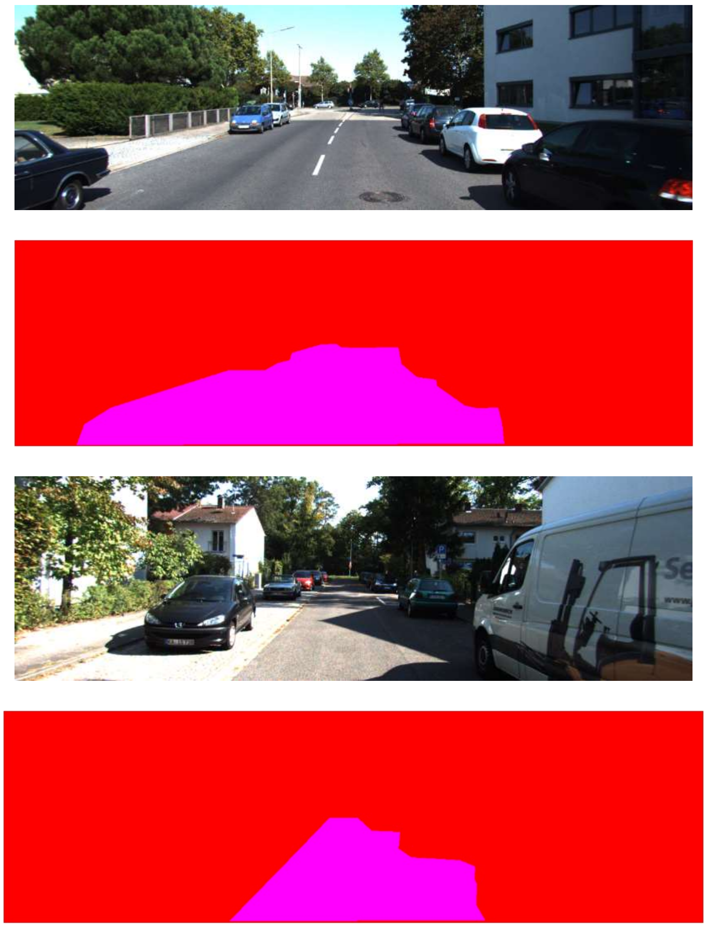


Fig. 4-2 Sample Images of KITTI dataset.

4.2 Experiment platform

Our method is trained and tested on personal computer with single NVIDIA GTX Titan X GPU (Maxwell). The specification of our experiment platform is listed in Table 4-2.

Table 4-2 Our PC specification.

Resources	Specification
CPU	Intel Core i7-2600 3.4GHz
Memory	32 GB
Operation System	Ubuntu 16.04
Graphic Processor Units	NVIDIA GTX Titan X (Maxwell)

We implement our method based on PyTorch, which is a deep learning framework developed by Facebook. PyTorch integrates NVIDIA Cuda and cudnn toolkit, which allows us to utilize strong GPU acceleration. We train and test our model using Cuda 9.0 and cudnn 7 with PyTorch 0.4.0.

4.3 Experimental Results

4.3.1 Inference time

We compare the inference time to other state-of-the-art semantic segmentation network. To fairly compare the running time, the speed of each method is measured on Cityscapes 19 classes segmentation with batch one configuration. The comparison result is summarized in Table 4-3.

doi:10.6342/NTU201803607

Table 4-3 Comparison of speed with other state-of-the-art near real-time method.

Method	FPS	GPU	Input resolution
Segnet [23]	16.6	TitanX	512x256
Enet [24]	76.9	TitanX	1024x512
Espnet [35]	**112.0**	TitanX	1024x512
Erfnet[17]	34.3	TitanX	1024x512
Contextnet[36]	18.3	TitanX	2048x1024
Ours	31.4	TitanX	1024x512

Edgenet has a frame rate of 31.4fps. Our method is slightly slower than ERFnet because we added residule SE-module. But it is faster than the Segnet and Contexnet.

4.3.2 The Experimental on Cityscapes Dataset

- **Evaluation methods**

To ensure that each class for training has enough number of pixel, Cityscapes dataset only use 19 categories for evaluation. For each category, the prediction result of pixels can be divided into 4 conditions Table 4-4.

Table 4-4 Definition of true positive, true negative, false positive and false negative.

(Pixel)		Ground truth	
		Positive condition	Negative condition
Prediction	Positive prediction	True Positive(TP)	False Positive(FP)
	Negative prediction	False Negative(FN)	True Negative(TN)

The result of the semantic segmentation is evaluated by the intersection-over-union(IOU). Each class will calculate its own IOU_c, as shown in

doi:10.6342/NTU201803607

equation(4-1). After averaging the *IOU* of each class, the *mIOU* can be obtained as the main evaluation standard. The calculation method is shown in equation(4-2).

$$IOU_c = \frac{TP_c}{(TP_c + FP_c + FN_c)} \tag{4-1}$$

$$mIOU = average(IOU_c)\ \forall\ c \text{ in } Class \tag{4-2}$$

Where TP_c, FP_c and FN_c indicate the number of true positive, false positive, false negative pixels of class *c*.

Since Cityscapes also provides the label in the category level, in addition to the class *mIOU*, the *mIOU* of the category level is also calculated as the criterion. While calculating the *IOU*, the value will easily be dominated by the large object in the class. This problem will be in street scenes with their strong scale variation, this can be problematic. In order to balance the importance of each instance, Cityscapes also use Instance-Level Intersection-over-union *iIOU* to average the contribution of each instance. The calculation formula is as equation(4-3)

$$iIOU_c = \frac{iTP_c}{(iTP_c + FP_c + iFN_c)} \tag{4-3}$$

Where iTP_c, FP_c, iFN_c indicate the number of true positive, false positive, false negative pixels of class c. However, in contrast to the standard IOU_c measurement, iTP_c and iFN_c are computed by giving different pixel different weight. The weight is based on the ratio of the class' average instance size to the respective ground truth instance size. Since the false positive pixels are not associated with any instance and thus do not require normalization.

● Experimental Result of accuracy

Our model is trained using the Adam optimization [28] of stochastic gradient descent. Training is performed with a batch size of 3, momentum of 0.9, weight decay

doi:10.6342/NTU201803607

of $1e^{-4}$, and we start with a learning rate of $5e^{-4}$. Learning rate of each epoch is formulate as following equation:

$$lr_{epoch} = lr_{start} \times \left[1 - \frac{(epoch_{now} - 1)}{epoch_{max}}\right]^{0.9} \tag{4-4}$$

We train the model with 150 epochs in all. Resolution of training images is 1024x512, which is the same size we used to evaluate inference speed.

We evaluate our model on Cityscapes testing set, which contains 1,525 testing images. Because Cityscapes dataset only provides the ground truth for training data and validation data, we have to submitted our result to their evaluation server. Cityscapes evaluation server provide several metrics for validating the performance of each method, which are shown in previous section.

The experimental results are shown in Table 4-5. These data are uploaded by their authors themselves. After submitting to the server, the score will be calculated and be displayed on the Cityscapes leaderboard.

Table 4-5 Evaluation result on Cityscpaes dataset

Method	*mIOU* class	*miIOU* class	*mIOU* category	*miIOU* category
Segnet basic [23]	57.0	32.0	79.1	61.9
Segnet extended [23]	56.1	34.2	79.8	66.4
Enet [24]	58.3	34.4	80.4	64.0
Espnet [35]	60.3	31.8	82.2	63.1
Erfnet[17]	69.7	44.1	87.3	72.7
Contextnet[36]	66.1	36.8	82.8	64.3
Ours	**71.0**	**46.6**	**88.5**	**75.0**

doi:10.6342/NTU201803607

Table 4-5 show that our method can achieves a better overall *mIOU* (**class**) than Segnet[23], Enet[24] , Espnet[35], Erfnet[17]. And our method also outperforms Contextnet[36] which use 2048x1024 input image to focuses on fine-grained segmentation details to improve the accuracy. The experimental results show that our proposed Edgenet can actually improve the accuracy on both class level and category level.

We list the segmentation results for each class in Table 4-6. Our method can perform better than other method in almost every class.

Table 4-6(a) *IOU* of each class on Cityscapes testing set

Method	road	Sidewalk	building	wall	fence	pole	Traffic light	Traffic sign	vegetation
Segnet basic	96.4	73.2	84.0	28.5	29.0	35.7	39.8	45.2	87.0
Segnet extended	95.6	70.1	82.8	29.9	31.9	38.0	43.1	44.6	87.3
Enet	96.3	74.2	85.0	32.2	33.2	43.5	34.1	44.0	88.6
Espnet	75.7	73.3	86.6	32.8	36.4	47.1	46.9	55.4	89.8
Erfnet	97.9	82.1	90.7	45.2	50.4	59.0	62.6	68.4	91.9
Contextnet	97.6	79.2	88.8	43.8	42.9	37.9	52.0	58.9	90.0
Ours	**98.1**	**83.1**	**91.6**	**45.4**	**50.6**	**62.6**	**67.2**	**71.4**	**92.4**

Table 4-6(b) *IOU* of each class on Cityscapes testing set

Method	terrain	Sky	person	rider	car	truck	bus	train	motorcycle	bicycle
Segnet basic	63.8	91.8	62.8	42.8	89.3	38.1	43.1	44.1	35.8	51.9
Segnet extended	62.3	91.7	67.3	50.7	87.9	21.7	29.0	34.7	40.5	56.6
Enet	61.4	90.6	65.5	38.4	90.6	36.9	50.5	48.1	38.8	55.4
Espnet	66.0	92.5	68.5	45.8	89.9	40.0	47.7	40.7	36.4	54.9
Erfnet	69.4	94.2	78.5	59.8	93.4	52.3	60.8	53.7	49.9	64.2
Contextnet	66.9	92.0	72.2	53.9	91.7	**54.0**	**66.5**	**58.4**	48.9	61.1
Ours	**69.7**	**94.9**	**80.4**	**61.1**	**94.3**	50.0	60.9	52.5	**55.3**	**67.7**

Table 4-7 shows the *iIOU* result, Since the *iIOU* is used for balancing the contribution of different instances in the class, *iIOU* will be calculated only if there is an

doi:10.6342/NTU201803607

independent instance in class. Compare to Contextnet and ERFnet, the *IOU* of our method is slightly lower in truck, bus, and train class. But we outperform them in *iIOU* criteria, which means that our network performs better on smaller instances.

Table 4-7 *iIOU* of each class on Cityscapes testing set

Method	person	rider	car	truck	bus	train	motorcycle	bicycle
Segnet basic	44.3	22.7	78.4	16.1	24.3	20.7	15.8	33.6
Segnet extended	49.9	27.1	81.1	15.3	23.7	18.5	19.6	38.4
Enet	47.6	20.8	80.0	17.5	26.8	21.8	20.9	39.4
Espnet	45.8	19.2	81.7	15.2	24.3	16.8	16.2	35.5
Erfnet	60.1	34.7	86.1	22.6	37.6	31.2	29.0	51.4
Contextnet	47.1	24.7	82.9	19.3	30.6	28.3	21.5	39.9
Ours	**62.5**	**38.1**	**88.4**	**25.4**	**38.3**	**33.0**	**34.4**	**52.5**

We also compared the *IOU* in category level with other method. From Table 4-8, we can see that, our method outperforms the others in all categories, especially in object categories which is very important for driving situation. Our method archives the highest *IOU* and exceed the second (ERFnet) by 3.4%.

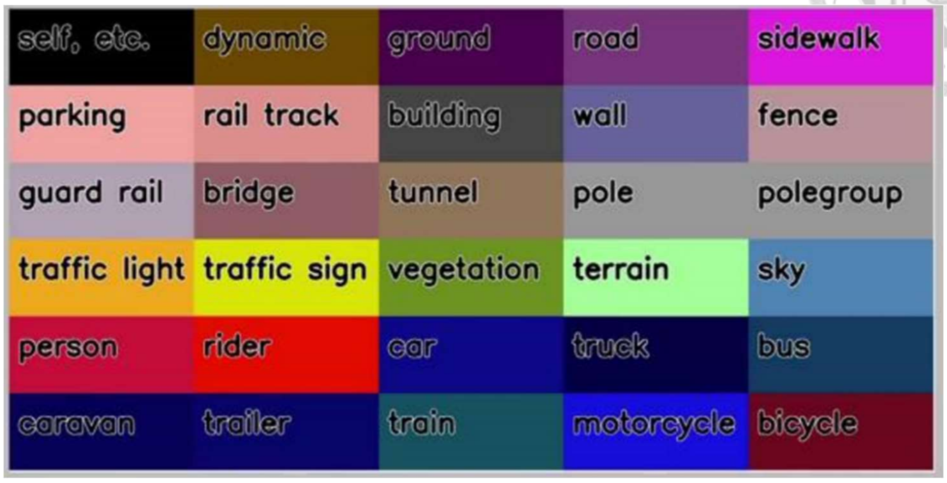
Table 4-8 *IOU* of each category on Cityscapes testing set

Method	flat	nature	object	sky	construction	human	vehicle
Segnet basic	97.4	86.7	42.5	91.8	83.8	64.7	87.2
Segnet extended	97.5	87.1	43.7	91.7	82.8	68.6	87.5
Enet	97.3	88.3	46.8	90.6	85.4	65.5	88.9
Espnet	95.5	89.5	52.9	92.5	86.7	69.8	88.4
Erfnet	98.2	91.5	65.1	94.2	90.6	78.9	92.3
Contextnet	97.8	89.6	47.7	92.0	88.9	72.6	90.8
Ours	**98.4**	**92.1**	**68.5**	**94.9**	**91.5**	**80.9**	**93.1**

The class corresponding to each color is displayed in Fig. 4-3. Some semantic

doi:10.6342/NTU201803607

segmentation results on Cityscapes dataset are shown in Fig. 4-4. These results show that our method can correctly segment every object on road scene.



self, etc.	dynamic	ground	road	sidewalk
parking	rail track	building	wall	fence
guard rail	bridge	tunnel	pole	polegroup
traffic light	traffic sign	vegetation	terrain	sky
person	rider	car	truck	bus
caravan	trailer	train	motorcycle	bicycle

Fig. 4-3 Color table of each class

doi:10.6342/NTU201803607

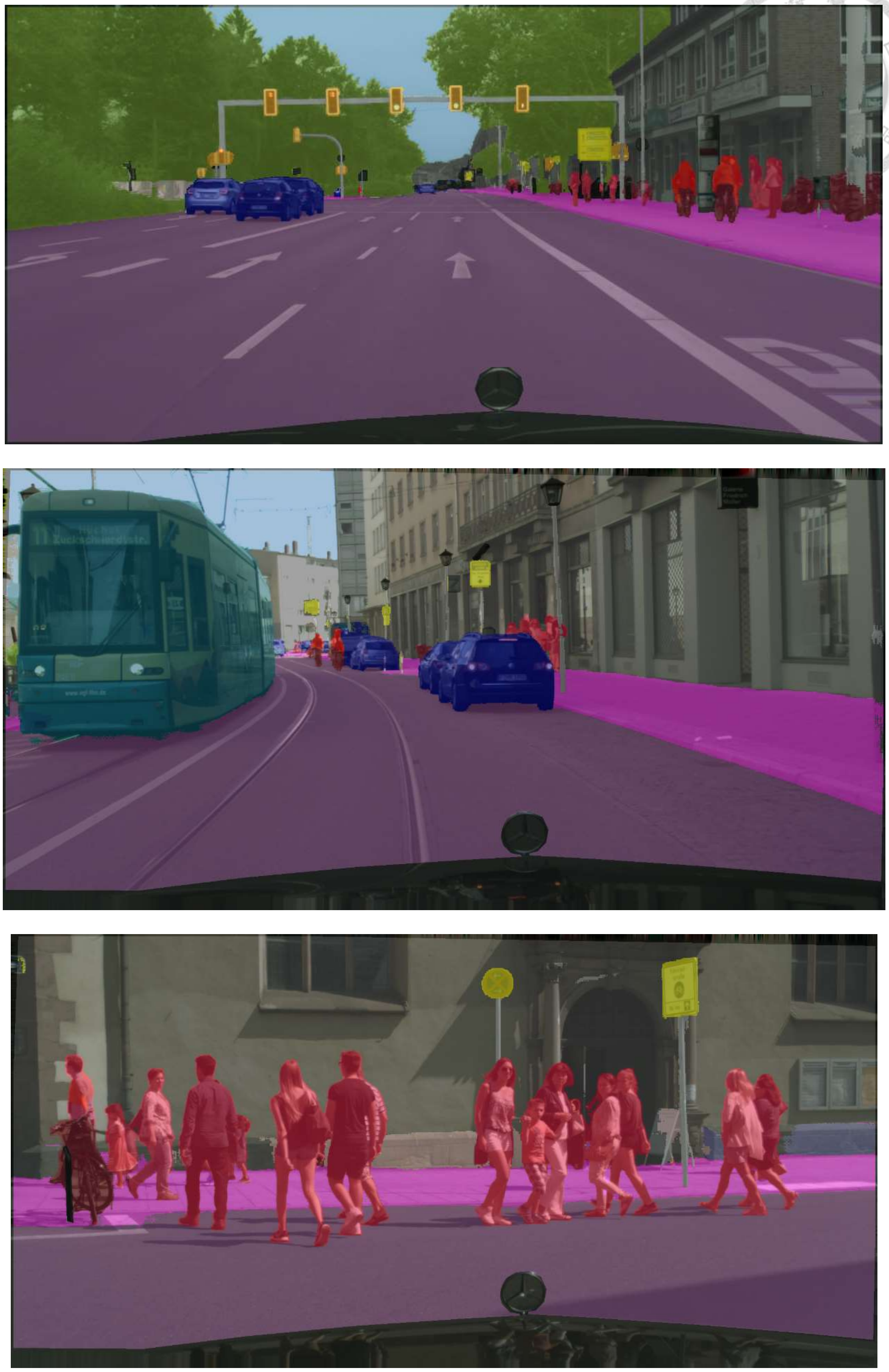


Fig. 4-4 Semantic segmentation results of Cityscapes dataset.

doi:10.6342/NTU201803607

To prove that Edgenet can really improve the results of the segmentation near the edge, we analyzed the results of Edgenet and ERFnet. Fig. 4-5 shows the difference between the number of pixels that are misclassified by two methods. (The number of pixel misclassified by ERFnet minus the number of pixel misclassified by Edgenet) We can see that our proposed method, the closer to the edge, the more misclassified pixels are improved.

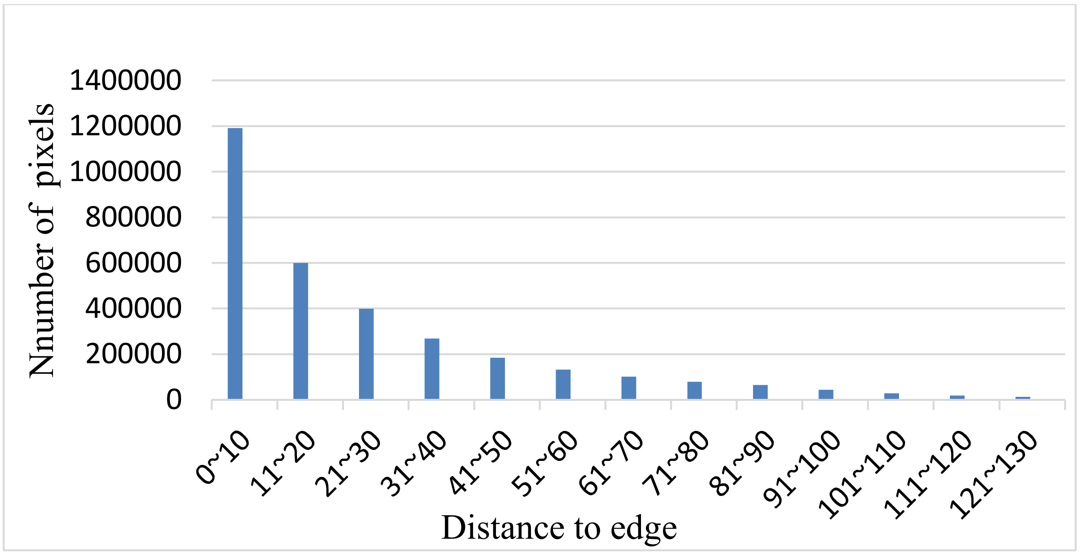


Fig. 4-5 Histogram of improved pixels.

4.3.3 The Experimental on KITTI Road Detection

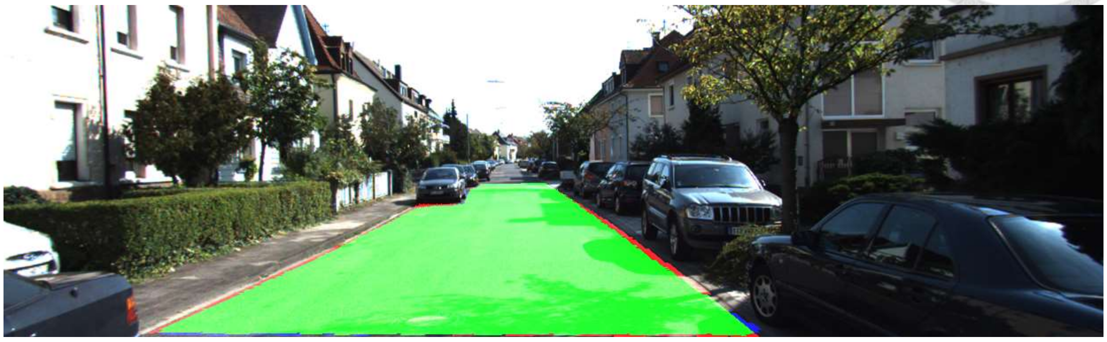
To verify whether our system can be used in road detection module of ADAS, we use KITTI Road Detection dataset for verification.

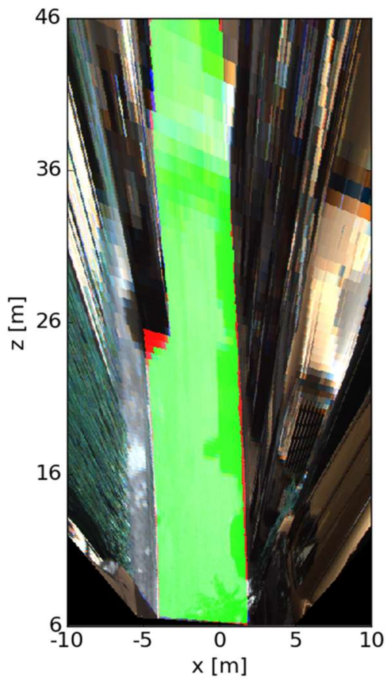
● Evaluation methods

KITTI dataset focus on on-road scene. Therefore, the method for evaluation is different from other semantic segmentation dataset. In perspective image (Fig. 4-6(a)), the road area which is closer to us will have relatively larger scale. This may influence the accuracy because the prediction result will be dominated by the road area near

doi:10.6342/NTU201803607

camera. To solve this problem, KITTI dataset will transform the prediction result into Bird’s Eye View(BEV), in which the road area will not be distorted, as shown in Fig. 4-6(b).


(a) Detection result on regular image



(b) Convert to BEV space

Fig. 4-6 Detection result in different space

We use F_1-score to verify the accuracy of prediction on KITTI dataset. The formula of F_1-score is shown in equation (4-5). The advantage of F_1-score is that it takes care of both recall and precision.

doi:10.6342/NTU201803607

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{4-5}$$

The formula of measuring precision and recall are defined as follows:

$$precision = \frac{TP}{(TP\ \ +\ \ FP)}$$

$$recall = \frac{TP}{(TP\ \ +\ \ FN)} \tag{4-6}$$

Besides F_1-score, KITTI dataset also uses False Positive Rate (FPR) and False Negative Rate(FNR) to evaluate the ratio of false prediction. The formulas are shown in the following equation:

$$FPR = \frac{FP}{(FP +\ \ TN)}$$

$$FNR = \frac{FN}{(TP\ \ +\ \ FN)} \tag{4-7}$$

● Experimental result on accuracy

As we have mentioned before, the size of images in KITTI dataset is about 1242x375 pixels. To keep the aspect ratio, we resize the input image to 1248x376 for the purpose of training. Because the input size is slightly smaller than 1024x512, the execution rate of Edgenet can achieve 35.50 FPS.

Our model is trained using the Adam optimization [28]. Training is performed with a batch size of 4, momentum of 0.9, weight decay of $1e^{-4}$, and we start with a learning rate of $5e^{-4}$.

Just like the requirement for working on Cityscapes dataset, we also need to submit the detection results to KITTI evaluation server. In Table 4-9, our experimental results

doi:10.6342/NTU201803607

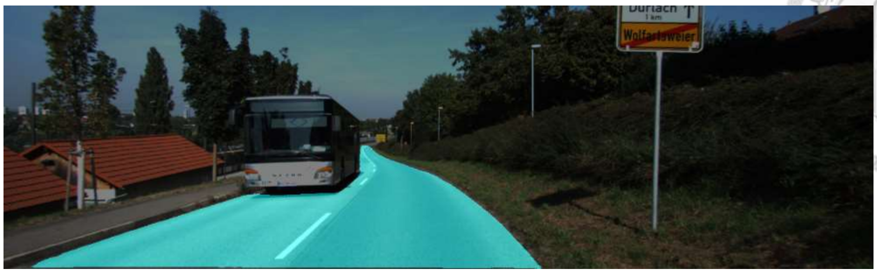
on KITTI road detection benchmark are shown below.

Table 4-9 The result on KITTI car detection.

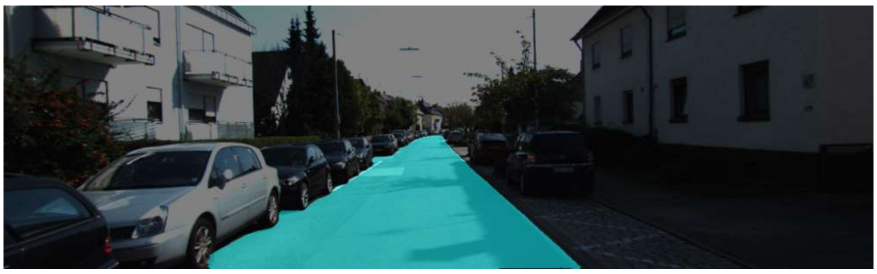
Method	F_1	Precision	Recall	FPR	FNR	FPS
iDST-VT	97.19%	97.09%	97.29%	1.61%	2.71%	1
DFFA[37]	96.35%	96.02%	96.69%	2.21%	3.31%	2.5
SSLGAN[38]	95.53%	95.84%	95.24%	2.28%	4.76%	1.4
RBNet[39]	94.97%	94.94%	95.01%	2.79%	4.99%	5.6
StixelNet II[40]	94.88%	92.97%	96.87%	4.04%	3.13%	0.83
Ours	95.52%	95.52%	95.52%	2.47%	4.48%	35.50

We compare our proposed Edgenet to several state-of-the-art road detection methods, such as iDST-VT proposed by Alibaba's artificial intelligence team, DFFA [37], SSLGAN [38], RBNet [39], and StixelNet II [40]. In particular, RBNet also combines the road edge detection module to aid road detection performance. However, since their edge detection module will participate in the inference stage of road detection, the overall method takes more time for computing. In contrast with RBNet, our Edgenet outperforms it in both F_1-score and speed. Moreover, our method can achieve comparable accuracy to those state-of-the-art methods with $5 \times \sim 40 \times$ speed. In Fig. 4-7, we show the road detection results of our network.

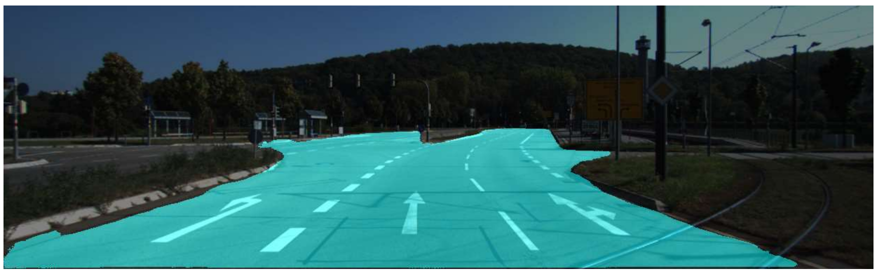
doi:10.6342/NTU201803607



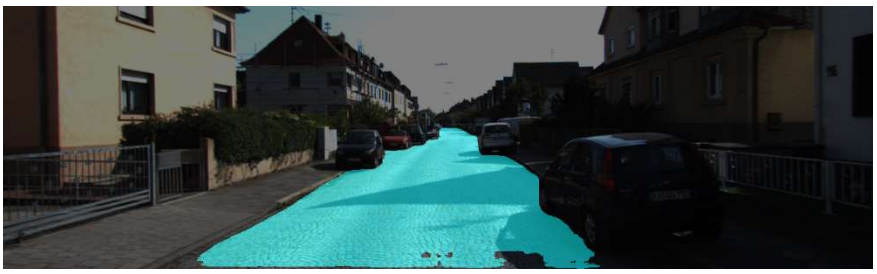
(a) Detection results on urban marked scene



(b) Detection results on urban unmarked scene



(c) Detection results on urban multiple marked lane scene



(d) Detection results on stone road

Fig. 4-7 Road detection results of Edgenet on KITTI testing set

doi:10.6342/NTU201803607

4.4 Autonomous Driving Application

In this section, we combined Edgenet with the Car Steering Angle Prediction System proposed by [41] to verify that Edgenet is suitable for use in autonomous driving application.

4.4.1 Car Steering Angle Prediction System

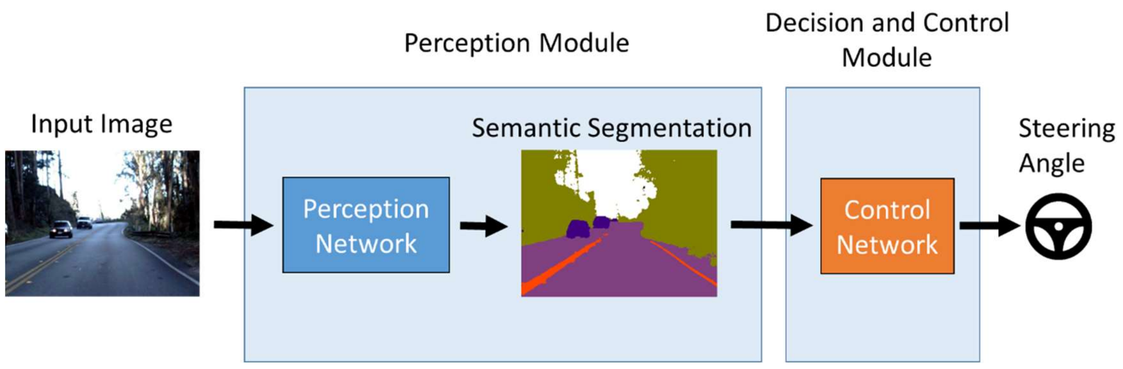


Fig. 4-8 System Overview

The Car Steering Angle Prediction system is shown in Fig. 4-8. The system can be divided into two parts. The first part is the perception network used to obtain the semantic segmentation result of the input image, and the second part is the control network, which is responsible for predicting the steering angle based on the result of the semantic segmentation.

The design of the control network is shown in Fig. 4-9. The Control Network is composed of four convolutional layer with sixteen filters followed by max-pooling layer with stride 2. Two fully connected layers with 256 and 1 neurons, respectively, are attached to the end of the Control Network. The output of the last neuron is the steering angle.

doi:10.6342/NTU201803607

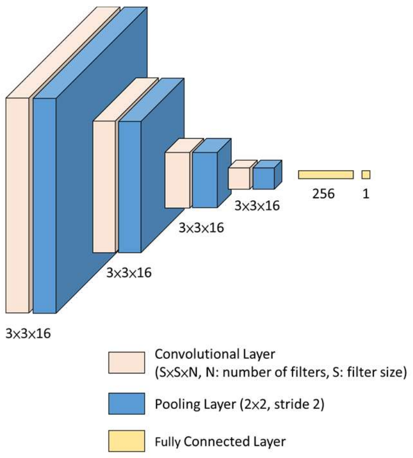

3x3x16
3x3x16
3x3x16
3x3x16
256
1
Convolutional Layer (SxSxN, N: number of filters, S: filter size)
Pooling Layer (2x2, stride 2)
Fully Connected Layer

Fig. 4-9 Architecture of the Control Network.

We implement two methods for perception network. The first is our Edgenet, and the second is ERFnet. We compare the results of these two systems to verify that improving the segmentation accuracy is helpful for autonomous driving applications.

4.4.2 Udacity Self-Driving Car Challenge 2 Dataset

The two Udacity Self-Driving Car Challenge Datasets contain 33,808 training images and 5,614 testing images, respectively. The resolution of images in this dataset is about 640 x 480 pixels. Some example images are showed in Fig. 4-10. The dataset contains several driving scenes in different lighting, road, and traffic conditions. All these images are captured from a front-facing camera installed on a car. The dataset also provides metadata such as speeds and steering angles. We use steering angles as the ground truth label. The steering angles are in the ranges from radius -2.0 to 2.0. For training perception network, we use semantic segmentation annotations provided by [41] which contain 335 labeled image.

doi:10.6342/NTU201803607

Fig. 4-10 Example images of Udacity Self-Driving Car Dataset.

4.4.3 Evaluation Metrics

The evaluation metrics used in the experiments are root mean squared error (RMSE).

RMSE is the common measurements for evaluating the accuracy of a regression model. The definition of RMSE is defined in equation (4-8):

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2} \tag{4-8}$$

doi:10.6342/NTU201803607

where $\hat{y}_i$ is the prediction of i-th sample, y_i is the ground truth label of the i-th sample, and N is the total number of samples.

4.4.4 Overall performance

The overall performances of the system with Edgenet is promising. The result shows that the system with Edgenet has an RMSE of 8.82×10^{-2} on the test set whereas the one with ERFnet has an RMSE of 9.0×10^{-2}. This means that our proposed Edgenet can extract more useful information and help the control network predict the angle more accurately.

Table 4-10 Result on the test set of Udacity dataset.

Model	**RMSE**	**Training Time (Epochs)**
With ERFnet	0.090081	51
With Edgenet	0.088227	47

doi:10.6342/NTU201803607

Chapter 5

Conclusions

In this thesis, a novel CNN based semantic Segmentation network, called Edgenet, is proposed. The proposed method is based on encoder-decoder framework, which allows the network to achieve real-time speed. By adding class-aware edge loss module and residual SE-block, Edgenet can improve the classification result of those pixels near the edges.

We used Cityscapes dataset to evaluate the semantic segmentation performance of our network. Our method outperformed other real-time methods, and is the only real-time method on the Cityscapes dataset which achieves over 71% *mIOU*. We analyzed the distribution of misclassified pixels, and the results show that our network does improve the classification result of those pixels near the edges. We also used KITTI road detection dataset to evaluate the performance of Edgenet as a drivable area detection module. The performance of our method on KITTI road dataset can reach 95% recall/precision and above. Moreover, our method can achieve comparable accuracy to state-of-the-art methods with $5\times \sim 40\times$ speed. For self-driving applications, we implemented Edgenet into the car steering angle prediction system. The result shows that, RMSE of the system with Edgenet is 8.82 x 10^{-2}, which is more

doi:10.6342/NTU201803607

suitable as a reception network for autonomous driving applications than ERFnet. To conclude, the proposed method is suitable to be adopted to other semantic segmentation network to improve their performance.

doi:10.6342/NTU201803607

REFERENCE

[1] P. C. Ng and S. Henikoff, "SIFT: Predicting amino acid changes that affect protein function," *Nucleic acids research,* vol. 31, pp. 3812-3814, 2003.

[2] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 886-893, 2005.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248-255, 2009.

[4] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision,* vol. 88, pp. 303-338, 2010.

[5] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213-3223, 2016.

[6] C. Chen, "Extracting Cognition out of Images for the Purpose of Autonomous Driving," Princeton University, 2016.

[7] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "Icnet for real-time semantic segmentation on high-resolution images," *arXiv preprint arXiv:1704.08545,* 2017.

[8] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, and M. Bernstein, "Imagenet large scale visual recognition

doi:10.6342/NTU201803607

challenge," *International Journal of Computer Vision,* vol. 115, pp. 211-252, 2015.

[9] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3354-3361, 2012.

[10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91-99, 2015.

[11] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, pp. 818-833, 2014.

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672-2680, 2014.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097-1105, 2012.

[14] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580-587, 2014.

[15] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431-3440, 2015.

[16] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution,

doi:10.6342/NTU201803607

and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence,* vol. 40, pp. 834-848, 2018.

[17] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo, "ERFNet: Efficient Residual Factorized ConvNet for Real-Time Semantic Segmentation," *IEEE Transactions on Intelligent Transportation Systems,* vol. 19, pp. 263-272, 2018.

[18] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556,* 2014.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770-778, 2016.

[20] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *arXiv preprint arXiv:1612.08242,* 2017.

[21] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *in Proceedings of the European Conference on Computer Vision*, pp. 21-37, 2016.

[22] R. Chandra and P. Bahl, "MultiNet: Connecting to multiple IEEE 802.11 networks using a single wireless card," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, pp. 882-893, 2004.

[23] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence,* vol. 39, pp. 2481-2495, 2017.

[24] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," *arXiv preprint arXiv:1606.02147,* 2016.

doi:10.6342/NTU201803607

[25] P. O. Pinheiro, R. Collobert, and P. Dollár, "Learning to segment object candidates," in *Advances in Neural Information Processing Systems*, pp. 1990-1998, 2015.

[26] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics,* pp. 400-407, 1951.

[27] F. J. Huang, Y.-L. Boureau, and Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-8, 2007.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980,* 2014.

[29] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," *arXiv preprint arXiv:1709.01507,* 2017.

[30] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *Advances in neural information processing systems*, pp. 3320-3328, 2014.

[31] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* vol. 86, pp. 2278-2324, 1998.

[32] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818-2826, 2016.

[33] J. Alvarez and L. Petersson, "Decomposeme: Simplifying convnets for end-to-end learning," *arXiv preprint arXiv:1606.05426,* 2016.

[34] L.-C. Chen, J. T. Barron, G. Papandreou, K. Murphy, and A. L. Yuille, "Semantic

doi:10.6342/NTU201803607

image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4545-4554, 2016.

[35] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, and H. Hajishirzi, "ESPNet: Efficient Spatial Pyramid of Dilated Convolutions for Semantic Segmentation," *arXiv preprint arXiv:1803.06815,* 2018.

[36] R. P. Poudel, U. Bonde, S. Liwicki, and C. Zach, "ContextNet: Exploring Context and Detail for Semantic Segmentation in Real-time," *arXiv preprint arXiv:1805.04554,* 2018.

[37] X. Liu, Z. Deng, and G. Yang, "Drivable Road Detection Based on Dilated FPN with Feature Aggregation," in *Proceedings of the IEEE Conference on Tools with Artificial Intelligence*, pp. 1128-1134, 2017.

[38] M. Karaduman, "Detection of Road Shape Based Determination of the Number of Traffic Signs and Road Lines."

[39] Z. Chen and Z. Chen, "RBNet: A Deep Neural Network for Unified Road and Road Boundary Detection," in *Proceedings of the International Conference on Neural Information Processing*, pp. 677-687, 2017.

[40] N. Garnett, S. Silberstein, S. Oron, E. Fetaya, U. Verner, A. Ayash, V. Goldner, R. Cohen, K. Horn, and D. Levi, "Real-time category-based and general obstacle detection for autonomous driving," in *Proc. IEEE Int. Conf. Comput. Vis. Workshop*, pp. 198-205, 2017.

[41] k.-H. Tu, "A Deep Learning Based Semantic Segmentation Approach for Car Steering on Urban Roads," Master, department of computer science and information engineering, National Taiwan University, 2017.

doi:10.6342/NTU201803607