Experimental Study on Scene Recognition and Multiple Road Lane Marks Detection Based on Machine Learning Methods

Zhou Xiaosong

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University

in fulfillment of the requirements for the degree of

Master of Engineering

2018

Statement of Originality

I hereby certify that the content of this thesis is the result of work done by me and has not been submitted for a higher degree to any other University or Institution.

. . . .

Date Zhou Xiaosong

Acknowledgment

I would like to express my sincere appreciation to my supervisor, Prof. Huang Guangbin, for his invaluable advices and continuous support. His guidance makes our project team well cooperated and his encouragement motivates me to make contributions. I would also like to express my appreciation to the project instructor Dr. Soh Ling Min, for his warm support on the project proceeding as well as his wise advices that help the project in making bigger progress.

I also thank the colleagues of our team, Mr. Wang Siqi, Dr, Zhou Xiangzeng, Mr. Zeng Yijie, Mr. Zeng Guodong, Mr. Chen Jichao, Ms. Jia Xiaofan and Mr. Xu Rui. They have given me great support with their professional knowledge and skills. During my research, they have shared their experience with me, which helps me a lot on making progress. The discussions about theories and practical techniques have made my research progress and the team project proceed efficiently. I thank them for their inspiring ideas and warm supports.

Furthermore, I would like to acknowledge the support from my friends, Mr. Wu Songwei, Mr. Yang Chule, Ms. Zou Ying, Mr, Jin Ruibing, Mr. Guan Mingyang, Mr. Jiang Wentao, Mr. Wang Yuanzhe, Dr. Guo Fanghong and Mr. Yue Yufeng. Their care and encouragement help me overcome various challenges in both my research and my life.

My gratitude also goes to Prof. Soh Yeng Chai, Assoc. Prof. Wang Han, and Assoc. Prof. Song Qing for their precious suggestions on my research as my Confirmation Exercise members. Moreover, I thank Prof. Wang Dan Wei, Assoc. Prof. Jiang Xudong and Assoc. Prof. Wang Jianliang for the inspiring discussions when I was enrolled in their courses. Last but not least, I would like to extend my appreciation to my family in China.

Table of Contents

Summary

This thesis is written based on two main topics: Scene semantic recognition and road lane marks recognition. The thesis first reviews the studies of general scene understanding or recognition method using semantic segmentation approaches, and then focuses on a more specific topic of analyzing road scene images for detecting road lane marks.

Semantic segmentation has been a popular topic in the field of computer vision research. The main purpose of semantic segmentation is to label pixels of interest in an image with corresponding categories of the objects. This thesis mainly focuses on scene recognition, a branch of semantic segmentation which takes more contextual information into consideration. This thesis presents an experimental study in which a multi-task method for scene recognition is proposed. In this method, edge information is used in enhancing recognition performance. A network which outputs both edge detection map and pixel-wise segmentation is designed. The network is based on FCN and the prediction branches of the two outputs are parallel. Each branch uses multi-scale features concatenation as the image representations. The method expects that the information from edge detection could contribute to the ability of extracting image features for pixel-wise segmentation.

Modern approaches on multiple road lane marks detection are facing several problems. First, insufficient database make related solutions with machine learning technique difficult to train a robust model for application; second, current researches focus on single lane marks detection, which pays less attention to entire roads' condition. To solve the problems, a database with proper ground truth of marks' label set is constructed and a method is developed for detecting and classifying road lane marks of entire roads

with Extreme Learning Machines (ELM). The implementation result shows promising performance and further improvement could be expected.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Computer vision has been developing fast over the years. Its theories and applications have become one of most practical technologies among multiple fields. Such technologies, at present, considerably improve efficiency of humans' works in different fields. Essentially, computer vision represents works on analyzing images. As the ability of the analysis increases, computer vision technology succeeds in assisting humans' work, for example, medical affairs, traffic environment, security, and entertainment.

The development of computer vision is closely associated with related algorithms and the computation ability of computers. A great innovation of computer vision technology is based on the generating and development of machine learning. A major target of computer vision research is image recognition which extracts meaningful information or representation from arbitrary images input. Traditional solutions have limited capability of extracting features from images for the functions used for processing are relatively fixed, making the extraction not robust. Machine learning technology provides multiple approaches to image representation as it has ability of adjusting functions' parameters based on images input, which makes them more flexible for image processing. In the modern time, deep learning, the framework that owns strong ability of function fitting,

possesses large part of image processing solutions. It could generate function models based on large number of data inputs of big variance for certain purpose. Based on such ability, more approaches of complicated image processing problems turn to deep learning for solutions and the results turns to be expecting. Besides, as more high-performance computation devices are developed and applied, the speed of the processing are accelerated and some applications satisfy real-time requirements while they are impossible in traditional ways.

In the field of public traffic, different approaches are involved in depicting traffic situations or solving related recognition problems. Featured solutions contain vehicles detection, traffic signs recognition, road surface detections. Problems of these fields usually have complicated data, in which images' color, structures, textures differ greatly. Machine learning solutions have proved robust on generating models from such data. The development of modern traffic makes auto-driving applications possible for practical implementations. An ideal auto-driving system relies on robust computer vision solutions. For example, to recognize road surface, the system must have pixel-wise recognition ability on traffic images to locate road regions precisely. On the other hand, the system should be aware of the lane where the vehicle is running. Both missions are essential for a regular driving system. In this thesis, the pixel-wise recognition and lane recognition solutions are discussed in detail.

1.2 Motivation

Scene recognition, or scene parsing, is a typical task of semantic segmentation. It labels a whole input image with corresponding categories. It is a pixel level labeling procedure. After an ideal scene parsing, every scene component and every object is delineated and tagged. Scene recognition is a combination of multi-label classification, segmentation, and detection. Different from common process, scene recognition does not only detect objects or regions, it also shapes them, which requires recognition methods to be sensitive on pixel-level information. Therefore, the related approaches consider of factors

that are different from normal object detection tasks.

The main challenges for scene recognition tasks is to produce valid representation of the visual information and the utilization of contextual information in the images. To realize ideal recognition, the representation for each region or pixel directly influences parsing result. If the representation is not properly built, classification errors may occur on some regions of images, or even the entire images. Contextual information extraction has been a popular topic among recognition approaches. As scene recognition requires precise segmentation on every object and region, the interaction between pixels or regions has value on performing more accurate predictions.

In the field of road lane marks detection, computer vision approaches are widely utilized to recognize the features of road lane marks. Besides, graphical model methods are used for video lane marks tracking tasks. Recent researches have paid more attention on discriminating the categories of lane marks. Different from the detection, classification on lane marks requires more knowledge on image patches' features. modern solutions of classification rely more on machine learning approaches, and some solutions have been presented. Among the solutions, lane marks are classified using simple criterion. By discriminating shapes and colors, the marks are categorized in different labels.

The main challenges of the lane marks detection and classification is the feature extraction and marks localization. Since the marks' samples are simple and contain less texture or symbols than other objects, less information can be obtained using general features extracting methods, which makes classification hard to determine the categories of the marks samples. In actual implementations, the scenarios of lane marks detection are changing in camera's viewpoints, illumination of the environments and surrounding conditions of the road surfaces. These factors influence the marks' features extraction difficult for more noises of different kind may interrupt. Therefore, finding the location of the marks are more difficult.

1.3 Objectives

The Main focuses of the thesis based on two independent sections. The first section introduces the research work on scene recognition is introduced. This work contains two parts. One is the project based tasks on refined scene recognition of normal Singapore scenarios. The other is the experiment on optimizing segmentation precision using multi-task network frameworks. The second focuses on optimizing road lane marks' detection and classification performance using machine learning methods. The solution is expected to optimize the classification performance on multiple marks' categories.

1.4 Contributions

The works presented in the thesis contains following parts. In the scene recognition part, a possible method for refining image features is proposed. It utilizes multi-task deep network framework for simultaneous feature extraction of both regions and edges, which in return potentially benefits the capability of the network for analyzing images' textures and structures. Besides, for the corporation project, a new convolution method called dilated convolution is used for high-resolution segmentation. The convolution method enables network process images without pooling convolved layers, which consequently retains image features' resolution and produces detailed segmentation map. In the section of road lane mark detection, a dataset is collected based on Singapore road. the dataset contains road lane marks of 6 categories from different road conditions, covering almost all situations of Singapore environments. Therefore, the dataset is able to represent Singapore road lane marks. The section also provides a method for detecting and classifying road lane marks. In the method, RANdom SAmple Consensus (RANSAC) is used for detecting road lane marks,in which it clusters neighbored lane marks edges and form a representation line for marks. Extreme Learning Machines (ELM) is used for lane marks classification. ELM is a single layer feedforward network. It has faster training speeds while preserving competitive performance of inference. The solution is experimented on the Singapore road lane marks dataset and the performance

is expecting.

1.5 Organization of the thesis

The thesis first introduces my study of scene recognition methods and road lane marks detection. Then an implementation of scene recognition for STE project is presented. An experiment is also performed on multi-task based scene recognition for more precise segmentation. The second part of the thesis is introducing my solution of multiple lane marks detection and classification in different scenarios.

Chapter 2

Literature Review

2.1 Scene recognition using non-deep learning methods

2.1.1 Representation of spatial context information

Scene recognition is a task that labels every pixel with corresponding categories. In the view of computer vision, scene recognition is a typical pixel-level task which execute computations on each pixel. Intuitively, scene recognition groups pixels of certain logics into one corresponding labels, which is similar to superpixel. As a typical non-deep learning solution of image segmentation, superpixel calculates entire image of input and draws a map of regions that groups visual similar pixels. The method is capable of extracting informations from input to depict an approximated structure of input. However, the method lacks ability of classifying grouped regions with proper labels, and does not provide information of connections among the grouped regions. Since the connection represents the context information over the image, certain model is required so that the image can be represented with both regions' internal features and the context.

Based on the requirement, the graphical model and its theory is utilized as a solution. According to formal definition, graphical model is sourced in probability modeling that provide an environment for depicting conditional dependence among random variables.

It contains vertices that represents elements and edges among vertices that represents the connection the elements. A typical graphical model encodes distribution of a set of variables or a graph. The input is not only limited in one dimensional sequences or independent elements but also complicated network of random variables. Some instances of graphical model have shown strong ability of collecting information of sequences and multi-dimensional networks. In the field of scene recognition, the input can be regarded as a two dimensional graph. To efficiently represent the image as required, a model that can handle both internal and context information is introduced which is called Markov Random Fields (MRF). Furthermore, to better build representation of the context, an integration of MRF, Conditional Random Fields (CRFs) is utilized in actual implementations.

Graphical CRFs

The CRFs models have two kinds: Linear-chain CRFs and Graphical CRFs. The Linear-chain CRFs is fundamental for CRFs theory and has been successfully used in natural languages processing. The graphical CRFs can be regraded as an expansion of Linear-chain CRFs. Following the basic properties of graphical models, CRFs consist of observations and labels. Denoting Y, X as random vectors, Y as labels and X as observations. In CRFs models, the observations and corresponding labels' distribution $p(\mathbf{y}|\mathbf{x})$ is formed with:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \exp\{\sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, \mathbf{x_t})\} \tag{2.1}$$

where $\{f_k(y, y, \mathbf{x}_t)\}_{k=1}^{K}$ is a set of feature functions of observations and $\theta = \{\theta_k\} \in \mathfrak{R}^K$ is corresponding parameter. $Z(\mathbf{x})$ is an input-dependent normalization function.

The distribution can be extended into graphical mode. Define a graph G on X and Y. And $\mathbf{x}, \mathbf{y}$ are values of X, Y, respectively. If the distribution The (X, Y) is a conditional random field $p(\mathbf{y}|\mathbf{x})$ factorizes in G, then (X, Y) is a conditional random field.

Here the distribution is defined as:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{a=1}^{A} \Psi_a(\mathbf{y}_a, \mathbf{x}_a) \tag{2.2}$$

where a is the clique index that $a = 1, ..., A$ covering entire graph. $\Psi_a(\mathbf{y}_a, \mathbf{x}_a)$ is the clique feature function on clique a and Z is the normalization function.

The inference through CRFs is to find $\mathbf{y}^\star = \operatorname{argmax}_{\mathbf{y}} p(\mathbf{y}|\mathbf{x})$ for the input $\mathbf{x}$. The input could be sentences to perform natural language processing, and images to perform the image segmentation. Due to the robust ability of obtaining information of neighbor variables, CRFs models are able to collect context information in an arbitrary image. However, directly computing the CRF's distribution is difficult. because the inference requires large storage space and computation resources.

Several methods are used for computing approximated results with CRFs. For example, *Markov Chain Monte Carlo (MCMC)* methods solves the problem of difficulty on approximate a marginal distribution directly. The method generates approximate samples from the joint distribution $p(\mathbf{y})$. *Belief Propagation (BP)* is a direct generalization that imitates ground truth distribution for linear-chain CRFs as well as tree-structured graphs. Both methods are proved valid for CRF model inference. But for general graphical CRFs inference, such methods may become more complex to perform computation.

Fully-connected CRFs

Graphical CRFs provide a structure for efficiently storing internal and context information of an image. However, inference of graphical CRFs is time-consuming due to the normalization function's computation and parameter updates. Therefore, a new approach to solve the computation problem is introduced, and the graphical CRFs model is changed accordingly. Standard CRFs, including graphical CRFs, only connects neighbor elements in the graph. The modified CRF model connects and encodes arbitrary two elements in the graph. Thus this new member of CRF model family is called full-connected CRFs[13].

By setting more connections between elements, fully-connected CRFs stores more information of context, which enables one element's receive enough information form entire image for inference. It seems to enlarge the computation load. Actually, a new method for the inference suits such model better, which saves time for computation.

Inferences that uses fully-connected CRFs compute the energy of the whole graphical model. The model can be an image or a sequence of images. Here only a single image is discussed. In terms of fully-connected CRFs, the model's energy follows *Gibbs Distribution*:

Definition 2.1 *A probability distribution P is* Gibbs Distribution *when it can be written as:*

$$P(\mathbf{x}) = \prod_{A \in \mathcal{C}} V_A(\mathbf{x}) \tag{2.3}$$

where $\mathcal{C}$ is a clique of variables, V_A is a positive function that depends only on $\mathbf{x}$ through the $\{x_t : t \in A\}$.

Based on the definition of Gibbs distribution, HMMs and CRFs can be described by Gibbs concepts.

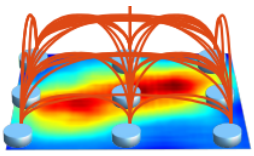

Figure 2.1: A brief illustration of fully-connected CRFs[1].

A fully-connected CRF $(\mathbf{X}, \mathbf{Y})$ is characterized by a Gibbs distribution. In the case of image segmentation, the model is described as fully connected pairwise CRF. In this model, the Gibbs energy is as following:

$$E(\mathbf{y}) = \sum_i \psi_u(y_i) + \sum_{i<j} \psi_p(y_i, y_j) \tag{2.4}$$

In this definition, i and j denotes the pixel indices, which ranges all over the image.

$\psi_u(y_i)$ is the unary potential that measures the energy independently for each pixel by certain classifier to recognize the pixel's label y_i based on the input image's features; $\psi_p(y_i, y_j)$ is denoted as pairwise potential. In this model, the potential has the form:

$$\psi_p(y_i, y_j) = \mu(y_i, y_j) \sum_{m=1}^{K} w^{(m)} k^{(m)}(\mathbf{f}_i, \mathbf{f}_j) \tag{2.5}$$

Each $k^{(m)}$ in the expression is a Gaussian kernel $k^{(m)}(\mathbf{f}_i, \mathbf{f}_j) = \exp(-\frac{1}{2}(\mathbf{f}_i - \mathbf{f}_j)^T \Lambda^{(m)}(\mathbf{f}_i - \mathbf{f}_j))$, the vector $\mathbf{f}_i$ and $\mathbf{f}_j$ are features that represent pixel i and j, $w^{(m)}$ are weights for each kernel, and μ is a label compatibility function. Each kernel $k^{(m)}$ consists of information from color and position which are defined as I_i, I_j and p_i, p_j, respectively;

$$k(\mathbf{f}_i, \mathbf{f}_j) = w^{(1)} \underbrace{\exp\left(-\frac{|p_i - p_j|^2}{2\theta_\alpha^2} - \frac{|I_i - I_j|^2}{2\theta_\beta^2} \right)}_{\text{appearance kernel}} + w^{(2)} \underbrace{\exp\left(-\frac{|p_i - p_j|^2}{2\theta_\gamma^2} \right)}_{\text{smoothness kernel}} \tag{2.6}$$

where parameter m here is 2. In the equation, the appearance kernel depicts the similarity of two neighbored pixels' color. If two neighbored pixels are similar or same in color, then they are more likely to be of one label. The similarity measurement is controlled by the factor θ_α and θ_β. The smoothness kernel suppresses small scattered regions. The parameters are trained with corresponding data.

The function μ computes label compatibility from Potts model, $\mu(x_i, x_j) = [x_i \neq x_j]$. Through this function, similar pixels that are assigned with different labels will increase the value of the function result, which leads to increase in overall energy of the model.

To accelerate the inference speed, a methods called *Mean Field Approximation*[14] helps the inference task. It approximates a distribution $Q(\mathbf{Y})$ based on given features of the input and minimizes the KL-divergence $D(\mathbf{Q}\|\mathbf{P})$ among all distributions candidates Q that can be decomposed into a product of independent factors, $Q(\mathbf{Y}) = \prod_i Q_i(Y_i)$.

A detailed algorithm of the inference is as following:

In the algorithm, each step possesses different proportion of entire execution time, in

Algorithm 1 Mean Field Approximation

Initialize$Q \triangleright Q_i(y_i) \leftarrow \frac{1}{Z_i}\exp\{-\psi_u(y_i)\}$
while not converge **do**
$\tilde{Q}_i^{(m)} \leftarrow \sum_{j\neq i} k^{(m)}(\mathbf{f}_i, \mathbf{f}_j)Q_j(l)\forall m \triangleright$ **Message passing** from all Y_j to all Y_i
$\hat{Q}_i(y_i) \leftarrow \sum_{l\in\mathcal{L}} \mu^m(y_i, l)\sum_m w^{(m)}\tilde{Q}_i^{(m)}(l) \triangleright$ **Compatibility transform**
$Q_i(y_i) \leftarrow \exp\{-\psi_u(y_i) - \hat{Q}_i^{(m)}\} \triangleright$ **Local update**
normalize $Q_i(x_i)$
end while

which the message passing step is most time-consuming. The message passing step collects information for one variable from all other variables, which means a summation is computed. And the summation iterates over all variables. Obviously the computation load is heavy. Here the *High-Dimensional Filtering* method is used to alleviate the computation load.

By re-writing the message passing step, it is 'transformed' into a convolution version with Gaussian kernel $G_{\Lambda^{(m)}}$ in feature space:

$$\tilde{Q}_i^{(m)}(l) = \underbrace{\sum_{j\in V} k^{(m)}(\mathbf{f}_i, \mathbf{f}_j)Q_j(l) - Q_i(l)}_{\text{message passing}} = \underbrace{[G_{\Lambda^{(m)}} \otimes Q(l)]}_{\overline{Q}_i^{(m)}(l)} - Q_i(l) \tag{2.7}$$

Using this expression, $\tilde{Q}_i^{(m)}(l)$ is to be 'convolved' from $\overline{Q}_i^{(m)}(l)$ because the convolution sums over all variables, while message passing does not sum over Q_i.

This step is similar to the low-pass filtering in the field of typical signal processing theorem. Based on this theorem, the message are 'sampled' with a band-limited function where the band width is $\overline{Q}_i^{(m)}$. And to ensure message reconstruction without loss, the band width of the filter in the frequency domain must wider than the band width of the message. Thus ensures the sampling or so-called downsampling and reconstruction (upsampling) can be completed without error.

The estimation for parameter of the model divides in three parts. For parameter $w^{(1)}$, expectation maximization and high-dimensional filtering is used; For parameter $\theta_\alpha, \theta_\beta$, grid search on a holdout validation is efficient, which is also available for estimating

Algorithm 2 High-dimensional filtering on message passing

$Q_{\downarrow}(l) \leftarrow \text{downsample}(Q(l))$
$\forall_{i \in V_{\downarrow}} \overline{Q}^{(m)}_{\downarrow i}(l) \leftarrow \sum_{j \in V_{\downarrow}} k^{(m)}(\mathbf{f}_{\downarrow i}, \mathbf{f}_{\downarrow j}) Q_{\downarrow j}(l)$
$\overline{Q}^{(m)}(l) \leftarrow \text{upsample}(\overline{Q}^{(m)}_{\downarrow}(l))$

$q^{(1)}$; For parameter $w^{(2)}$ and θ_{γ}, in the smooth kernel, they do not affect classification accuracy, significantly. Here they are set by 1, which proved to work well in practice.

Integration

In the modern technique of scene recognition, CRFs and their extensions still contributes to high-accuracy tasks[1]. Combined with deep networks, CRFs acts as a final adjustment for labeling results. It corrects some tiny errors on the labeling maps to make the shapes and boundaries of the map more precise. Besides, its high-performance representation on graphic model still makes contribution to related fields[15].

2.1.2 Flexible modeling for image

Solutions using CRFs model are capable of encoding images with local and context feature. Related researches have made progress on the semantic segmentation tasks. However, such methods have some disadvantages: (1) Functions used for CRFs models are limited in categories; (2) CRFs and their integrations may faces limited topology for modeling relationship between elements; (3)although several estimation algorithm are presented, CRFs' computation speed is not satisfying.

Auto-context method provides a faster and easier routine for image segmentation and related tasks. The method aims at computing posterior distribution directly in a supervised approach. The method also aims at computing a marginal distribution of the posterior. Each image for training is input along with its corresponding labeling map ground truth. A classifier is trained to recognize the label of each pixel. Two kinds of features can be utilized by the classifier for training: (1) image features extracted from the local image patches centered at the current pixel, and (2) context information

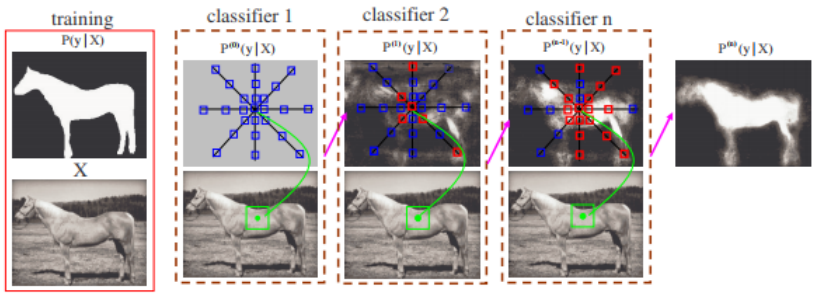

Figure 2.2: Illustration of auto-context procedure where the classification map updates at each iteration. The red patches are selected context patches for the pixel in training[2].

or features which connect to the current image patch. Auto-context initially produces a classification map using certain classifier, which is an initial feature for further iteration. The initial classification map does not construct any context information. The trained classifier will produce an updated classification map for the next classifier. The algorithm perform iterative approach to the ground truth until the error between ground truth and predicted map convergence. In the actual implementation, the method follows the same processing steps as the forward steps of training in which the trained classifier sequence is applied.

Auto-context

Auto-context[2] is presented to better compute the marginals. To start an auto-context procedure, an initial classification result or label map of image patches is produced by a learned classification model. The result can be written as:

$$\mathcal{P}^{(0)} = (\mathbf{p}_1^{(0)}, ..., \mathbf{p}_n^{(0)}) \tag{2.8}$$

where $\mathbf{p_i^{(0)}}$ is the posterior marginal for each pixel i learned by certain classifier.

Because the algorithm requires computation on image patches, some definitions are set in advance. Denote a training set $S = \{(y_{ij}, X_j(V_i)), j = 1, ..., m, i = 1, ..., n\}$. The method receives image patches $X_j(V_i)$ centered at each pixel (i is the pixel index) as input where V_i denotes all the pixels in the patch.

In auto-context, the training is executed in an iterative way. Based on the initial probability map $\mathcal{P}_j^{(0)}(i)$ on image patch $X_j(V_i)$ for each j in the image, a classifier is set and trained using the probability maps and the training set. The training outputs a new probability map which is iterated in a new classifier. Consequently, a classifier sequence is built for actual implementation. For each image patch, a set of context pixels are chosen and trained together with the patch in one iteration. These context pixels can be chosen in neighbor regions of i pixel or the ranged regions. It is determined by the learning algorithm. Once a new classifier is trained, the algorithm iteratively perform the training based on the previous result until the error between ground truth and prediction converges. To conclude, the algorithm is an iteration processing of the following update on marginal distribution:

$$p^{(n)}(y_i|X(V_i), \mathcal{P}^{(n-1)}) \rightarrow p(y_i|X) = \int p(y_i, y_{-i}|X)dy_{-i} \tag{2.9}$$

In the actual implementation, the first classifier does not produce context feature for next training. Because the probability map is not robust enough for clear recognition. However, some applications can utilize such properties. e.g. medical image segmentation, the structure of image elements are relatively fixed in the view of certain organ, which makes the features selection easy to determine. One then can use a probability map as the initial $\mathcal{P}^{(0)}$.

Algorithm 3 Auto-context method

Input: $S = \{(Y_j, X_j), j = 1..m\}$

Produce probability maps $\mathcal{P}_j^{(0)}$ for each input X_j with uniform distribution on all the labels. For $t = 1, ..., T$:

(1) Set up $S_t = \{(y_j, (X_j(V_i), \mathcal{P}_j^{(i)})), j = 1..m, i = 1..n\}$ as training set;

(2) Use image features from $X_j(V_i)$ and context feature maps from $\mathcal{P}_j^{(t-1)}(i)$ to train a classifier;

(3) Compute new classification maps $\mathcal{P}_j^{(i)}$ with trained classifier;

Output: $p^{(n)}(y_i|X(V_i), \mathcal{P}^{(n-1)}(i))$. A sequence of trained classifiers

Understanding auto-context

The auto-context method works mainly on computing marginal distribution, while Belief Propagation (BP) has been a popular solution for the same problems. There are some differences between BP and auto-context that influence their procedure significantly: (1) On the graphical model, BP measures every pixel pairs on all possible labels; Auto-context uses selected filters to evaluates a sequence of learned classifiers. Therefore, auto-context is capable of measuring long range of support features. (2) BP is performed on fixed structures and share same update procedure. Auto-context could choose the specific way of classification. (3) In BP, the processing is separated into several stages, computing pixel-wise energy and pair-wise energy then compute the distribution. Auto-context directly computes the marginal distribution and the actual inference have same steps of processing as the learning phase.

Integration

Auto-context method has great value on scene recognition development. In the manuscript, one opinion is worth consideration that label (in probabilities) contexts greatly improve the segmentation/labeling result. The opinion provides a important idea that context information is important for semantic segmentation. Current research also prove that in some tasks of representing fixed structures, e.g. medical images, auto-context method could still contribute to making good performance for it collect information with a fixed routine on images, which could gather information not only from texture but also structures.

2.2 Background of Scene Recognition and Deep Learning

2.2.1 Introduction

With the development of deep learning, image segmentation and its related tasks find a robust way of extracting images' information and performing further computing. Deep learning methods and their applications benefit from modern computation device such as high-performance CPU and GPU, getting state-of-the-art performance as well as high processing speed. Compared with traditional methods, deep learning methods suits GPU computation framework, which has been utilized in a number of video games' high speed vision rendering. Besides, the methods provides big framework to extract input data's information, or called features, acquiring massive and valuable data for related tasks. Based on the advantage of computation devices, some new methods of convolution operation are introduced to accelerate convolution and optimize data usage [17].Therefore, deep learning and its relative applications dominates the field of image representation and processing.

Currently, a popular and robust network type is introduced and widely utilized: Convolutional Neuron Networks (CNN). It produces a series of feature vectors for regions of multiple sizes centered around every pixel in the image, covering a large context[18]. Generally, the features extracted from CNN mainly represents images' pixel-wise information. For a feature map of an image, each level contains different degree of information As convolution level goes deeper, the feature map on every point could store information with wider regions in the original images, and low-level features' representations on original image are more local.

Obviously, deep network requires proper learning methods to construct a model for further application. Back propagation (BP) is used in most of learning procedure. Because of massive dataset and complexed parameter frameworks, learning procedure could be time-consuming even in high-performance computing devices. A number of modern

learning methods have shown that, once a general model is properly learned for a basic task, e.g. image classification, it could be further used in other tasks using same network structures. Learning procedures that are based on such models could perform fine-tuning with corresponding dataset instead of learning a new model, which could save much time and the performance is still robust.

Another important part of deep learning is data. As mentioned above, deep learning requires massive data to train and test networks. Generally, a dataset of images contains images to train and corresponding classes or labels for loss computing and testing. Datasets are established for certain target, e.g. ImageNet[19] is mainly for image classification, VOC[7] is mainly for image segmentation, NYU[20] dataset provides image segmentation data as well as depth information for related researches; KITTI dataset[21] focuses on traffic related data collection, including traffic objects detection and driving-related scene components recognition. ADE20K[22] is for scene parsing. For each dataset, categories differs for their unique purposes. For major datasets such as VOC 2010[7], 460 categories are contained in 19,840 images. Based on their images and labels' capacity and diversity, different dataset may influence learning result in different degrees.

This section focuses on reviewing fundamental deep learning frameworks of scene recognition and analysis of corresponding datasets.

2.2.2 Recurrent Neural Networks (RNN)

Introduction

The CRFs models and auto-context models provide robust routines for encoding internal and context information over images. However, such models' execution is time-consuming and reported low performance when processing complicated images. The problems may source from the way of representation. The CRFs models build representations directly from pixels. As an image contains numbers of pixels, computing connections among them definitely costs large resources. Besides, computation over

pixels pays less attentions on regional information. Admittedly, pixel-level processing contributes representation over an image. However, such representation does not provide high-level features for classification since processing pixels individually cannot bring more information. Therefore, when processing complicated or general image, its robustness may fall. Similarly, auto-context sets up a fixed structure for extracting context information, which may negatively influence its training on general images as they do not share similar structures.

Based on these reasons, more techniques are used for enhancing the context representation. One efficient solution is that, grouping neighbor pixels into patches, and building dependencies over these patches. Such routine is similar to the combination of CRFs models and auto-context, which utilizes pixels clusters' information and establish context representation over entire image. One model called Recurrent Neural Networks (RNN) [3] is capable of such representation. It encodes such context into spatial dependency, which provides information among image patches over entire image.

For Recurrent Neural Networks (RNN), images can be represented in certain spatial logic. The main target for RNN to extract spatial features is considering the spatial dependencies in images. Recent studies [3] [4] usually separate original images or its feature maps into $n \times n$ square regions and use RNN framework to learn the features in certain orders. Such methods gains information of connections among image patches. A straightforward way is to learn all types of image region combinations, which costs huge resource of computation and the effect is not reliable. To better learn the dependencies, the network should have ability of *memory* that all the processed data can be saved and their correlations can be analyzed.

In this section, a typical RNN framework is introduced and the combination of RNN and CNN is presented.

RNN theory

RNNs are modified neural network that one hidden state receives feedback from the output of the previous hidden states. The feedback procedure performs as a loop in basic RNN frameworks. The frameworks are designed for extracting context features of corresponding data, especially for sequential data flows[3]. Recent researches report good performances on natural language processing using RNN[16]. RNN mainly processes a set of data with sequential order based on certain rule. Denote the length of sequence as S. Set $x^{(s)}$ as the input data of the sequence at state $s \in [1, ..., S]$, $h^{(s)}$ as the hidden layer, and $y^{(s)}$ as the output. A typical RNN forward procedure can be described as:

$$h^{(s)} = f_h(W_{hh}h^{(s-1)} + W_{ih}x^{(s)} + b_h) \tag{2.10}$$

$$y^{(s)} = f_o(W_{ho}h^{(s)} + b_o) \tag{2.11}$$

W_{ih}, W_{hh}, W_{ho} are the transformation matrices or parameters connecting $x^{(s)}$ and $h^{(s)}$, $h^{(s-1)}$ and $h^{(s)}$, $h^{(s)}$ and $y^{(s)}$, respectively. b_h and b_o are the constant bias terms, and f_h and f_o are the non-linear activation functions.

The feedback loop of RNNs enables the framework store all the previous inputs and their extracted information. The storage is progressively increased by updating W_{ih}, W_{hh}. W_{ho}. Ideally, the RNNs can store all the information they have processed. Such information is valuable for establishing useful connections between current data and its context. In the field of computer vision or semantic segmentation, RNN is utilized for extracting information for internal regions' connection inside one image and find the dependencies among regions.

Input data for RNN processing

Generally, images can be directly sent to RNN for further processing. However, due to huge diversity of every image, raw image inputs also carry massive noises which may ruin RNN performance. Thus, the input data should keep their own feature as well as enhancing the compatibility of interacting with other data. Fortunately, As CNN develops, such requirement can be easily satisfied. As talked before, CNN get raw images and produce rich image representations for each image, which contains rich features to describe images and suppresses noises of images. Therefore, the image features are suitable as input data of RNN processing. And experiments proved it efficient. Here, the combined network of CNN and RNN is called C-RNN.

RNN for images

Similar to text and sound, images also have contextual information and carries structured dependencies.

In the view of human vision, images have multiple structures based on their colors, edges, and different objects they contain. Such complexity produces information as well as complications for image processing that they do not have existing routines to store their structures. Using RNNs on them face difficulties of the ways to represent the images' context. To make the representation uniform and easy for further processing, the image is transformed into 1D sequence that contains all the image patches of same size. Using the sequence, RNN learns the spatial dependencies for these patches. Furthermore, the RNN can only utilize previous context. In fact, future context is also helpful, especially in image data, as images do not hold predetermined structures or dependencies. Therefore, a four-direction sequences is generated from one image and an RNN is used to process these sequences to extract context information of all the directions mentioned.

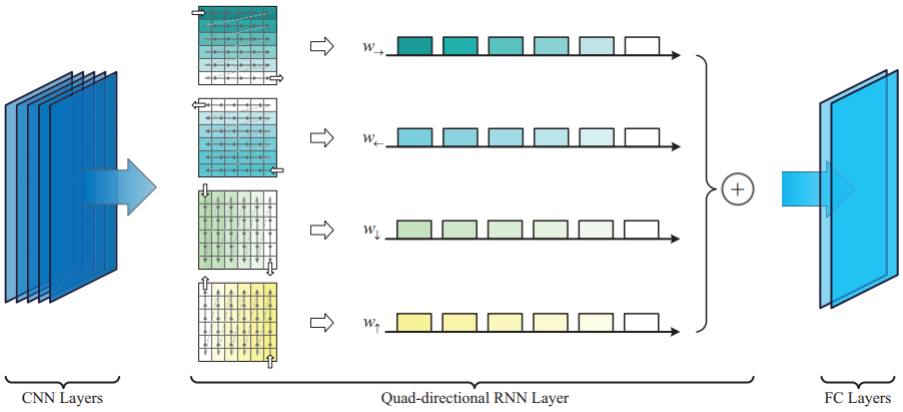

Figure 2.3: An illustration of RNN image sequence construction[3]

In the implementation, RNN searches all the sequences one by one in four orientations: from bottom to top, from top to bottom, from left to right, and from right to left. For each orientation, unified weights W_{hh} and W_{ih} are shared for all the image region features in each orientation. The weights are specified by directions: $W_{hh\rightarrow}$ and $W_{ih\rightarrow}$ for the left-right, $W_{hh\leftarrow}$ and $W_{ih\leftarrow}$ for the right-to-left, $W_{hh\downarrow}$ and $W_{ih\downarrow}$ for the top-bottom and $W_{hh\uparrow}$ and $W_{ih\uparrow}$ for the bottom-top.

For each sequence, the RNN can be formulated as:

$$
\begin{aligned}
h_{\rightarrow}^{(s)} &= f_h(W_{hh\rightarrow}h_{\rightarrow}^{(s-1)} + W_{ih\rightarrow}x^{(s)} + b_{h\rightarrow}) \\
h_{\leftarrow}^{(s)} &= f_h(W_{hh\leftarrow}h_{\leftarrow}^{(s-1)} + W_{ih\leftarrow}x^{(s)} + b_{h\leftarrow}) \\
h_{\downarrow}^{(s)} &= f_h(W_{hh\downarrow}h_{\downarrow}^{(s-1)} + W_{ih\downarrow}x^{(s)} + b_{h\downarrow}) \\
h_{\uparrow}^{(s)} &= f_h(W_{hh\uparrow}h_{\uparrow}^{(s-1)} + W_{ih\uparrow}x^{(s)} + b_{h\uparrow}) \\
h^{(s)} &= h_{\rightarrow}^{(s)} + h_{\leftarrow}^{(s)} + h_{\downarrow}^{(s)} + h_{\uparrow}^{(s)}
\end{aligned}
\tag{2.12}
$$

In the processing of training, the parameter updating is done by back propagation through time (BPTT), which is a variant of BP.

Obtaining information by four sequences and take their summary reflects basic routine of the image representation with RNN. However, the specific method has big space for optimizing. Sequencing image regions direction by direction makes computation

complicated and the information may get loss for each sequence contains context information of one direction. A new kind of sequencing method is needed that also takes image regions as element but uses different sequencing order.

DAG-Recurrent Neural Networks

One of the successful example is the Directed Acyclic Graph (DAG)-RNN framework. The method also takes CNN image representation as input. Different from the RNN method, DAG-RNN[4] uses Undirected Cyclic Graphs (UCG) and decomposes it into multiple DAGs to store spatial dependencies information.

Since UCG is a cycle structure, it is unable to unfold its graph into an acyclic processing condition for the context information in UCG is closed. Therefore, some new graph representation for 2D images that is accessible in RNN should be used to construct the representation of the image. Thus, the UCG is decomposed into many small cliques called DAG.

In the method, an image I is stored in the form of graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_i\}_{i=1:N}$ stores the images patches and $\mathcal{E} = \{e_{ij}\}$ stores the connection between patches (e_{ij} denotes the connection from v_i to v_j). The graph is regarded as input of the whole network and a hidden layer is built with the same form as the graph so that a forward processing on the sequences can be established by scanning $\mathcal{G}$. Following the input layer, The hidden layer of nonlinear function $\mathbf{h}^{(v_i)}$ receives its corresponding local input $\mathbf{x}^{(v_i)}$ and other hidden layers' output as input, which is the typical structure of RNN. The output from other hidden layers are summed as the input to the current hidden layer. And the local input $\mathbf{x}^{(v_i)}$ is formed by feature of the corresponding image patch. Based on the definition above, the basic forward procedure of DAG-RNN can be

expressed as Eq.2.13.

$$\begin{aligned}\hat{\mathbf{h}}^{(v_i)} &= \sum_{v_j \in \mathcal{P}_G(v_i)} \mathbf{h}^{(v_j)} \\ \mathbf{h}^{(v_i)} &= f(U\mathbf{x}^{(v_i)} + W\hat{\mathbf{h}^{(v_i)}} + b) \\ \mathbf{o}^{(v_i)} &= g(V\mathbf{h}^{(v_i)} + c)\end{aligned} \tag{2.13}$$

where $\mathbf{x}^{(v_i)}$, $\mathbf{h}^{(v_i)}$, $\mathbf{o}^{(v_i)}$ denote the input, hidden and output layers located at v_i in the graph, respectively, $\mathcal{P}_{\mathcal{G}}(v_i)$ is the direct predecessor set of vertex v_i in the graph $\mathcal{G}$, $\hat{\mathbf{h}}^{(v_i)}$ is a summation over the information of all predecessors of v_i. As the number of predecessors for each vertex in $\mathcal{G}$ is fixed, a specific W can be learned for each predecessor. Thus a trained dependency of the context model can be acquired.

In the training phase, similar to typical networks, DAG-RNN also requires backward pass to update parameters. In this phase, DAG-RNN computes derivatives on each vertex in the graph. The derivative at vertex v_i is computed as following:

$$\begin{aligned}\Delta V^{(v_i)} &= g'(\mathbf{o}^{(v_i)})(\mathbf{h}^{(v_i)})^T \\ d\mathbf{h}^{(v_i)} &= V^T g'(\mathbf{o}^{(v_i)}) + \sum_{v_k \in \mathcal{S}_{\mathcal{G}}(v_i)} W^T d\mathbf{h}^{(v_k)} \circ f'(\mathbf{h}^{(v_k)}) \\ \Delta W^{(v_i)} &= \sum_{v_k \in \mathcal{S}_{\mathcal{G}}(v_i)} d\mathbf{h}^{(v_k)} \circ f'(\mathbf{h}^{(v_k)})(\mathbf{h}^{(v_i)})^T \\ \Delta U^{(v_i)} &= d\mathbf{h}^{(v_i)} \circ f'(\mathbf{h}^{(v_i)})(\mathbf{x}^{(v_i)})^T\end{aligned} \tag{2.14}$$

where $\circ$ denotes the Hadamard product. $g' = \frac{\partial L}{\partial \mathbf{o}(\cdot)} \frac{\partial \mathbf{o}(\cdot)}{\partial g}$ is the derivative of loss function L with respect to the output function g and $f'(\cdot) = \frac{\partial \mathbf{h}}{\partial f}$. The second term, $d\mathbf{h}^{(v_i)}$, propagates local information over vertices.

The UCG is decomposed into a set of DAGs: $\mathcal{G}^{\mathcal{U}} = \{\mathcal{G}_1, ..., \mathcal{G}_d, ...\}$. Therefore, the UCG-structured images are represented as the combinations of a set of DAG-structured images. Each DAG factor executes computation of DAG-RNN independently as their own hidden layer $\mathbf{h}_d$ is built. Each DAG-RNN produces an output o by aggregating the

independent hidden layers (Eq. 2.15).

$$\begin{aligned}\mathbf{h}_d^{(v_i)} &= f(U_d\mathbf{x}^{(v_i)}) \sum_{v_k \in \mathcal{S}_\mathcal{G}(v_i)} W_d\mathbf{h}_d^{(v_j)} + b_d \\ \mathbf{o}^{(v_i)} &= g(\sum_{\mathcal{G}_\Gamma \in \mathcal{G}_\mathcal{U}} V_d\mathbf{h}_d^{(v_i)} + c)\end{aligned} \tag{2.15}$$

where U_d, W_d, V_d and b_d are weight matrices and bias vector for the DAG $\mathcal{G}_d$, $P_{\mathcal{G}_d}(v_i)$ denotes the direct predecessor set of vertex v_i in $\mathcal{G}_d$.

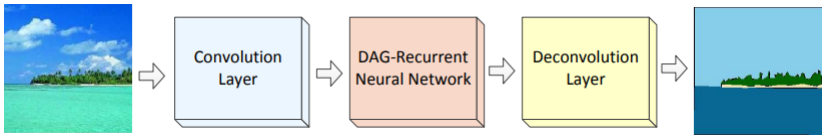

Figure 2.4: The general procedure of DAG-RNN processing which combined with CNN feature extraction and deconvolution label map reconstruction[4]

The whole network combines Convolution network, DAG-RNN network and deconvolutional network. The convolutional network produces compact and rich features for DAG-RNN to process. Then DAG-RNN receives the features and performs inference. Finally, the deconvolutional network is used to upsample the features maps that DAG-RNN produces by learning a set of filters.

Implementation

In the experiment, the SiftFLow dataset, which contains general scene label ground truth, is used for measurement. For the DAG-RNN fully labeling network, VGG-16 with corresponding pre-traiend model is used for image feature map extraction.

The image feature comes from the conv5 layer of VGG-16. In DAG-RNN part, there are two kinds of UCG decomposition. The figure shows that there are 4 and 8 neighborhood systems. As exampled in the figure, the length of propagation from v_9 to v_1 is in $\mathcal{G}_{nw}^8$halved to that in$\mathcal{G}_{nw}^4$.

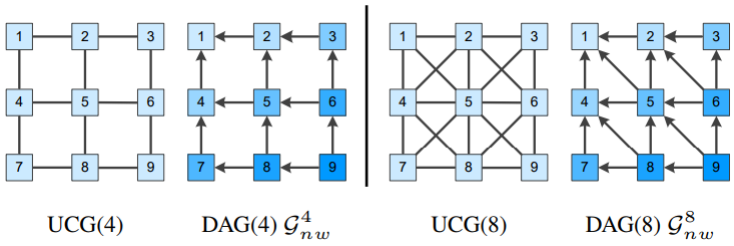

Figure 2.5: Two ways of UCG decomposition[4]

The DAG-RNN is trained using Stochastic Gradient Descent (SGD) method. The parameters are updated iteratively as the training images are inputed one by one.

2.2.3 Fully Convolutional Networks

Introduction

CNNs are driving advances in recognition networks. Developed from classification[23], more researches utilize CNN for local tasks including object detection[24], regional prediction, and local correspondence.

The further step of image inference, is to make predictions at every pixel. Previous work has made accomplishments on related fields with convnets. However, These works may occur some short-comings. The fully convolutional network (FCN) [5]is developed to improve the performance of image segmentation. And as the framework develops, FCN has been the basis of many popular networks for image segmentation and related tasks.

A main contribution of FCN is that it provides an end-to-end, pixel-wise on baseline for semantic segmentation. It provides an efficient solution for high-performance dense prediction of inputs in arbitrary size. Similar as traditional deep network training, the training and test of FCN can be performed with feedforward computation and Back Propagation. Unlike former methods, FCN extend standard deep network for image classification, using pre-trained network model and change the fully connected layers into fully convolutional layers to process images and their pixel-wise representations.

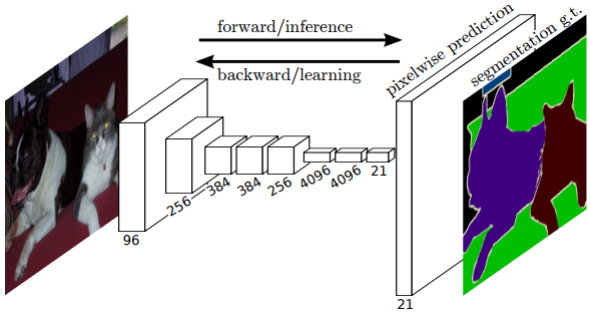

Figure 2.6: A brief procedure illustration of FCN framework.[5]

Multi-scale context features

To get ideal results of scene recognition, obtaining proper image features is crucial for further prediction. By computing with CNN framework, feature extraction has become efficient and robust. In the field of image classification, a well-trained CNN model could provide robust image features (mostly single scale) for accurate classification. Scene recognition requires pixel level information for accurate segmentation and classification, thus the network should provide features with richer information.

Fully Convolutional Network (FCN) has provided multi-scale feature structure for object segmentation. The multi-scale structure collects feature map from different scales with respective to the original image and concatenates them together for further prediction. The FCN network contains two modules including downsampling path and upsampling path. This kind of combination can be traced back to some previous work which concatenates several feature elements as a pixel's or region's representation[25]. The downsampling path, similar to normal deep network, provides features for classifications, and upsampling path contains deconvolution, which upsamples features and output score masks. These two paths could extract high level information, which helps upsampling path predicting the score masks in a pixel-level way.

One key point for multi-scale feature is the receptive field of convolution network. As convolution goes deeper, each point of the feature map has larger receptive field than its former layer, which means that in the high-level feature map, each feature point contains rich context information of the corresponding pixel and its neighbor pixels in

the original image. In the field of image segmentation and scene recognition, executing pixel-level prediction requires information from the pixel and its neighbors. The quality of these information totally depends on the corresponding feature map. Thus, a proper receptive field for each feature point leads to an ideal segmentation result.

In the case of scene recognition, a pixel label's prediction based on not only its own information, but also the neighbor pixels' information because scene components are logically connected. For example, there cannot be a ship running on the road, or a car completely floating in the air. Collecting context information helps suppress such ambiguity. Besides, some scene components require consistency in the labeling results, for example grassland and ocean. Using context information could help reduce the appearance of the inconsistency.

Therefore, based on FCN, the multi-scale feature map with contextual feature representation is established for this scene recognition model. Specifically, the network contains a number of convolutional layers, 5 max-pooling layers for downsampling and 3 deconvolutional layers for upsampling pixel-wise map and edge map respectively.

Network architecture

In a typical FCN framework, each layer outputs a data feature of height h, width w and number of channel d. The first layers receives input image of 3 dimensions, namely $d = 3$ and its size is $h \times w$. In the higher layers, each data features element stores a bunch of information from certain region of input image. The region is called *receptive fields*.

The network is built in translation invariance. The basic components of the network (convolution, pooling, and activation functions) are deployed to each layer and serve for their own input regions. Each component's computation only focuses on relative spatial dependencies, which means the feature of one layer cannot be processed by the components from other layers. Denote $\mathbf{x}_{ij}$ as the data input at location (i, j) in certain layer, and $\mathbf{y}_{ij}$ as the output for the layer which is also the input for the following layer.

$\mathbf{y}_{ij}$ is then computed by:

$$\mathbf{y}_{ij} = f_{ks}(\{\mathbf{x}_{si+\delta_i, sj+\delta_j}, 0 \leq \delta_i, \delta_j \leq k\}) \tag{2.16}$$

where k denotes the kernel size. s denotes scanning steps of computation kernel, and f_s determines the layer type: convolution, pooling (including average pooling, max pooling), or some non-linear activation functions. In addition, some special functions for specific methods can also be concluded.

Based on this procedure, the network computes or establishes a filter, which is formally called Fully Convolutional Network. The result of the FCN is of same size as the input.

As the network changes, which instead produces a map of multiple value, the loss function for the network is also changed for suiting the framework. In FCN, the loss function $l(x;\theta) = \sum_{ij} l'(x_{ij};\theta)$ is computed by outputing gradient that sums over the gradients of each of its spatial components. Therefore, the SGD on l changes simultaneously with the one on l'.

One feature of FCN is that it could provide dense prediction. Dense prediction is a procedure that could output a map of predictions instead of a single dimensional result from traditional deep networks. Such feature makes the FCN becomes a base framework for multiple tasks that requires pixel-wise information. In fact, the fully convolution layers of FCN is a integration or special case of fully connected layers in typical deep network. As the fully connected layers generates a one dimensional output from the convolved result of last layer, it holds a special convolution operation to convolve the feature map into a one dimensional vector. Developed from the fully connected layer, the fully convolution layer makes the convolution operation more general and provides some spatial information.

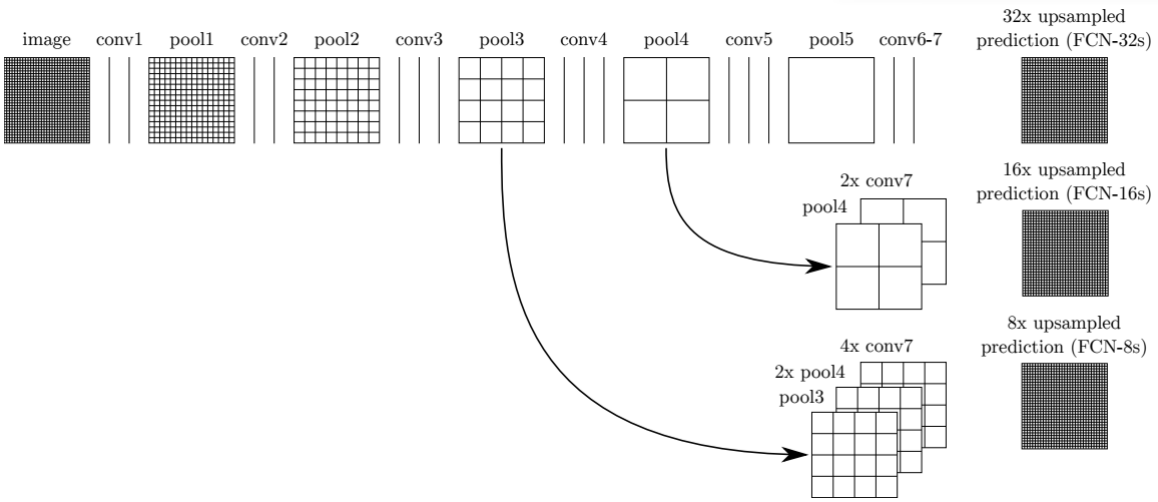

Figure 2.7: Detailed structure of FCN. There are three kinds of FCN frameworks:(1)FCN-32s: Directly upsampling conv7 output at stride 32 to original size; (2)FCN-16s: upsampling pool4 output at stride 16 and conv7 output and sum them up to get the final result; and (3)FCN-8s: upsampling pool3 layer output (at stride 8), pool4 layer output and conv7 layer output and then sum up to get the final prediction[5]

.

Traditional methods tried to use image patches to recognize the label of the patches' center pixel, thus complete the pixel-wise prediction for an image. Therefore, such methods will compute the label map of an image patch by patch. In FCN, the framework directly computes a pixel-wise output which significantly reduces the computation time.

The pixel-wise output makes the FCN popular in the modern solutions of related tasks, for example semantic segmentation. Since the output/ground truth and the input image are provided, the framework could perform feedforward and backward training in a straightforward way. Dense prediction. From the procedure of FCN one could obtain that, the output of different sampling layers selection is different in resolution. In these maps, the output are downsampled with the factor f due to the pooling operations. To obtain a refined prediction result from these coarse maps, one way is to upsample the map with corresponding factor according to the downsampling factor. Another way for the reconstruction is interpolation. For example, by using bilinear interpolation, the final output element y_{ij} is computed based on the nearest four elements from the coarse output, thus expands the size of the result.

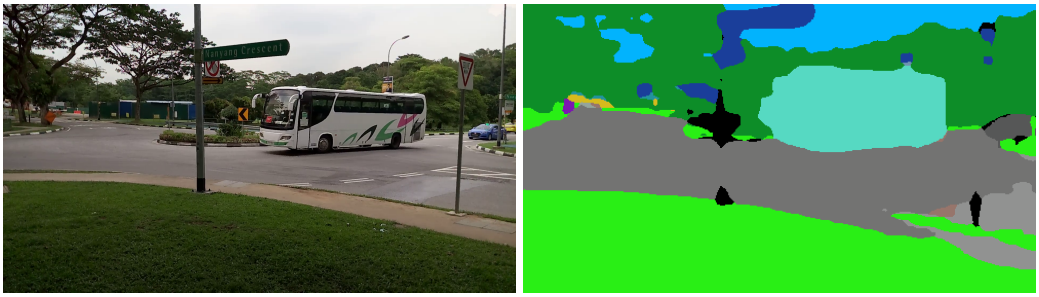

Figure 2.8: A initial result of scene recognition with FCN based on pascal-context dataset. The process is done without any post-processing.

In the formal release of the FCN framework, the method for acquiring a dense prediction with high resolution is concatenating multiple results from different layers output features. Since different layers output feature maps of different sizes, feature maps upsampling is executed. Based on the downsampling factor f, the raw output feature maps are upsampled accordingly with factor $1/f$. However,simply upsampling the feature map ignores information reconstruction which may lead to wrong prediction results. Therefore, the deconvolution operation is introduced. Deconvolution is a controversial operation against convolution, expanding one element to a feature map patch according to the deconvolution kernel. Different from convolution, deconvolution also expands the size of the feature map. By adding deconvolution and replacing upsampling, the information can be transmitted end-to-end and back propagation is straightforward in the framework. Note that similar to convolution, deconvolution has parameters for the framework to learn using BP.

Implementation

The FCN framework has been experimented on multiple datasets corresponded to semantic segmentation. The result shows that the framework is robust enough for handling multiple kinds of segmentation tasks.

As the example shown above, the FCN framework could generally construct a labeling map for images. However, there are still some short comings: (1) shapes of some objects are not precise; (2)Some objects that occupy small areas are not recognized. The poles,

and traffic signs from far away do not appear on the labeling map; (3) Some regions' classification is confusing. The sidewalk and the road are not clearly distinguished. These shortcomings are typical for scene recognition tasks. And several method are presented as contributions to solutions.

2.2.4 Integration on FCN framework

Although FCN provides an ideal solution for semantic segmentation, it still occurs some shortcomings. As pointed above, some detail failures of labeling might mislead understanding of the whole images. Several methods are used for better predictions. In this section, the improvement on FCN will be discussed.

Modern deep networks could collect rich information from images. However, images also produce noises that make features difficult to be recognized or classified. The noise usually comes from image details which both provide detailed texture features and possible misleading data for networks. Methods are needed to treat the textures that may interrupt performance of the networks. A possible way is to scale the image's or its features in different ratios. Traditional networks are using pooling layers to scale features in different steps of convolution.

Traditional pooling method is concatenated with convolution in the network. The main purpose is to expand receptive fields so that a feature element could represent wide region in the raw image. However, pooling in this way may lose some contextual information as it discards some information near the selected elements. Besides, as mentioned above, apart from denoising, the pooling also loses some texture feature. Therefore, pooling method need improvement to solve the problems.Here a new method is presented in which pooling with different ratio is executed in parallel way on the same feature map, which is called pyramid pooling[6].

The pyramid pooling module transforms features into four different pyramid scales. In this framework, a global pooling is used in a pyramid structure. The framework decomposes the input feature into several sub regions and executing pooling operations

on them, which is same as the typical neural networks. The resulted representations of the input are of different size. After the pooling, the outputs are upsampled and concatenated into one feature map for further labeling.

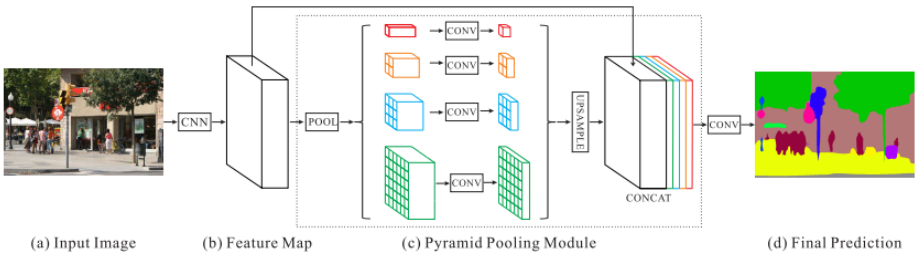

Figure 2.9: Illustration of pyramid pooling module[6].

The pyramid pooling is useful for store pooled information which may lose contextual information and some detailed data. Equivalently, it collects information from images of different scale at the same time, producing features from local to global scale. Thus the information of images can be better utilized compared to traditional FCN. However, as the pooling requires huge computation, it may possesses considerable space on computation devices.

2.2.5 Data

Review on dataset

For deep learning tasks, There are two things that directly define the algorithm and influence the performance. One is the network that was reviewed above. The other is the data, which will be discussed in this section. Dataset provides basic information for network to train and perform corresponding tests. According to the content of the dataset, networks are taught what to learn and the result they need to produce. Even for the same kind of datasets, the difference among them also influences the actual performance of the network.

In the field of scene recognition, The datasets are collected from multiple scenes, including urban area, wild place, mountains, forests, beach, and other categories. Each

kind of scenes contains objects and scene components of different kinds. Using them to construct a scene dataset with certain category distribution is the source of generating different scene recognition tasks. Networks that use specific dataset may perform well in corresponding tasks if well-trained. However, as the tasks are becoming general, whether a certain dataset and its network can preserve their performance is under discussion.

In this section, some datasets are reviewed to discuss how these data influence deep learning.

Pascal-context dataset

Pascal-context is a dataset extended from *PASCAL VOC 2010*. it labels every pixel of *PASCAL VOC 2010* with a semantic category. The dataset contains pixel-wise labels for the 10,103 training and validation images of the PASCAL VOC 2010 detection challenge. 540 categories are included in this dataset, ranging from little objects, vehicles, to huge buildings and scene components.

Figure 2.10: Examples of PASCAL context dataset[7].

The PASCAL VOC 2010 dataset has been a major benchmark for detection and segmentation tasks for years, showing its considerable ability of handling wide range of applications, and basis of adapting other datasets for fine-tuning.

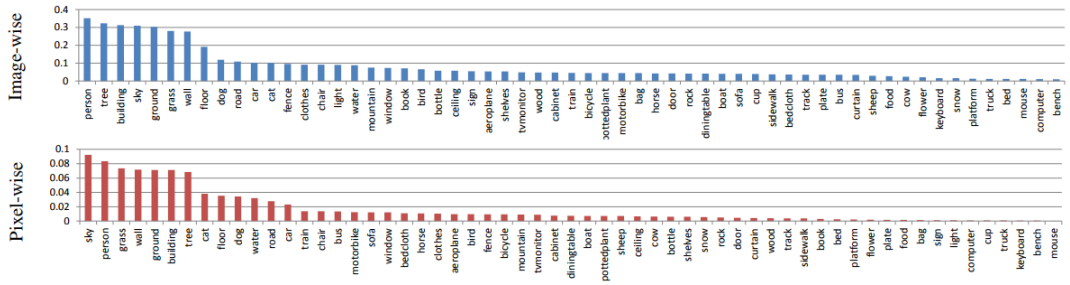

Figure 2.11: Distribution of pixels and images for the 59 most frequent categories (460 categories in all)[7].

The distribution of pascal-context dataset is fairly common. The dataset mainly aims at outdoor scenes. The most labeled pixel in the dataset consists of sky, road, person, grass and ground and buildings. Besides these labels, there are also several labels that rarely appears in the images. However, as they exist, researchers should pay attention to them and a new topic to process rare classes is established. In this field of research, rare classes are more concerned because their occupations are very little but usually crucial to help understand the whole images, e.g. traffic signs and traffic lights have always been the main target to be detected in traffic camera, but they occupy little areas in the image and may not appear in images in most of the images in the dataset. Similar cases have raised a requirement that rare class are needed for precise segmentation. Fortunately, this dataset provides the corresponding environment as well as general capability to train the network and get ideal results.

ADE20K dataset

The ADE20K dataset[8] is a typical data source for scene recognition which contains massive raw images and corresponding labels. In detail, the total number of the image for training is 20,210, validation 2,000 and test 3,000. All the images are annotated in pixel-wise mode with objects. For each object, the dataset also records that if it is cropped or overlapped. The dataset is still increasing in image number and categories' number.

The annotation of the images are crucial for dataset, which leads to high accuracy recognition in applications. In ADE20K, the images are precisely labeled by multiple kind of labels. The labeling data for each image consists of 3 layers, storing 3 kinds of labels: objects segments with names, parts and attributes. Each object has an independent annotation with proper label to ensure modern training methods could accurately pick information from it.

Figure 2.12: Examples of annotation on images of ADE20K. Multi-layer structure is used for storing the maps of same image[8].

Compared to other dataset, ADE20K contains more diverse scene, where the average number of object classes per image is 3 and 6 times larger than COCO and Imagenet, respectively. With respect to SUN, ADE20K his 35% larger in terms of images and object instances.

In this dataset,objects and scene components are clearly labeled, the labels are highly distinguished. For example, in the view of a car, the dataset labels it with categories car, wheels, car doors and glasses. This kind of segmentation make the dataset more general, not limited to specific missions. As mentioned above, there are three channels for a label map of an image. which stores different labeling results for different labeling tasks. Thus the dataset is able to handle more types of semantic segmentation.

Based on its huge volume of images and rich label categories, ADE20K dataset is able to be trained as a general model that network training on other datasets can use the

pre-trained model on ADE20K for fine-tuning.

Discussion

As an important component of deep learning, dataset should preserve its robustness and flexibility for their corresponding tasks. The training effect depends on both network setup and dataset choice.

An important criteria for the quality of certain dataset is the ability of training networks such that they could successfully complete certain task. There are several factors that influence their qualities. (1)The number of images that the dataset contains determines its training volume. As deep networks requires data to adjust their parameters, using data as much as possible enables parameters' computation more reliable. To come degree, more data input means more evidence of dataset distribution, as network receives enough evidence, the distribution of dataset and the task that the dataset represents is clear to it, making the network works better on corresponding tasks. (2)Different dataset has different contents and label categories, which defines the usage of datasets. Training network with dataset of certain label category limits its usage, e.g. a dataset of urban area make the network has ability to deal with problems of urban scene but usually results in failure in forest areas. (3)image distribution in datasets determines the label distribution. If certain label or several labels appear too frequently in the dataset, it may lead to label bias that causes the network learns abnormal distributions of images, leading to errors in actual applications.

However, there are some potential problems that need answers. First, should the volume of training data grow as large as possible? It is obvious that if a network receives more data, the training result would better reflect the representation of the dataset. However, dataset will expand itself as new samples are worth training. If the training processing takes more data than expected, what will happen to the network? Some experiments has shown that, for a general training task, if the data size is too big than expected, the network will fall into *overfitting* that only reflects dataset itself and has poor performance on actual application. As networks train more data, it is clear that the model that

network produces will better fits the dataset. One important factor of training is that, by learning the dataset, networks get related information and use them in other tasks that have similar information with the data, which means the network should consider its generalization ability instead of becoming the slave of data. Therefore, it's better to limit data volume to avoid overfitting.

Second, is there any challenge on training data? In the case of training scene recognition data, similarity among pixels confuses network. When pixels of similar textures are labeled with different categories, the network may have difficulties to separate these two categories in actual application. For example, After trained with a dataset of city scene, the network may be confused by street regions and road regions, which requires clear separation in some tasks. Data does not only contribute to networks, it also confuses networks with other properties. As mentioned above, labels distribution has influence on training effect. If certain label possesses large proportion of images, the model that is trained will give more weights to this label and the effect of the network will not be as ideal as one with commonly trained model.

Summary

In this section, two main dataset for scene recognition are introduced. They have significant features that can satisfy tasks of different related fields. Besides, the work of dataset are discussed, and two problems are presented. To sum up, dataset has equal importance with networks themselves. If networks act like sketches of a paint, datasets provides color and textures, which make the whole deep network task a completed and target-directed work.

2.3 Road lane marks detection

Methods for lane detection are classified into computer vision and graphical models approaches. In this section, several representative methods, including both non-machine learning ways and machine learning ways, are introduced. These methods are all based on image information from single camera which is easy to set up implementation environment.

2.3.1 Locating lane marks

A basic function of road lane marks is setting boundaries, both fixed and flexible, for moving vehicles. Localizing road marks provides supporting information for auto driving systems which control directions of the vehicles. In common scenarios, road lane marks are regular and the scene is ideal for visual computation, which benefits the processing of locating the marks.

A significant feature for road lane marks is edge. Most images provide clear view of road surface and the edges of the marks are easy to catch. One feature of the edges is that most of them are striaght lines. In some curve cases, the edges can also be decomposed into clusters of straight lines. Based on the feature, multiple methods of straight line detection can be utilized.

Determination of region of interest (ROI)

Before detecting the straight lines, images of road surface need adjustment for correct detection. In actual implemntation, input images contains not only road surface but also surroundings. The regions of surroundings contains multiple objects, for example, buidlings, banners, plants. Since the detecting method is executed over entire image, such surroundings generate some straight line instances that interrupt the final result. Therefore, such surroundings should be removed and the regions of road surface is preserved. The preserved regions is called region of interest (ROI). Defining ROI in the

image reduces the computation on the image by reducing the size of the image while focusing on the required regions.

Figure 2.13: Region without or with road[9].

Previous methods define ROI based on vanishing points, which is time-consuming and might leads to incorrect defining results. Therefore, this method establishes a ROI window of fixed position and size (fig. Fig. 2.13).

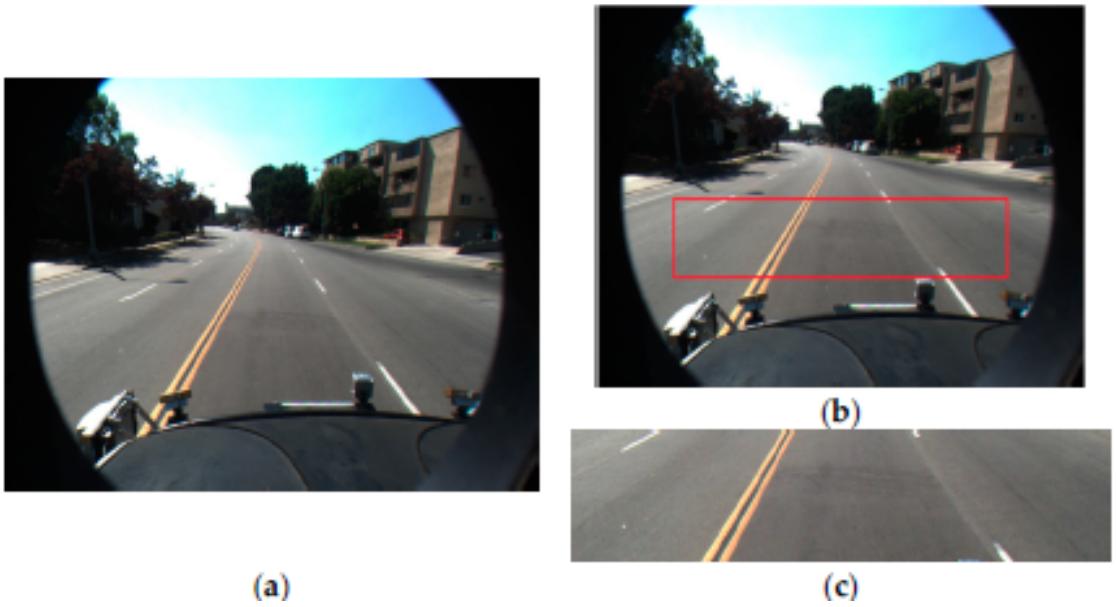

Figure 2.14: Determination of ROI. The red rectangle is the ROI window[9].

Lane marks detection

Generally, the road lane marks detection approaches first convert the input image into IPM or the bird's eye image. Such images will make the lane marks parallel to each

other, which further makes the process easier to extract corresponding features. However, actual implementations do not always meet the parallel conditions. Besides, the IPM view method requires camera position for parameter adjustment, while most datasets do not provide such information. Some other research approaches also utilize Hough transformation to detect lane marks' edges. However, the solution takes long processing time and too many incorrect results will occur. In addition, it cannot perform classification among dashed lane marks and solid marks.

In the solution, the line segment detector (LSD)[26] is used for locating the lane marks edges in an image. The LSD method process an image without parameters adjusting. The method produces line segments from an input and stores them as $S = \{s_1, s_2, ..., s_k\}$. Each line segment $s_i, (i = 1, 2, ..., k)$ is defined as:

$$s_i = \{x_{1i}, y_{1i}, x_{2i}, y_{2i}, \theta_i\}, i = 1, 2, ..., k \tag{2.17}$$

where (x_{1i}, y_{1i}) and (x_{2i}, y_{2i}) are the coordinates of the starting point and the ending point of line segment s_i, respectively. θ_i is the angle of line segment s_i and is calculated by:

$$\theta_i = \frac{180}{\pi} \arctan\left(\frac{y_{2i} - y_{1i}}{x_{2i} - x_{1i}}\right), i = 1, 2, ..., k \tag{2.18}$$

Based on the LSD algorithm, a coarse result of detection is made. However, there are still come detected segments that are not satisfy the requirement of correct detection. Therefore, further tune is designed for correction.

In the correction, some limitations are added for eliminating wrong segments. The input image is divided into two parts, left side and right side according to a vertical line in the center of the ROI image. For the left side, a limitation of line segment angle is set as $\theta_{\text{left}} \in [25°, 75°]$, and the one of right side is $\theta_{\text{right}} \in [105°, 155°]$. The complete

expression of this correction is:

$$S_{\text{left}} = \{s_i^L | x_{1i} \leq \frac{w_{\text{ROI}}}{2} - 1, \theta_i^L \in [25°, 75°]\}, i = 1, 2, ..., p(\text{if } H_{\text{ROI}}/3 \leq y_{1i} \leq H_{\text{ROI}} - 1)$$
$$S_{\text{left}} = \{s_i^L | x_{1i} \leq \frac{w_{\text{ROI}}}{2} - 1, \theta_i^L \in [\theta_i^{L*} - 10°, \theta_i^{L*} + 10°]\}, i = 1, 2, ..., p(\text{if } 0 \leq y_{1i} \leq H_{\text{ROI}}/3) \quad (2.19)$$

$$S_{\text{right}} = \{s_i^R | x_{1i} \leq \frac{w_{\text{ROI}}}{2} - 1, \theta_i^R \in [105°, 155°]\}, i = 1, 2, ..., q(\text{if } H_{\text{ROI}}/3 \leq y_{1i} \leq H_{\text{ROI}} - 1)$$
$$S_{\text{right}} = \{s_i^R | x_{1i} \leq \frac{w_{\text{ROI}}}{2} - 1, \theta_i^R \in [\theta_i^{R*} - 10°, \theta_i^{R*} + 10°]\}, i = 1, 2, ..., q(\text{if } 0 \leq y_{1i} \leq H_{\text{ROI}}/3) \quad (2.20)$$

where R_{ROI} is the width of the ROI region and H_{ROI} is the height of the ROI region.

Each line segment has a starting position and an ending position. In the solution higher position (lower in value in y-coordinate) is the starting position and the other end is the ending position. The sets of segments satisfying the conditions of Eq.2.19 and Eq.2.20 are obtained as S_{left} and S_{right}, respectively, and are considered correct line segments.

Refinement

The road lane marks detection methods' performance is generally influenced by the environments of the images which makes the method may detect multiple line segments from one edge of road lane marks. Thus the solution should further cluster the line segments which are recognized as a output from one lane mark.

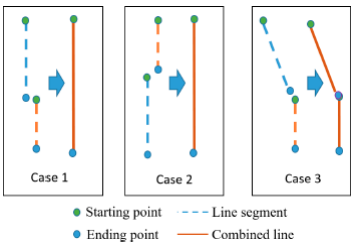

Figure 2.15: Angle feature used for eliminating wrong line segments[9].

Fig. Fig. 2.15 shows three cases of line segments combination. The first two cases construct one straight lane mark representation and the third constructs a curve representation. As mentioned above, there is a predestined rule that the starting point of a line segment is lower than the ending point in y coordinate.

Two conditions are considered on evaluating the connection of two line segments: the distance threshold(thr_{dst}) and angle threshold (thr_{angle}). The thresholds is set empirically according to the experiment results.

The thresholds is computed with the start point location of segment i and the ending point location of segment j, or the ending point of segment i and the starting point of segment j, according to the position of the i and j. The distance is denoted as $diff_{dst}$. If $diff_{dst}$ is less than thr_{dst}, the solution evaluates the condition of angle to determine if two lines segments belong to a straight line or a curve line. There is also a threshold thr_{angle} for determination. Line H is the output of combination on ith line with the jth line.

$$\text{Line } H = \begin{cases} \text{straight line, if } diff_{angle} \leq thr_{angle} \\ \text{curve line, if case 3 of Figure Fig. 2.15 occurs} \end{cases} \tag{2.21}$$

After the checking, the result of line marks detection is completed. Based on the result, the discrimination between solid lane marks and dashed lane marks is straightforward. Seen from the result, multiple detected segments are combined into long lines which represent multiple categories of lane marks. One thing worth attention is that, the solid lane marks always go across the ROI window, which means their starting points and ending points reaches the boundary of the ROI window. On the contrary, lines of dashed lane marks do not have such feature that either starting points or ending points of the lines do not reach the boundary of the ROI window, or both. Therefore, the classification method is completed. Set line H as a completed line, (x_{ht}, y_{ht}) as starting point of H and (x_{hb}, y_{hb}) as ending point of H. Also denote the left-top contour of the ROI window's

coordinate($x_{\text{ROI}}, y_{\text{ROI}}$):

$$\text{Line } H = \begin{cases} \text{Solid line, if } y_{ht} = y_{\text{ROI}} \text{ and } y_{hb} = y_{\text{ROI}} + H_{\text{ROI}} \\ \text{Dashed line, otherwise} \end{cases} \tag{2.22}$$

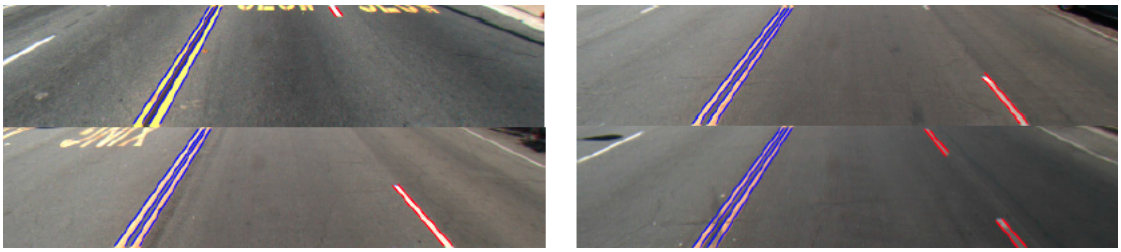

Figure 2.16: Example of the processing result[9].

Measurement

The method introduced in this section shows a standard solution of detecting lane marks and classifying them without using machine learning methods. The method used for detecting regions of marks is essential for most integrated detection methods. We notice that the solution is efficient for detecting lane marks in a limited regions, which is not general but useful in actual applications.The method uses a simple way to discriminate solid marks and dashed marks, which means it does not require much data for training a model.

On the contrary, Some shortcomings also negatively influence the performance of the solution.First, the solution only discriminates solid and dashed marks, while there are multiple categories of lane marks, for example double solid marks, zig-zags, road edges. Besides, some marks that do not shape the lanes, for example arrows, ground texts, also contain straight lines that are easy to detect using LSD. These factor make the classification of marks in this solution limited in general applications. Second, The ROI window defined in this solution is limited in collecting information of entire road, which causes some lane marks of other lanes cannot be seen in the window so that they cannot be detected. In some applications, detecting multiple lane marks from different lanes are

important for understanding traffic situation. Therefore, the solution is not compatible of handling complexed lane marks recognizing missions.

2.3.2 Road lane detection using frequency features

Besides utilizing straight lines, region information is also valuable for locating the lane marks. In the previous part, straight lines detection shows disadvantages of limited detection spaces that only small part of road surface are allowed for detection to suppress noise from surroundings. In this method, not only the edges of road lane marks, but also the regions around the edges are used for detection.

Different from pure edge detection, this method pay more attention to the orientation of edges, which needs information from the surroundings. In a typical road surface instance with road lane marks, an edge of one mark has an essential property that the mark appears on the one side of edge and road surface on the other side, which provide potential information more than naive edges' shapes and locations. To depict the property, the gradient map is utilized.

The advantage of using gradient map is that it shows orientations of the edges and it is easy to extract. In the road region, Such features can be efficiently extracted. However, the feature extraction will be influenced by illumination condition and shadows. As illumination condition changes, the difference between lane marks and road surface becomes smaller, and the gradient map on the boundary regions may not be as significant as common ones. To solve the problem, some improvements have been done to increase the features' quality. Some solutions uses additional devices to obtain information of other types. By using Hough transform, the straight lane boundary can be easily extracted without the influence of light and shadow conditions. However, it has difficulty on extracting curved lanes. Some solutions turns to spatial information for better performance, for example, Lane finding in ANother domAin (LANA)[27] extracts frequency domain information that shows the magnitude and orientation of the boundaries. Likelihood of Image Shape (LOIS)[28] method applies likelihood function into lane marker

detections using spatial edge information.

Based on the work above, This method[10] also uses gradient maps and frequency information on input images which sources from single camera. Different from the methods above, this method focuses on local patches features. The advantage of the this extraction method is that the gradient result is unaffected by the illumination conditions or shadows since it discards the environment's information and only focuses on boundaries regions.

In this method, The gradient features are used for detecting road lane edges as they significantly indicate changes from lane marks regions to road regions. Because lanes are commonly vertical to the horizon, the vertical gradient kernel is used for image convolution:

$$G(m,n) = I(m,n) * h(m,n) \tag{2.23}$$

where $I(m,n)$ is a grayscale input image, $G(m,n)$ is the corresponding convolution result and the convolution kernel:

$$h(m,n) = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \tag{2.24}$$

For an input image, five windows of 15×15 is used and randomly placed along the lane edges. These windows crop the convolved image and the patches are further transformed by Fourier transform (FT). An average is computed on these transforms' magnitude. This average reflects the overall condition of the image's lane marking, including the illumination and texture.

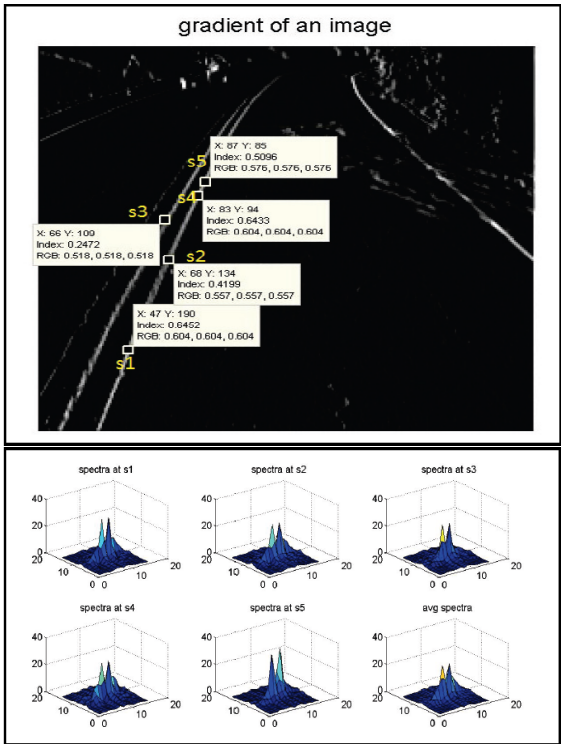

Figure 2.17: Illustration of choosing sample patches and computing average FT spectra[10].

Based on this average, further search on the image is executed and the searched patches are transformed using FT. The magnitude of these transformed patches are then compared to the average to determine whether the patch represents the lane edges. The detailed steps are as following:

- Set up several horizontal lines on the input image as searching lines. Each line has a 15×15 sliding window which moves along the corresponding line;
- The step of the sliding window is set as half of the window width. Each step of the sliding will crop the patch in the current window. The patch will be further processed as image gradient map.
- The patches cropped from the image is transformed using FT. The transformed map is then compared with the average map. If the correlation is high, then the patch is classified as region of road lane marks.

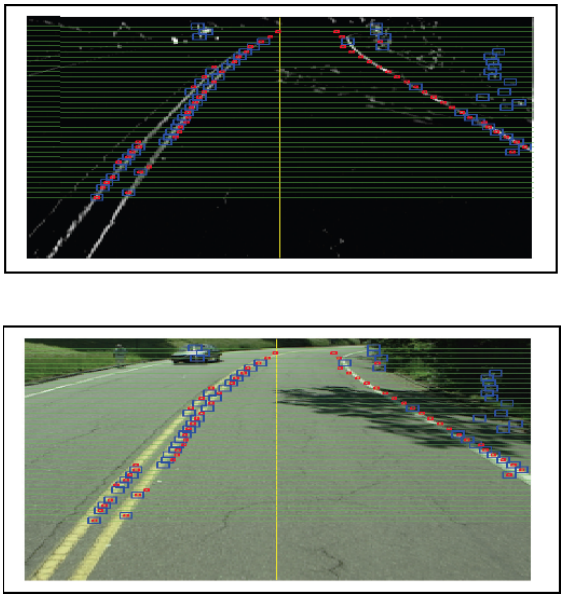

Figure 2.18: Illustration of detection using sliding windows[10].

Measurement

The method introduced in this section has the advantage of available for parallel computing, where searching can be executed simultaneously. Besides, it could handle different illumination conditions. However, the method occurs with some shortcomings. When facing with roads that are not clean, it may make some wrong detections. In some conditions, same road surface may differs in colors, which would make similar frequency maps as the one of road marks, making detection and classification difficult.

2.3.3 Recognizing road lane marks by extracting shapes

Locating the road lane marks provides useful information for road scene recognition. Related methods are capable of drawing representative lines of the lane marks. However, such methods ignore a problem that they cannot recognize the specific function of certain mark. In the previous reviewed methods, road lane marks are decompoed into parts for feature extraction. The reunion of the features does not consider the mode of the marks so that further classification is not developed. It is necessary to distinguish

the functions of detected marks so that benefits processing of auto driving.

By analyzing the road lane marks, a fact can be obatined that the functions are closely related to the lane marks' shape. For drivers' convenience, road lane marks should indicate their functions immediately after seen by drivers. Therefore, their shape are distinctive according to their functions. Based on this, classifying marks by utilizing shapes is a potential solution for this problem. Inspired by the methods reviewed above, a procedure for extracting shapes of the marks is developed.

Feature extraction

In this method, the multiple search lines is also estalbished. The difference is that two filters are used to search for the left edges and right edges of one mark instead of sliding patches. The two filters, f_l, f_r are used to detect edges of both sides of a mark by scanning an image $I(x, y)$ line by line. The width of the filter w_f is determined by analyzing the camera positions and corresponding sights.

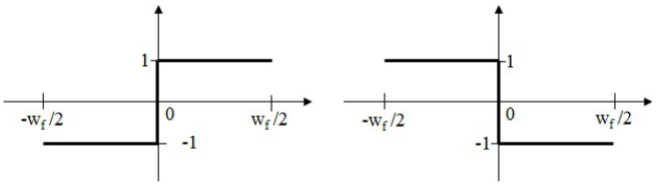

Figure 2.19: The illustration of filter functions for extracting left edges and right edges of input image.[11]

The filtered result $I_l(x, y), I_r(x, y)$ shows the outputs of the two filters in Fig. Fig. 2.19, and the images are further processed by thresholding $T(\cdot)$ for eliminating some noise to ensure that the features can be extracted without other useless factors in different

illumination conditions.

$$I'_l(x,y) = \begin{cases} I'_l(x,y) & \text{, if } I_l(x,y) > T(\frac{\sum_{t=x-w/2}^{x} I(i,y)}{w/2}) \\ 0 & \text{, otherwise} \end{cases}$$
$$I'_r(x,y) = \begin{cases} I'_r(x,y) & \text{, if } I_r(x,y) > T(\frac{\sum_{t=x}^{x+w/2} I(i,y)}{w/2}) \\ 0 & \text{, otherwise} \end{cases} \tag{2.25}$$

Based on the two processed edge maps, a lane marks feature $\mathbf{x_i} = (x_i, y_i)$ is generated. The feature contains the information of lane mark's location. The mark feature is determined by finding a maximum value of two filters' output in pair. However the maximum pair is not picked randomly. The distance between the two edges should be larger than a minimum estimated mark feature width w_m and less than a maximum mark feature width w_M. The value of these boundaries are predetermined by considering the camera position and corresponding sight, which is similar to the filter width w_f.

Marks generation

Based on the extracted features, lane marks extraction is further proceeded. As each search line on the image detects corresponding features of the road marks, clustering them into marks shapes' representation is straightforward. Generally, features comes from same mark have similar properties. Utilizing such properties, features are logically grouped.

This solution takes positions and angles properties as clusering parameters. After the clustering, the representations of the marks' shapes are generated. They are region segments that covers the marks in the images and depict their shapes and directions. In this solution they are called supermarkings. Algorithm 4 shows the detailed clustering procedure.

Algorithm 4 Supermarkings generation[11]

Input: The lane feature $\mathbf{x_i}$,N_x

$N_x \leftarrow 0$

for $i = 1$ to N_x **do**

  for $j = 1$ to N_x **do**

    if dist$(\mathbf{x_i}, \mathbf{x_{s_j}}, \theta_{s_j}) < \phi$ **then**

      $s_j \leftarrow \mathbf{x_i}$ and update $\mathbf{x_{s_j}}, \theta_{s_j}$

    end if

  end for

  if $\mathbf{x_i}$ is not assigned to any s_j **then**

    $s_{N_s+1} \leftarrow \mathbf{x_i}$

    $N_s \leftarrow N_s + 1$

  end if

end for

Output: The supermarking cluster s_j, N_s

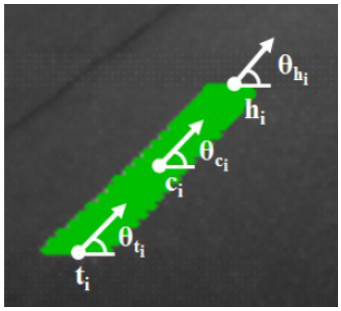

Figure 2.20: An example of supermarking s_i.[11]

In Algorithm 4, $\mathbf{x_{s_j}}$ and θ_{s_j} are the location and direction of the supermarking s_j in the image coordinates and N_s denotes the number of supermarkings.. The values will be updated if the cluster receives new patches of feature that is recognized as a component of this supermarking. The criterion of the recognition is that, it computes the visual distance between the current feature $\mathbf{x}_i$ and the supermarking s_j. If the value is lower than a predetermined threshold ϕ, then the feature is concluded into the s_j. On the contrary, if the feature is excluded, then a new cluster is set which starts with this feature.

The distance is calculated with the function dist$(\mathbf{x}_i, \mathbf{x}_{s_j}, \theta_{s_j})$ as shown in the algorithm.

To make further processing convenient, some symbols on each supermarking are denoted. As shown in the Fig.Fig. 2.20, there are some properties are set. $\mathbf{c}_i, \mathbf{t}_i, \mathbf{h}_i$ denotes the location of the center, tail, and head middle point of supermarking s_i, respectively. $\theta_{\mathbf{c}_i}, \theta_{\mathbf{t}_i}, \theta_{\mathbf{h}_i}$ denote the corresponding direction angle of each points, respectively.

Road lane marks clusters construction

Previous processing extracts shapes and positions of lane marks out of original images as segmentation maps. The output contains corresponding parameters, namely coordinates and angles of each supermarking. The next step is transforming individual marks into marks clusters. For example, several dashed lane marks can form a dash lane sequence on the road surface as a guide line for cars driving. In this part the CRFs are used. The brief introduction is shown in last chapter. The result depicts a refined representation of road lane marks and their representations of function on the road surface.

The clustering consists of two level: coarse and refined clustering. The coarse clustering is conducted to directly use the simple computer vision methods to cluster the supermarkings that are obviously clustered. For example, some dashed lanes which are very closed to each other can be clustered into one mark symbol. In this part, the clustering probability is calculated between arbitrary two supermarkings. The probability is calculated by the Eq. **??**.

$$P(s_i, s_j) = \frac{1}{z} \exp(-\frac{|\theta_{c_i} - \theta_{c_i c_j}|^2 + |\theta_{c_j} - \theta_{c_i c_j}|^2}{\delta^2}) \tag{2.26}$$

θ_{c_i} and θ_{c_j} are the direction angles of each supermarking symbols' center, and $\theta_{c_i c_j}$ is the direction angle of the vector $\mathbf{c_i} - \mathbf{c_j}$ that indicates the difference in direction between the centers of s_i and s_j. In order to complete the coarse clustering, supermarkings s_i and s_j should follow the criterion below:

- The supermarkings can not overlap with each other in the y-axis in the image coordinates;

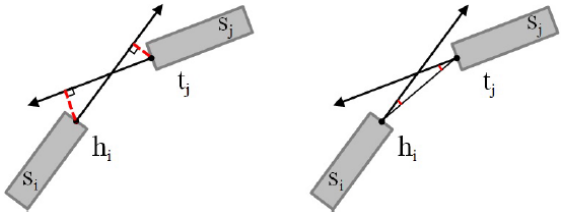

Figure 2.21: Illustration of measurement functions: Left: The geometric distance measurement function dist_{geo}. Right: The directionality measurement function dist_{dir}.[11]

- One s_i can only be clustered with one other supermarking s_j. And the clustering is determined by using checking the highest probability in M among all the cases in the iteration of i and j.

The next step is refined clustering with CRFs approach. A CRF graph $G = (V, E)$ is constructed based on all the supermarkings generated in the previous steps. In this graph, The vertices set V collects all the possible clustered pairs of supermarkings, and the edge set E of the graph stores the connection that may exist among the pairs in V. To construct vertices showed in the Fig Fig. 2.21, two measurements are built: dist_{geo} and dist_{dir}. These measurements output the geometric distances and directions of two supermarkings:

$$\begin{aligned} \text{dist}_{geo}(s_i, s_j) &= |(x_{t_j} - x_{h_i}) \sin\theta_{h_i} - (y_{t_j} - y_{h_i}) \cos\theta_{h_i}| \\ &\quad + |(x_{t_j} - x_{h_i}) \sin\theta_{t_j} - (y_{t_j} - y_{h_i}) \cos\theta_{t_j}| \\ \text{dist}_{dir}(s_i, s_j) &= |\theta_{t_j} - \theta_{h_i t_j}| + |\theta_{h_i} - \theta_{h_i t_j}| \end{aligned} \tag{2.27}$$

where h_i denotes the head of the supermarking s_i, t_j is the tail of the supermarking of s_j, and $\theta_{h_i t_j}$ denotes the slope angle of a line made by h_i and t_j.

In the refined clustering, supermarkings are recognized as one lane mark sequence and set as an element in vertices set V: $v_k = (s_i \to s_j)$ if both dist_{geo} and dist_{dir} are below certain thresholds that are set to contain all positive instances in the datasets. Edges in the CRF graph contains all the vertices connection that made in the previous step, making a sub set of vertices as a clique.

Each vertex v_k in the $V = \{v_1, ..., v_K\}$ has a label l_k that are valued as 0 or 1.If two

supermarkings are determined as clustered the value will be set as 1, otherwise 0. The label set $L = \{l_k, ...l_K\}$ performs as representation of supermarking clusters variable in the energy minimization process. The optimal output of the process could be regarded as the best clustering of the supermarkings. The clustering limits the total number of connections among the supermarkings that one head of a supermarking can connect at most one tail of another supermarking and vice versa. The head and tail of one supermarking can be found in the list of supermarking components while constructing the supermarking by checking their positions. The constraints is obvious that in the actual traffic road conditions, one lane mark can only serve as one function for driver to read and one lane mark can only be connected to one other mark to keep its function singularity. The mathematical expression of the criterion is as following:

$$\begin{aligned} \sum_{v_k \in H_i} l_k \leq 1, \text{where } H_i = \{v_i | (s_i \rightarrow s_j) \in V\} \forall s_j \in S \\ \sum_{v_k \in T_i} l_k \leq 1, \text{where } T_i = \{v_i | (s_j \rightarrow s_i) \in V\} \forall s_j \in S \end{aligned} \tag{2.28}$$

H_i collects all the heads of s_i that are connected in the vertices; and T_i collects all the tails of s_i that are connected in the vertices.

A CRF graph is constructed on the clustering criterion introduced above, in which its unary term represents the connection between two supermarkings and the pairwise term represents the clustering of these connections that forms the final clustering result.

$$\begin{aligned} P(L|S) &= \frac{1}{z} \exp(-\Psi(L|S)) \\ \Psi(L|S) &= \sum_{l_k \in L} U(l_k|s_k) + \sum_{C_{cl(G)}} \phi_C(l_C|s_C) \end{aligned} \tag{2.29}$$

In Eq. 2.29, $P(L|S)$ denotes the overall probability of a label set. $\Psi(L|S)$ is the energy of the clustering over entire graph. It is a summation over all the unary potentials $U(l|s)$ and pairwise potentials $\phi(l|s)$. C is a clique of clique group $cl(G)$ of entire graph G.

Here in this method, the unary term is designed to show how likely the two supermark-

ings are to be clustered, in which a quadratic regression is utilized.

$$
\begin{aligned}
U(l_k|s_k) &= -\ln(P(l_k|s_k)) \\
P(l_k|s_k) &= \frac{1}{1+\exp(-f_{\text{dist}}/\sigma)} \\
f_{\text{dist}} &= \begin{bmatrix} d_g \\ d_d \\ 1 \end{bmatrix}^T \begin{bmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{bmatrix} \begin{bmatrix} d_g \\ d_d \\ 1 \end{bmatrix} \\
d_g &= \text{dist}_{geo}(s_{k_1}, s_{k_2}) \\
d_d &= \text{dist}_{dir}(s_{k_1}, s_{k_2})
\end{aligned}
\tag{2.30}
$$

The coefficients in the quadratic regression model are trained from a dataset that the method has collected. In the dataset, the method made an annotation of the supermarkings with clusterings among them. To build the clique potentials model or representation, a linear regression is built to evaluate the probability of each clique. The potential of a clique can be calculated by the following equations:

$$
\begin{aligned}
\phi_C(l_C|s_C) &= -\ln(P(l_C|s_C)) \\
P(l_C|s_C) &= \frac{1}{1+\exp(-(\phi - f_{\text{err}})/\sigma)} \\
C \in cl(G), f_{\text{err}} &= \sum_{k=1}^{K(C)} |y_k - f_C(x_k)|^2 / K(C)
\end{aligned}
\tag{2.31}
$$

Figure 2.22: Result of CRF clustering. Left is coarse clustering and right is refined clustering.[11]

In Eq.2.31, f_{err} computes the average error between predicted model f_C and lane marks feature $\mathbf{x_k}$. The ϕ is the regression curve's center and σ is the degree of gradient of the regression curve, which is the parameter to be trained by the data. $K(C)$ is the number of lane marks' features of supermarkings in clique C.

After the definition of the potentials, the energy is obtained. The best result for label deployment $L*$ is determined by energy minimization.

$$\begin{aligned} L* &= \underset{L}{\operatorname{argmin}}\, \Psi(L|S) \\ &= \underset{L}{\operatorname{argmin}} \sum_{l_k \in L} U(l_k|s_k) + \sum_{C \in cl(G)} \phi_C(l_C|s_C) \\ &= \underset{L}{\operatorname{argmax}} \sum_{l_k \in L} \ln(P(l_k|s_k)) + \sum_{C \in cl(G)} \ln(P(l_C|s_C)) \end{aligned} \tag{2.32}$$

The final clustering map is conducted using the minimization result.

Measurement

The method introduced in this section is a quick and efficient solution of extracting segments of road lane marks. The result shows a precise segmentation on the normal road conditions. Moreover, it provides a robust way of filtering components on the road surfaces which could be adjusted according to camera position. However, the method is not capable of discriminating the categories of each segments. Also, if the road condition is complexed, or illumination condition is changing, some components that are not of lane marks are tend to be segmented.

2.4 Classification using Extreme Learning Machines (ELM)

2.4.1 Single layer feedforward network

Extreme Learning Machines is a single hidden-layer feedforward neural network which has developed multiple variations[29][30][31]. Standard ELM contains three layers for

computation: input layer, hidden layer and output layer. Hidden layer consists of a number of nonlinear nodes whose number is defined by actual implementations. There is a connection with randomly initialized weights and bias between the input layer and hidden layer. The connection between hidden layer and output layer is computed according to the input data and corresponding ground truth.

For N training samples $(\mathbf{x}_i, t_i)$ where $\mathbf{x}_i \in \mathcal{R}^p$ and $t_i \in \mathcal{R}$, ELM can be formulated as:

$$\sum_{i=1}^{L} w_i g(W_{in(i)} \cdot \mathbf{x}_j + b_i) = o_j, j = 1, ..., N. \tag{2.33}$$

This equation denotes the forward processing of ELM. $\mathbf{x}_j$ is input vector, and W_{in} is weight layer that is randomly generated. Similar to typical machine learning methods, ELM also has a bias b_i which is also generated. $g(\cdot)$ is a nonlinear activation function, which generates the hidden layer. The variable $w_i \in \mathcal{R}$ is the output weight connecting hidden layer i and output vector. $o_j \in \mathcal{R}$ is the corresponding output of input $\mathbf{x}_j$. $g(\cdot)$ can be compressed into matrix-vector form:

$$H\mathbf{w} = \mathbf{o} \tag{2.34}$$

where

$$H = \begin{bmatrix} g(W_{in(1)} \cdot \mathbf{x}_1 + b_1) & \cdots & g(W_{in(L)} \cdot \mathbf{x}_1 + b_L) \\ \cdots & \cdots & \cdots \\ g(W_{in(1)} \cdot \mathbf{x}_N + b_1) & \cdots & g(W_{in(L)} \cdot \mathbf{x}_N + b_L) \end{bmatrix}_{N \times L} \tag{2.35}$$

$\mathbf{o} = [o_1, ..., o_N]^T$ and $\mathbf{w} = [w_1, w_2, ..., w_L]^T$. H is the hidden layer The ith row of H $\mathbf{h}_i$ is the hidden representation with respect to inputs $\mathbf{x}_i$. Since $W_{in(i)}$ and b_i are randomly generated before computation, $\mathbf{h}_i$ is only related to the inputs.

If the ELM model with L hidden nodes can learn these N training samples with no

residuals, then w_i exists to make ELM trainable with following processing:

$$\sum_{i=1}^{L} w_i g(W_{in(i)} \cdot \mathbf{x}_j + b_i) = t_j, j = 1, ..., N. \tag{2.36}$$

where t_j is the final output or the ground truth of ELM. And

The matrix form of Eq.2.36 can be written as

$$H\mathbf{w} = \mathbf{t} \tag{2.37}$$

where $\mathbf{t} = [t_1, ..., t_N]^T$ is the output vector. Since the As the input weights and the hidden layer bias have been randomly chosen in the beginning of learning, $W_{in(i)}$ and b_i are randomly generated, Eq.2.37 becomes a linear parameter system. In ELM, the output weight is computed using smallest norm least squares.

$$\mathbf{w} = H^{\dagger}\mathbf{t} \tag{2.38}$$

where $H^{\dagger}$ is the Moore-Penrose generalized inverse of H.

2.4.2 Regularized ELM

Standard ELM has advantages in training speed and shows generalization performance. However, it faces some drawbacks. First, ELM has high probability of facing overfitting problem because it is a instance of empirical risk minimization principle. Second, it directly calculates minimum norm least-squares solutions which may provide weak control capacity. Third, the estimation that the algorithm perform may be less robust. Based on these problem, the regularized ELM[31] is developed.

The regularized ELM is based on structural risk minimization (SRM) principle of statistical learning theory[31]. The method could then be expected for better performance compared to standard ELM. Both standard ELM and regularized ELM uses Moore-Penrose generalized Inverse for final output. Therefore, it is expected to provide better

performance than standard ELM. Both regularized ELM and standard ELM uses Moore-Penrose generalized inverse to compute final result.

The regularized ELM is presented for solving the problem of prediction risk. This problem is defined in the theory of statistical learning. In detail, the risk contains empirical learning risk and structural risk. In the view of machine learning, strong ability of balancing the two risks is an important sign of good generation of a model. The prediction risk is a weighted summation of empirical risk and structural risk, which is represented by $||\varepsilon||^2$ and $||\beta||^2$, respectively. Their weight can be regularized using γ for empirical risk. In order to acquire a robust estimation that suppresses outlier interruption, the error variable $||\varepsilon_j||^2$ is expanded into $||D\varepsilon||^2$ where $D = dialog(\nu_1, \nu_2, ..., \nu_N)$ and $\varepsilon = [\varepsilon_1, \varepsilon_2, ..., \varepsilon_N]$. Such modification can adjust each ε in different weights. Based on the settings above, regularized ELM model is proposed as:

$$\begin{aligned} &\min \frac{1}{2}||\beta||^2 + \frac{1}{2}\gamma||D\varepsilon||^2 \\ &s.t. \sum_{i=1}^{\tilde{N}} \beta_i g(w_i x_j + b_i) - t_j = \varepsilon_j \\ &j = 1, 2, ..., N \end{aligned} \tag{2.39}$$

where $\tilde{N}$ is the number of hidden nodes in the hidden layer of ELM. By regularizing γ, the empirical risk and structural risk are adjusted in proportion. The best balance between the risks leads to optimal generation performance of the model. The procedure is done by Lagrangian on Eq.2.39:

$$\begin{aligned} &L(\beta, \varepsilon, \alpha) \\ &= \frac{\gamma}{2}||D\varepsilon||^2 + \frac{1}{2}||\beta||^2 - \sum_{j=1}^{N} \alpha_j (\sum_{i=1}^{\tilde{N}}) \beta_i g(w_i x_j + b_i) - t_j - \varepsilon_j \\ &= \frac{\gamma}{2}||D\varepsilon||^2 + \frac{1}{2}||\beta||^2 - \alpha(\mathbf{H}\beta - \mathbf{T} - \varepsilon) \end{aligned} \tag{2.40}$$

where $\alpha_j \in R(j = 1, 2, ...N)$ is the Lagrangian multiplier with the constraints of

Eq.2.39. The result for the Lagrangian optimization is:

$$\alpha = -\gamma(\mathbf{H}\beta - \mathbf{T})^t \tag{2.41}$$

Thus β is calculated as:

$$\beta = \left(\frac{I}{\gamma} + \mathbf{H}^T D^2 \mathbf{H}\right)^{\dagger} \mathbf{H}^T D^2 \mathbf{T} \tag{2.42}$$

The expression Eq.2.42 only involves the inversion of a matrix of order $\tilde{N} \times \tilde{N}$, where $\tilde{N} << N$. Thus the speed of computing β is quick.

When $D = dialog(\nu_1, \nu_2, ...\nu_N)$ as a unit matrix I, β can be calculated by the following expression:

$$\beta = (\frac{I}{\gamma} + \mathbf{H}^T\mathbf{H})\mathbf{H}^T\mathbf{T} \tag{2.43}$$

This equation is called unweighted regularized ELM. Obviously, when $\gamma \to \infty$, Eq.2.43 becomes standard ELM.

Chapter 3

Experimental Study on Scene Recognition Using Deep Learning Methods

3.1 Implementation on STE project

The research work contains a corporation project task. The project demands a segmentation map on entire scene images. In detail, the segmentation result should contain major traffic components and objects. Besides, some objects on the streets are expected to be recognized. Obviously, the existing FCN framework is not capable of completing the task. In addition, PSPNet may be difficult for the task because it take too many memory space. To both satisfy the task of refined segmentation and requirement of less memory occupation, the method uses dilated convolution for segmentation network improvement.

In traditional FCN framework, there are pooling layers between convolutions. The pooling computation reduces feature maps' spatial dimension and select the most significant feature elements for further computations. The advantage of pooling is that it usually eliminates noise feature elements, expands features' receptive field, and makes feature

learning more compatible. However, pooling operation discards information with a fixed criteria, e.g. max pooling stores elements with the highest value and discard others. Such operations keep significant information while have possibility of losing non-significant but valuable data. Furthermore, the information loss indicates that there are some data from raw image is lost, leading to difficulty of restoring dense prediction of raw size.

Different from traditional networks, a convolutional network without pooling layers might bring some different to prediction performance. Such networks need to pay attention to both computation speed and accuracy. A new kind of convolution kernel is introduced to satisfy the requirement. This kernel is called dilated convolution[12]. Consider a convolution processing. In traditional processing, the convolution kernel scans elements one by one without skip. In the way of dilation, the kernel is 'dilated' that grabs elements from wider positions. The illustration of dilated convolution is shown in Fig. 3.1. The figure uses red dots as convolution points and cyan patches as receptive field. The first map depicts the first convolution step in the whole processing, which is the same as the normal convolution. The second map shows the following dilated convolution. In this step, there are strides among convolution points. If this step executes normal convolution, the receptive field show be 5×5, as each points receives a 3×3 region's information from raw image and 9 points are clustered into a convolution kernel. In dilated convolution, these 9 points are clustered with strides 1, so that the locations of receptive fields are expanded compared to the one of normal convolution. Thus the receptive fields is larger in dilated convolution. Similarly in the third map, as the strides among convolution points increases, the region of raw image information received by the points are further expanded. Such receptive field expansion has equivalent effect as the pooling processing. Thus the pooling layers can be removed in dilated convolution to acquire better resolution of the features.

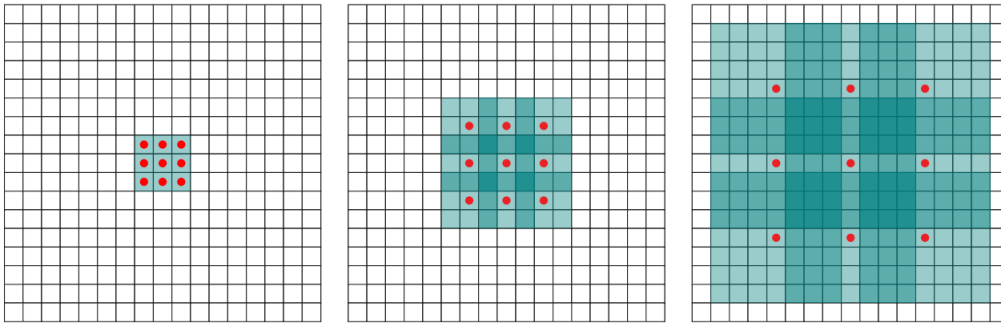

Figure 3.1: The illustration of dilated convolution. The cyan blocks are receptive fields and the red dots are elements to be convoluted. The first image shows that the convolution kernel is 3×3 and the receptive field size is 3×3. The second convolution uses kernel with dilation $D = 2$ (written as 2-dilated) and the receptive field is 7×7. The third uses 3-dilated kernel and the receptive field is 15×15. [12]

As the convolution goes deeper, each element's receptive field becomes bigger with size $(2^{i+2} - 1) \times (2^{i+2} - 1)$($i$ is the index of layers). By using dilation, receptive field expansion is realized without losing any elements' data. Therefore, the pooling operation is no longer needed.

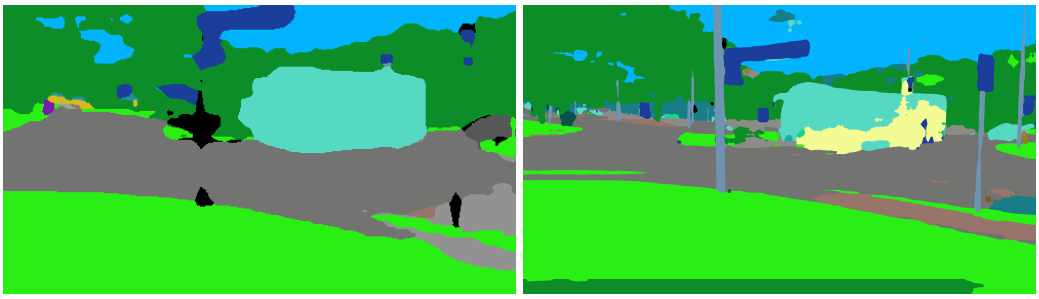

Figure 3.2: Left: Dense prediction using standard FCN; Right: Dense prediction with dilated kernel

Seen from the comparison figure, the dilated convolution considerably improves the segmentation effect, some small areas are clearly labeled and the shape of the bus is better described. Admittedly, there are some labeling errors on the map, from which further improvements are expected.

Method	Overall accuracy	Mean IoU
Conv. Dilation	84.2%	67.1%
FCN+CRF	71.5%	42.5%
FCN-16s	66.9 %	38.4%

Table 3.1: Comparison based on pascal-context dataset

3.2 Multi-task processing on scene recognition using edcge information

3.2.1 Introduction

Scene recognition, as a segmentation-based task, is evaluated with following criteria: First, the recognition result should correctly describe the objects' and scene components' shape. Second, the segmented objects or components should be in the right location with respective to the raw images. Third, classification on these components and objects should be correct and logical.

Recent researches on scene recognition have considerably improved classification accuracy and segmentation precision. Most of the popular methods for segmentation mainly focus on predicting a category label for each pixel. However, these methods pay less attention on objects and scene components' shapes. A shape mismatch is very likely to occur in the training process. When trained with large datasets, the shape problem could be alleviated, but this could either lead to over-fitting or the training processing being too time-consuming. On the other hand, a scene image is filled with objects and scene components. An important factor to divide these parts is the *edge* among them. The pixel information mainly represents an image's content features, while edge information could provide its structural features. **Proper edge information, or edge feature, has potential of improving image segmentation performance**.

The basic purpose of utilizing edge information is to enhance the segmentation perfor-

mance. In this work, the method is solely based on CNNs, without using external solutions, e.g. CRFs, super-pixels or auto contexts, to evaluate the method's performance correctly. The procedure is divided into two sub-missions: 1) Pixel-wise segmentation, which is similar to tradition segmentation methods. 2) Edge extraction, which focuses on 'labeling' edges of an image.

We propose a simple multi-task framework that is used to utilize both pixel-wise labels and edge maps. In this framework, pixel-wise feature and edge feature share one network model, which saves time of training. And the prediction of segmentation and edge map are calculated simultaneously. The training procedure of the method is totally end-to-end.

The challenge of enhancing segmentation with edge map is two-fold. First, the combination of pixel-wise features and edge features is quite straight-forward. As they share one network for feature extraction, the interaction between these two extraction procedures has unknown influence on the network. Second, one purpose of this method is to get structure information of images. The information is useful on some specific tasks, e.g. medical image segmentations. However, scene recognition uses data from different scene categories and the images do not share a uniform structure, which may make the structure learning more complex.

3.2.2 Multi-task framework

Multi-task framework has been proposed as a method of improving the generalization of a classifier by forcing it to learn more than one related tasks[32]. What is learned for each task can help other be learned better.

Generally, two related tasks are processed in the same framework. It has been proved that multi-task network can perform back propagation, which means that modern deep neuron network could utilize the multi-task network. In single-task training, the back propagation might fall into local minima. By adding an extra task to the network, it might help solve the problem because the local minima of the new task may be in

different place, which indicates that the multi-task framework could help find a more optimized solution for one task.

Besides enhancement, there might be competition inside multi-task frameworks. As two tasks share one input feature and one network, each hidden unit may receive information from two related but different source. During training, it is unknown if a hidden unit has tendency for certain task[33]. This competition will cause hidden units to be trained for a task only if the force made by one task is stronger than other tasks. Therefore, the main target of the multi-task should be determined such that the 'force' of the tasks involved with the framework can be manually adjusted for an ideal result.

In the task of scene recognition, the network takes an image of arbitrary size as the input, and outputs two kinds of maps, a pixel-wise segmentation map and an edge map. The framework is simple: A VGG-16[34] based FCN network as the main network, two branches for pixel-wise label prediction and edge prediction, respectively.

The key point of this framework is that two prediction outputs share one network, which means that they use the same feature map for processing. Traditionally, FCN provides feature map only for image segmentation. The feature only contains information for every pixel. However, an ideal scene recognition result requires representation for the whole image, which indicates that the image structure is also important for segmentation performance. Therefore, the image feature should contain pixel information as well as structural information.

The main advantage of multi-task framework is that it could enhance one task with another task's information. What directly influences the performance of the framework is the way of combining to tasks' information. Obviously, a standard multi-task framework should share the same input feature. My method mainly focuses on the way to share the input feature.

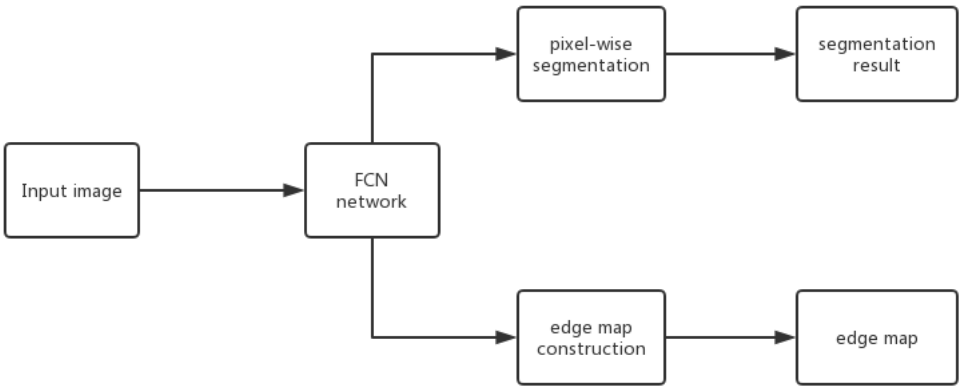

Figure 3.3: The overall structure of multi-task framework for scene recognition

3.2.3 Network design

The multi-task framework is trained in a similar way as common deep network. The significant difference is that, as it requires ground truth of two tasks, the loss computations will be executed simultaneously. In one training procedure, a raw image is inputed. After the network's computing, two outputs, i.e. pixel-wise segmentation prediction and edge map prediction, are generated. Then these outputs are used to compute loss respectively with their corresponding ground truths, based on which the back propagation is performed.

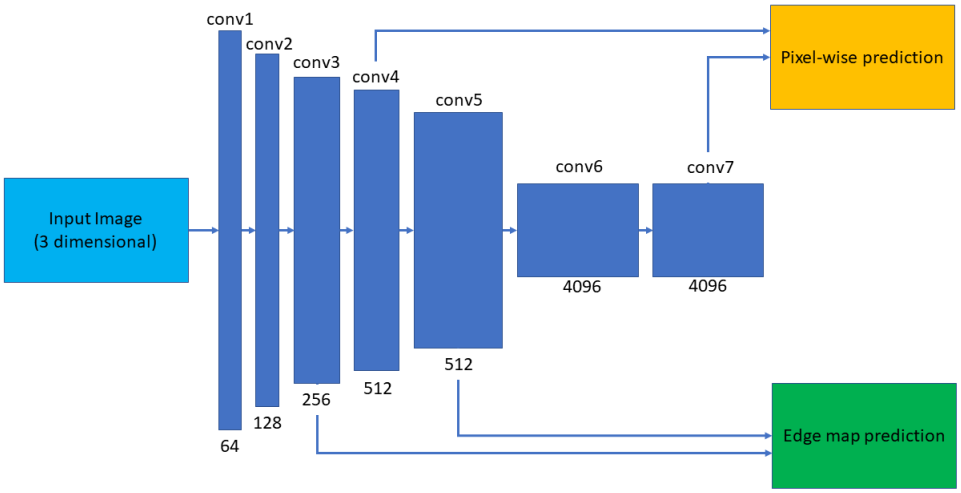

Figure 3.4: Detailed structure of the multi-task network. The network is based on FCN-16s network, and edge map prediction uses different features for prediction from the one of pixel-wise prediction

In the pixel-wise branch, the prediction uses feature map from layer $conv4$ and $conv7$. Layer $conv4$ provides detailed but not image specified feature maps, which contributes to training effect; $conv7$ layer brings high-level semantic features for better classification for each pixels. This selection is based on the experiment from original FCN semantic segmentation. The experiment showed that using these two layers' features, the prediction could restore an ideal pixel-wise labeling map. The result showed in figure Fig. 2.8 is based on this feature map combination. More feature map concatenation may improve the segmentation accuracy but also bring inconsistent labeled regions. Thus only two feature maps are chosen for prediction. In the edge map branch, the semantic information is not necessary for edge detection does not need to know the information of pixel classification. Instead, the detailed information helps find boundaries between regions. Thus the $conv3$ and $conv5$ layer are chosen. Further experiments are needed to measure whether these selections are optimized. The final network design is presented in figure Fig. 3.4.

The key point is how to combine two losses together. The edge map can provide certain information to the network, however, its negative influence cannot be ignored. The main

purpose of the network is scene recognition, which requires edge information's support, not pollution. Therefore, a set of loss weight is needed for both pixel-wise predictions and edge map predictions.

For the detailed computation method, the loss is computed with Softmax Loss.

$$p(y=j|\mathbf{x}) = \frac{\exp(\mathbf{x}^\top \mathbf{w}_j)}{\sum_{k=1}^{K} \exp(\mathbf{x}^\top \mathbf{w}_k)}, \quad j = 1, ..., K \tag{3.1}$$

The above equation computes the possibility that the input pixel variable $\mathbf{x}$ is assigned to label $y = j$. If this assign is different from the ground truth, the loss will be used to correct this difference:

$$\tilde{l}(\mathbf{x}, \mathbf{w}, y) = -\log(\frac{\exp(\mathbf{x}^\top \mathbf{w})_y}{\sum_{k=1}^{K} \exp(x^\top \mathbf{w}_k)}) \tag{3.2}$$

In this task, this loss is computed for both pixel-wise loss and edge map loss. For pixel-wise loss, the loss equation can be re-written as:

$$\tilde{l}(\mathbf{x}, \mathbf{w}_o, y) = -\log p_o(\mathbf{x}, y_o(\mathbf{x}), \mathbf{w}_o) \tag{3.3}$$

where $y_o(\mathbf{x})$ denotes the predicted pixel label y_o based on input $\mathbf{x}$, and p_o represents the possibility of the $\mathbf{x}$ labeled as $y_o(\mathbf{x})$. For the parameter $\mathbf{w}_o$ which denotes the network weights from input to pixel-wise output , because the network is separated into main network and pixel-wise branch, to make the notation more clear, the term $\mathbf{w}_o$ is divided into W_o and W_s, which represents weights of main network and weights of pixel-wise branch network, respectively. Thus the loss can be written as:

$$\tilde{l}(\mathbf{x}, \mathbf{w}_o, y) = -\log p_o(\mathbf{x}, y_o(\mathbf{x}), W_o, W_s) \tag{3.4}$$

Then sum up all the $\mathbf{x}$ in the image $\mathcal{X}$:

$$\mathcal{L}_o(\mathbf{x}, \mathbf{w}_o, y) = -\sum_{\mathbf{x} \in \mathcal{X}} p_o(\mathbf{x}, y_o(\mathbf{x}), W_o, W_s) \tag{3.5}$$

where $\mathcal{L}_o(\mathbf{x}, \mathbf{w}_o, y)$ denotes the loss of the whole image.

Similarly, for the edge detection branch, The solution defines $\mathbf{w}_c$ as weights of the network from input to edge map output, W_c as the branch's weight, $p_c(\mathbf{x}), W_c, W_s))$ as the possibility of the $\mathbf{x}$ labeled as $y_c(\mathbf{x})$ and $\mathcal{L}_c(\mathbf{x}, \mathbf{w}_c, y)$ as the whole image's loss on edge detection. The loss can be expressed as:

$$\mathcal{L}_c(\mathbf{x}, \mathbf{w}_c, y) = -\sum_{\mathbf{x}\in\mathcal{X}} p_c(\mathbf{x}, y_c(\mathbf{x}), W_c, W_s) \tag{3.6}$$

For the total loss of the network, the method simply sums up the branch loss together. Here the expression of the total loss is:

$$\begin{aligned}\mathcal{L}_{\text{total}}(x; \mathbf{w}) &= \lambda\psi(\mathbf{w}_) + \mathcal{L}_o(\mathbf{x}, \mathbf{w}_o, y) + \mathcal{L}_c(\mathbf{x}, \mathbf{w}_c, y) \\ &= \lambda\psi(\mathbf{w}) - \sum_{x\in\mathcal{X}} \log p_o(x, l_o(x); W_o, W_s) - \sum_{x\in\mathcal{X}} \log p_c(x, l_c(x); W_c, W_s)\end{aligned} \tag{3.7}$$

Similar to normal FCN networks, the network parameter $\mathbf{w} = \{\mathbf{w}_o, \mathbf{w}_c\}$ is optimized by minimizing the $\mathcal{L}_{\text{total}}$ with back propagation mentioned above.

3.2.4 Implementation

In the actual training and testing, PASCAL VOC 2010 dataset with pascal-context labeling set is used as the data source. The edge map ground truth is generated from the pixel-wise map which uses spatial differential. In each edge map, the edge pixels are marked as 1 and others are 0. The entire framework is constructed on Caffe[35] and executed in Python environment. Each training set is inputed with size unchanged. Training period is 800 and then a test is performed.

However, this framework proposal does not return expected results. The result in the figure shows that. This method can not even provide available predictions. The analysis of this failure will be discussed below.

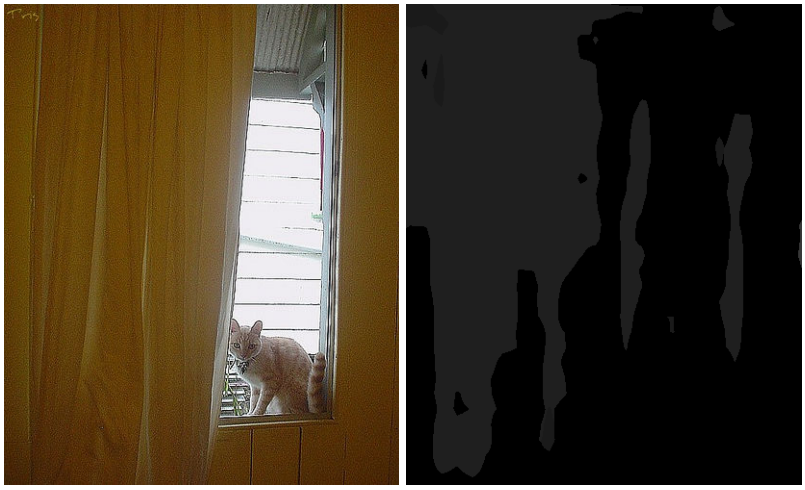

Figure 3.5: The left is input image, and the right is the prediction result.

3.2.5 Experiment results

Based on the implementation described above, the experiment is implemented on VOC2012 dataset. The VOC2012 dataset provides not only segmentation labels but also edge labels, 1 as edge regions and 0 as others, which is ideal for training the proposed network. The experiment returns bad results as seen in the 3.5. The segmentation map cannot depict the scene components and objects segments at all. The experiment on the edge information contribution to semantic segmentation ends with failure.

3.2.6 Problem analysis

This method requires good performance for both segmentation and edge detection. However, naive FCN network does not satisfy effective edge detection. In this task, the final result is not good that the shape of objects cannot be recognized and the scene recognition is not as good as traditional way. In this section the reasons are analyzed.

Since the framework is modified from a typical segmentation network which is successfully trained with certain model, the problem may be sourced from edge detection part. In this method, edge detection and pixel-wise segmentation uses same network procedure, which means that their ways of processing information are the same. However,

the difference between two training ground truth indicates that the network cannot be considered as similar.

Different from pixel-wise segmentation, edge detection aims at judging whether a pixel is on edge, which indicates that the task is in fact a two-fold classification. Besides, in the view of edge detection ground truth data, there is a significant difference between it and semantic segmentation ground truth. The edge detection's label maps only contain two categories, as it is for two-fold classification, and the distribution of the number of these categories is fairly imbalance. In a typical edge map, only marked edge pixels are set as positive and all the other pixels are negative, which means that negative regions occupy huge part of the whole image. Simply using segmentation network has little effect on this kind of map because most features extracted by the network are classified as negative such that the generalization of the classifier makes it hard to distinguish edge features and non-edge features. Thus this naive edge detection network is difficult to make effective prediction, and cannot provide useful information back into the main network to enhance pixel-wise segmentation.

Facing this problem, an alternative method is needed as solution. Initially, as edge detection is a two-fold classification, the computation of the training loss only contains two categories[36]:

$$-\mathcal{L}_{\text{edge}}(x;\theta) = -\sum_i y_i \log P(y_i = 1|x_i;\theta) - \sum_i (1-y_i) \log P(y_i = 0|x_i;\theta) \quad (3.8)$$

where i is the index of pixels, x is input feature element, y is predicted label, and θ is network parameter. In the case of edge detection, a pixel is marked as 1 if classified as edge, otherwise 0. In this equation, the sum of pixels marked as 0 is much bigger than the one of pixels marked as 1. Consequently the loss is influenced mostly by the negative part. To solve the problem, the loss is rewritten with proper weight:

$$-\mathcal{L}_{\text{edge}}(x;\theta) = -\sum_i \alpha y_i \log P(y_i = 1|x_i;\theta) - \sum_i \beta(1-y_i) \log P(y_i = 0|x_i;\theta) \quad (3.9)$$

where α and β are corresponding weights for positive loss and negative loss, respec-

tively. By adjusting the ratio between α and β, the influence of negative loss will be reduced. The detailed ratio is to be confirmed by more experiments.

Another possible solution is to concentrate on the way of extracting features[37]. As discussed above, edge detection and pixel-wise segmentation cannot be considered as similar tasks. Edge detection may require other methods of image representation. Since the edge ground truth map is imbalance in categories, considering changing the ratio of the positive and negative may improve the detection performance. One way to change the ratio is using image patches as training input.Before training, the image and corresponding edge map is separated into small patches. each patch, if the edge label is crossing the center of the patch, can be defined as the positive patch, otherwise the negative. Normally, a positive patch has higher ratio between edge pixels and negative pixels, and the image patch usually contains two kinds of regions, which make the network much easier to extract features. In the training procedure, the training patches are sent into the network in spatial order, and the edge map of whole image reconstruction is performed after edge detection.

Alternatively, edge shape on each positive patches may vary in different classes. To further ease the training complexity, similar shape of edge can be clustered into one class which is represented by one edge shape. The edge detection could then be transfered into shape classification problem. The only difficulty on this method is to construct such edge representation clusters.One problem for this method is the combination with original pixel wise network, which may occur huge space of device memory and some training procedures have to be compact enough to save computation resources.

3.2.7 Conclusion

In this section, the development of Scene recognition is reviewed and an experiment is performed for measuring and evaluating the edge information on recognition performance through multi-task networks. The results shows that, traditional segmentation methods is not applicable for edge detection, because the features for representing edges

are not similar to the ones for representing regions. Simply synthesizing their extraction proceedings does not enhance the representation ability on the whole image. The competition of back propagation between these two branches consequently makes damage on original network parameters which should have been available for image segmentation. Therefore, to optimize the performance of the scene recognition, further works should be done on representing images' texture, color, and other regional information.

Chapter 4

Multiple Road Lane Marks Detection Using Machine Learning Methods

4.1 Introduction

Based on the methods reviewed above, one fact can be obtained that, these methods have relatively robust ability of detecting the road lane marks. By extracting corresponding features, the methods could depict a picture of road lane mark representation. The methods used in the solutions are basically from classic computer vision techniques, which proved robust by multiple experiments[38].

Different approaches are utilized for extracting shapes' feature for classification[39]. For example, the method mentioned above used ROI- associated method to discriminate solid lane marks and dashed marks. Some methods record the shape feature of the marks, for example, two solid line of double solid marks, blank between two dashed marks and simple straight white mark of single solid marks. Based on these features, the detected marks are classified.

The methods mention in the previous chapter are fundamental for modern marks classification that requires machine learning for higher accuracy. One requirement for such approaches is data. Classification based on machine learning needs data to train a model

for actual application, which is one of critical problem of lane marks classification. In the field of lane marks detection, database with both annotation of marks and corresponding categories is not sufficient for training. Most of the datasets only provide representative lines of marks and are not easy for access[40]. In addition, since current methods extract partial patches of the marks, corresponding classification do not guarantee robust results.

Therefore, A solution of two main procedures is developed. First, construct a database which contains both lane mark representation and corresponding labels for classification. Second, Building a detector that could extract entire set of lane marks for further classification.

4.2 Database

4.2.1 Review

Similar to normal object recognition missions, there are multiple datasets for lane detection and classification. Such datasets capture images from vehicles with drivers' sight that suits the demand of auto-driving. These datasets differs in data scenarios and annotation methods. For example, KITTI dataset[21] records road segments and store the label as segmentation maps. Caltech datasets[40] for lane detection uses b-Spline to represent the lane marks and label the categories with different colors. Other datasets, such as ROMA dataset[41], SLD dataset, store the lane marks' representation as segmentation maps, where lane marks are segmented as white patches and other components as black.

These datasets, in general, act as basic test standards for different lane detection methods. However, these major datasets may show some disadvantages during using. Caltech dataset, although straightforward for testing, it does not provide convenient interfaces for access to the data labels. And the labeler is not easy to use. ROMA dataset, which provide detailed shape of lane marks, lacks information of connection among marks.

In other words, the dataset does not group the marks to show that which marks belong to the same lanes. Furthermore, these datasets have one feature in common is that in the images, few objects appears on the road. On the other hand, however, actual implementation of road detection methods always occur with complicated road conditions. Samples of simple road conditions may not contribute efficiently to the methods. Consequently, these dataset could not provide enough information or criterion for further research. Therefore, A road lane dataset that contains multiple kinds of information is required.

4.2.2 Singapore traffic road dataset

My research is associated with an industrial project. It contains multiple requirements, including scene understanding and object detection. In the part of scene understanding, one major mission is to recognize road lane and give each recognized lane mark a proper kind. The project instructors intend to use the machine learning to solve related problems. Besides, the project wants to use the data collected from Singapore for the research. Based on the condition above, the project team requested the related data from ST Kinetics (STK) and further process are completed.

The data from STK is a video captured from a MRT bus, second floor. the camera captures front view of the bus. The final image sequence extracted from the video is about 17744 frames in each patch, and 27 patches in total. In this video, the camera recorded the bus's routine of both directions. And the there are two different sights of the camera in this data, the near view and the far view. To make the data more efficient to use, the project team chose the far view of the video, which contains more objects and roads regions. Besides only one direction of the bus routine is used to avoid scene similarity. Consequently, the data consists of 7 patches of image sequences with 17744 frames each and 1 patch with 1800 frames. In this set of data, the scenarios concludes urban, uptown, and wilds, which covers almost all the scenes that may found in Singapore. Some examples of the dataset is shown in Fig. 4.1.

Figure 4.1: Example of the Singapore traffic dataset

The major focus of the research is recognizing lane marks on the road. In the dataset, there are several marks appearing on the road, namely guiding marks: double solid lanes, solid lanes, dashed lanes, and zig-zags; ground signs including direction arrows, speed limitation signs, forbidden districts and etc.. In the solution, the main focus is the guiding marks.

Figure 4.2: Examples of the annotation

To record and label these marks, the annotation method should both consider covering the marks and recording the relation among marks. For example, in the condition of double solid lanes, the labeling procedure should consider covering two individual lane marks and annotating them into single label. Therefore, a method for the annotation is developed. In the method, an open-source annotation tool, Labelme, is used. It draws a polygon which contains entire object and give the polygon a label, namely the label of the object. For the lane marks annotation, such polygons on the related lane marks is constructed, where polygons' edges are close to the marks' edges, only narrow spaces appear between the two, and the shape of the polygons are similar to the corresponding lane marks. The example of the annotation is shown on Fig. 4.2.

Our method considered both shapes and connections of the marks , which improved the problems mentioned above. The annotations record contours of the polygons, which could further provide information for curve fitting. The basic annotations of the lane marks provide shape and region information for the further works, but it lacks representation of lane marks' structure. Researches have shown concentration on what kind of lines that the lane marks constructs. By acquiring the related information, the lane marks are expected to form lines that properly describe the directions and structures. Since the annotations provide contours of each polygon, it helps the construction of these representation lines. In the implementation,The Least Squares method is utilized to build the

Figure 4.3: Examples of the representing curves on the dataset. Red: Road edge; Yellow: Double solid lane; Blue: Dashed lane; Green: Solid lane.

fitted curves. The lines do not cross the contours of the annotations, but make sure it minimize the sum of distance to all the contours. Usually, such lines have determined formula, namely they are of straight lines or quadratic functions. In this dataset, the lane marks are basically classified into two kinds: straight lanes and curves. Obviously, different fitting models are needed for each kind. For straight lines, linear functions is used for fitting, and for curves, quadratic functions is used. Some examples are shown in Fig. 4.3. We can see that the fitted lines properly represent the annotations. For each lines, their functions' parameters and some key points are recorded.

Based on the annotation and further processes, certain algorithms are expected to detect and classify the labeled marks.

Category	Count	Proportion
Solid lane	1159	11.81%
Double solid lane	881	8.98%
Dashed lane	4689	47.78%
Zig-zag	228	2.32%
Road edge	2857	29.11%

Table 4.1: Categories of road marks and corresponding stastatics

4.3 Preprocessing

4.3.1 Problem formulation

To deal with road lane detection problems, An important factor is extracting proper features of the marks for further detection and classification. Different from normal objects, road lane marks have following important properties:

- **Simple shapes**

  It is obvious that most of the lane marks have unique colors and shapes. For some marks that have special functions, their color may differ from the normal one. Regardless of damages or reconstruction, a whole piece of lane mark owns only one color. Besides, their shapes are always simple. Most of the guiding marks are of regular lines (in the view of actual design) with fixed width.

- **Significant Contrasts**

  Lane marks are always designed to be very distinctive against road surface. When focusing on the roads, one can easily find edges of the marks. In the view of computer vision, under normal scenes, it is convenient to compute gradients of the marks edges, which provides a possible way of extracting marks regions.

- **Complicated environments**

  Although lane marks are simple in shape and color, these properties do not ease related detection and classification. On the contrary, their simplicity stresses the problem solving. The marks' surroundings, including the road surface, vehicles, road edges, parallel lines of different types and even illuminations could provide huge information which could damage the quality of lane marks features. In some cases, these factors could directly change the road marks' properties of color and shapes in the view of computer vision. For example, strong illumination could make the road surface bright enough to make the marks not significant as normal conditions, which make the marks hard to find even using humans' eyes. On the other hand, when occurring shadows, lane marks' vision properties are also

changed: their contrast against road surfaces are lessened, their shapes are not significant or regular, as their brightness may decreases significantly when covered by strong shadows.

Based on these conditions, lane marks detection and classification requires unique feature extraction and processing methods that are different from typical object detection.

4.3.2 Non-linear diffusion

To emphasize the structural information of lane marks, the method should focus on extracting significant and unique features of the marks. As discussed above, edge and shape features are most remarkable features of the lane marks. Finding a proper way for the extraction will enhance the further process.

Methods for edge detection are wildly developed and implemented, most of which uses gradient information and combines with other pre-processes. one of the reasons for using gradient maps is that the edges are sharp enough to distinct, and by applying gradient kernel, one can obtain the information of the edges' magnitude as well as orientation. Such features are useful for fitting representation lines of the marks. However, in normal cases, the gradient are not easy to obtain because of the surrounding texture. Road surface may produce noises while computing the gradient, which will make expected edges sink into the surrounding. Removing the noise could ease the problem but damage the quality of edges.

In order to get clear edge efficiently for further gradient computing, some pre-processing methods are needed. In this solution, non-linear diffusion is used for the image enhancement. The pre-processing expects smooth surface and distinctive edges of lane marks. Normal linear diffusion, for example, Gaussian filter for smoothing, tends to smooth both noise and edges, and it also dislocate the edges when one moves from a finer to a coarser scale. Thus a nonlinear method for noise removal is introduced.

The nonlinear diffusion is originated by Perona and Malik[42] who called their filtering

method anisotropic diffusion.In their solution, the following equation is used:

$$g(|\nabla u|^2) = \frac{1}{1 + \frac{|\nabla u|^2}{\lambda^2}} \tag{4.1}$$

where u denotes the input image, and λ denotes the threshold. If $|\nabla u|$ is far larger than λ, Then the function $g(|\triangle u|^2) \rightarrow 0$,which means if the divergence is large enough, then the filter will have little influence on the image region. On the other hand, if the divergence is very small, $|\triangle u| \rightarrow 0$, $g(|\nabla u|^2) \rightarrow 1$, indicating that the region with small divergence will be significantly modified by the filter.

The result of using the filter is effective. In the case of image edge detection, the edges are clearly preserved while some texture features are blurred. Therefore, this method is expected to enhance the images to make road marks clearer and swipe other noises.

4.4 Road mark patches extraction

4.4.1 Edge extraction

As discussed above, the edges are significant for detecting road marks. Compared to other regions' texture, edges of road marks indicates big changes of the value on the neighbor region. Intuitively, applying gradient to the image could draw a clear map of the edges. However, after some pre-processing methods, for example non-linear diffusion, some noises still exist. To further enhance the quality of edge map, a different method is used.

Below is 1-dimensional Gaussian filter. Gaussian filter, as a normal filter for images, are wildly used in image blurring. One main implementation of Gaussian filter is Gaussian low-pass filter. Mathematically, a Gaussian filter modifies the input signal by convolution with a Gaussian function; this transformation is also known as the Weierstrass

transform.

$$g(x) = \sqrt{\frac{a}{\pi}} \cdot e^{-ax^2} \tag{4.2}$$

The frequency response is given by Fourier transform:

$$\hat{g}(f) = \exp(-\frac{\pi^2 f^2}{a}) \tag{4.3}$$

where f is the ordinary frequency.

The filter preserves image components of low frequency and reduces or even suppresses components of high frequency. In the view of image frequency, high frequency region or components indicates that such regions contain sharp changes in texture or color, and on the other hands, low frequency regions represent smooth regions or textures with slight changes. By using low-pass filters, image components of slight changes are preserved and components with sharp changes are reduced or blurred.

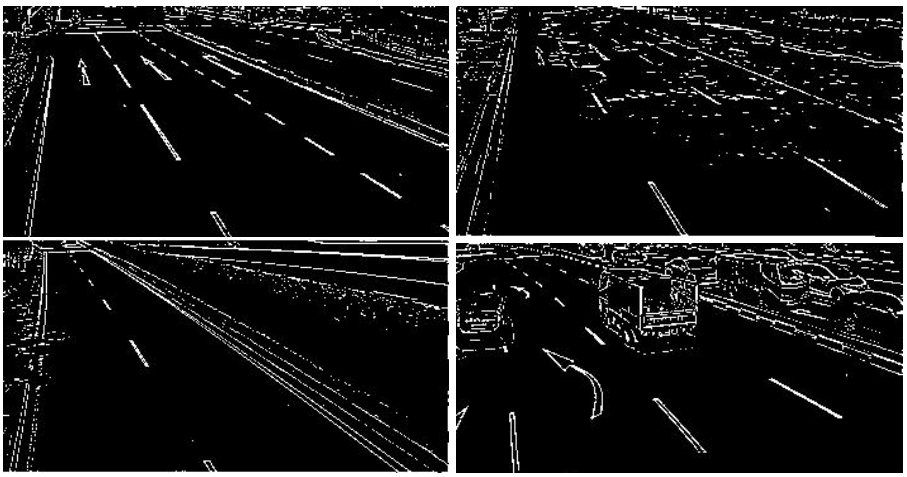

Figure 4.4: Example of high-pass filter processing.

In the context of road marks detection, the main purpose is to preserve the edges, namely some regions with significant changes. Therefore, the property of low-pass filters is utilized for edge extraction. The low-pass filters preserve smooth or stable regions, based on which desired regions can be extracted by simply eliminating the stable regions. As a result, the regions of edges and other sharply changing textures are preserved. Obtained

from the figures, the lane marks regions' edges, and other textures or components are suppressed. Which is ideal for further mark extraction.

4.4.2 Line fitting using RANSAC

After using high-pass filters and related methods mentioned in the pre-processing part, the input images are significantly simplified and some valuable information is exposed. However, there are still some regions that contain features points after the pre-processing. Therefore, proper method is needed to locate the position of the land marks and extract the marks into individual patches.

In this dataset, almost all the edges are of straight lines or slightly curved in this dataset's camera view. In the view of feature maps, such lines are ideally extracted and distinctive against other feature points and regions. Based on the maps, further works are needed for marking these edges to make road marks preserved and other regions are completely suppressed.

There are several ways for the method focusing only on these edges. For example, hough transformation could quickly extract potential straight lines; Line segment detector (LSD) provides a quick way of marking straight but short lines which could be further processed into desired lines shapes. In this method, however, these methods have difficulty in finding the road marks' edges, besides, they tend to extract the lines from other regions which are not expected. In the context of this feature maps, lines are expected to approximately fit the direction and shape of the road marks. Namely, if a line that is properly built based on feature points on the edges, it will ideally represent the corresponding marks. Therefore, a method called RANdom SAmple Consensus (RANSAC)[43] to handle the line fitting missions.

As a line fitting algorithm, RANSAC does not directly collect all the possible points for further fitting, it is done by a iterative routine for estimating the parameters of a specific model by picking a subset of data source in each iteration step. In the data source, there are inliers and outliers, The inliers are the elements that will influence the parameter

updating while outliers do not make such influence. Therefore, it can be interpreted as an outlier detection method. The algorithm was first published by Fischler and Bolloes in 1981.

In the view of RANSAC, the data source can be represented by certain model with adjustable parameters. In this model, the outliers are not fitted. If some inliers are given or found, RANSAC will make a procedure to adjust the model parameter to make the inliers have a optimized model as representation.

RANSAC focuses on estimate parameters of a model by randomly sampling data input. A simple interpretation of RANSAC is that, it sets a voting procedure for a model. If an element suits the model, then it is concluded, otherwise it will be recognized as noise and then discarded. The RANSAC follows two steps that are iteratively executed:

(1) Randomly select a subset of data from data source and fit it into a certain model to be generated. Then corresponding computation is performed on the data chosen for the parameter adjust;

(2) RANSAC checks the entire data source if there are elements consistent with the model and corresponding parameters generated in the first step. An element will be recognized as outlier if it does not fit the model or the result of parameter which is computed with the element is beyond certain threshold which defines the border of offset on the model.

The set of inliers acquired for the fitting model is called consensus set. The algorithm will be executed iteratively until the consensus set has enough elements according to the setting before execution.

The input of RANSAC concludes data elements, a model to fit and corresponding parameters to compute. RANSAC achieves the model updating and data input selection with following steps:

(1) Randomly select elements from data source as inlier potentials;

(2) The input model is computed with the inlier potentials;

(3) All other data in the data source is tested in the model, if the data element suits the model, which means the parameter within the model is not changed over threshold, then the element is contained in the consensus set;

(4) If the consensus set contain enough number of elements, the model is recognized as robust model.

(5) The model can be improved with all the consensus set candidates.

A brief conclusion of RANSAC algorithm is shown in algorithm 5:

Algorithm 5 RANSAC

Input: The lane feature map **x**

Initialize the score of fitted lines as s, a maximum iteration time n, and a line storage l

for $i \leftarrow 1$ to n **do**

Get random sample points x as input points;

Use these input points to build a initial line l_0

Compute the score s' of the line using certain method.

if $s' > s$ **then**

update s with s' and l with l_0

end if

end for

Output: The fitted line l

An advantage of RANSAC is the ability to do robust estimation of the model parameters. By iteratively picking elements and fitting models, the algorithm could gradually find the ideal model that best describe the desired distribution. On the other hand, some disadvantages also exist. As the algorithm follows the procedure of iteration, it may require a number of iteration due to the difficulty of model fitting. Thus the implementation time could be long. Besides, one instance of RANSAC algorithm can only estimate one kind of model. It is not capable of multiple model fitting. This method only focuses on straight lines, thus such problem will not occur.

As an actual implementation, this method that fits the road lane marks utilizes RANSAC, in which the extracted edge points and other noises are used for fitting the model. The model expected here is the straight lines that best describe lane marks' shape and direction. Traditionally, RANSAC can only fit one model in each execution. In the normal scenes there are more than one lane marks need extraction. Therefore, a procedure is designed for the whole image's line fitting:

(1) Initialize the algorithm instance with one input feature map;

(2) Use RANSAC with certain number of iteration to estimate the first line marks, build and save the fitted lines;

(3) Delete the points in the feature map that are used for the fitting in the step above. The new feature map is then processed with RANSAC for the next lane mark estimation;

(4) Repeat step 3 until the number of points that are used for fitting an estimated line is less than 10;

(5) Calculate the slope of each lines, if the value is outside a threshold which is set as $[0.4, 4)$ and $[-0.5, 0)$ in this scenario, the discard the corresponding line;

(6) End of the procedure.

Some results for the line fitting are shown in Fig. 4.5.

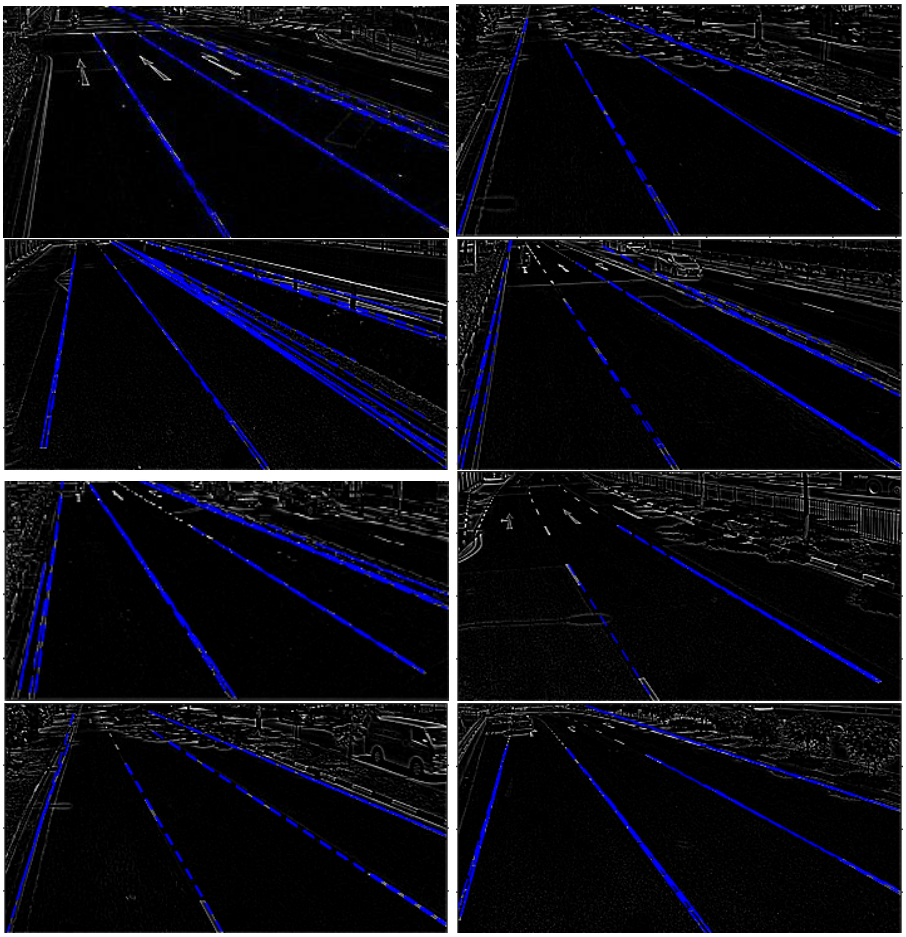

Figure 4.5: An example of RANSAC fitting results

4.4.3 lane mark patches extraction

Based on the fitting result with RANSAC, the solution could continue on extracting the patches of lane marks. By reviewing the lines' condition on the feature map, a fact can be obtained that the lines are located inside the marks or just around the edges. Therefore, based on these extracted lines, some patches are generated for segmenting the marks.

One of the dataset's features is that the images are captured from the same camera view. This property brings advantages for image regions extraction. Since the camera is fixed, the region of road and corresponding components are stable in its shape and location. Therefore, the the road marks from different images share same coordinate

system, which means an extraction method is implemented with same parameters on different images, the result could be equally reasonable. Specifically, lane marks are similar in size and shape among different images, which means one detector can grab these marks without any adaptive modification. The detailed procedure is as following:

(1) Input the lines found in the RANSAC step;

(2) Read the location of each line, as well as begin and end points' locations;

(3) For each line, set up an offset of fixed value δ. Then move the line by the value along x axis both to the left and right. As a consequence, each line has two copies at its left and right, each of which has distance δ to the original fitted line;

(4) For each line, record the contours of its two copies and crop the image according to these contours. The cropped image is the lane mark on the road;

As discussed above, since the view is fixed, the offset δ are not needed changing for each image. Some sample of the extraction is shown in Fig. 4.6.

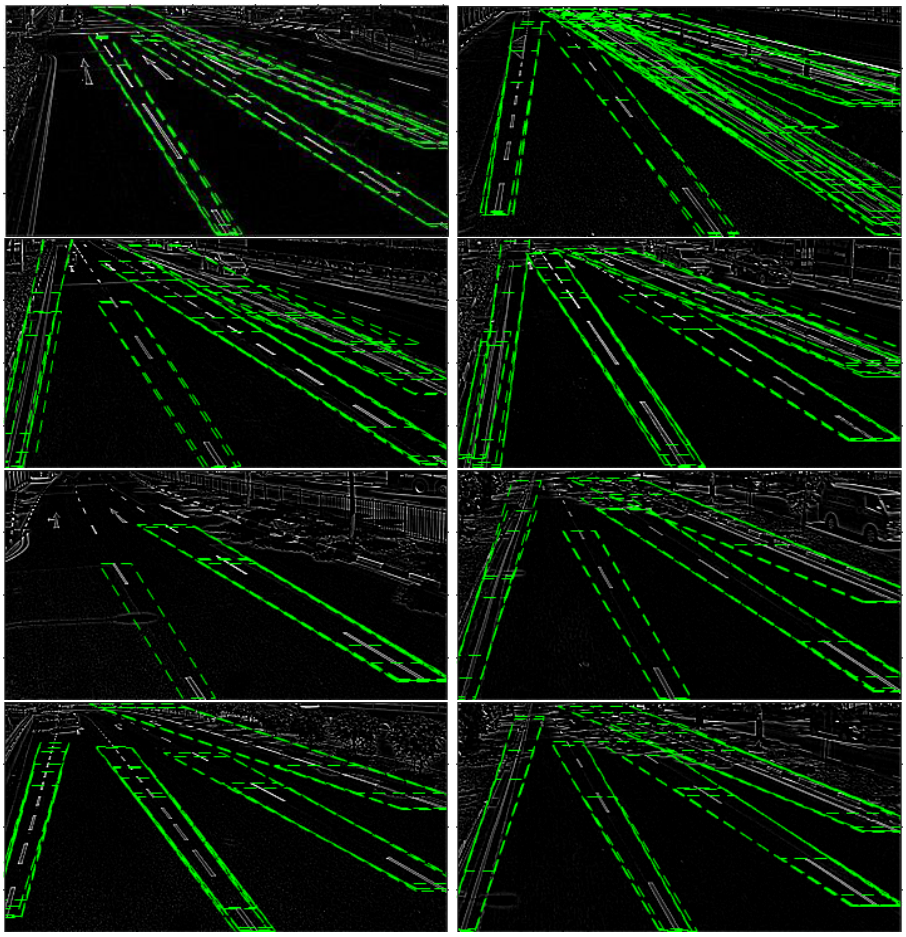

Figure 4.6: Some examples of marks patches segmentation maps

Based on the segmentation, patches of the lane marks are extracted. The format and structure of the patches are similar to the ones of the dataset mention in the former section.

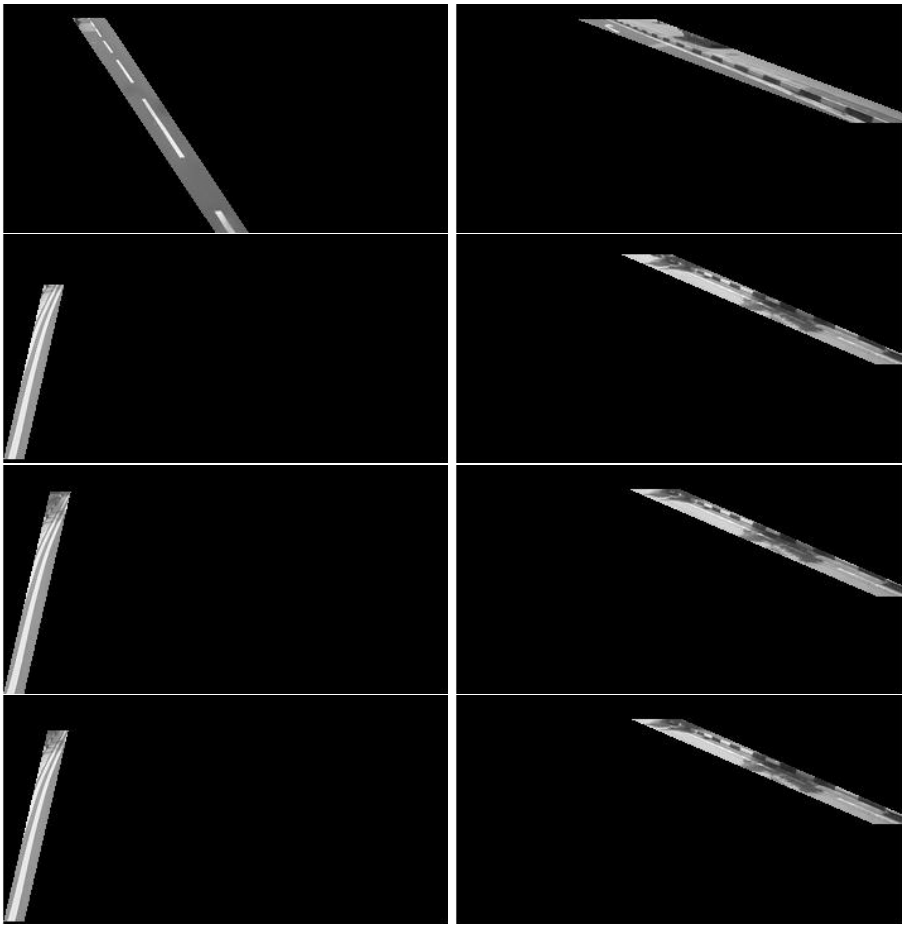

Figure 4.7: Some examples of marks patches

4.4.4 Implementation

In the actual experiment on the dataset, the annotated patches are extracted individually and marked with corresponding label. These patches' features are described using histograms of oriented gradients HOG[44] features. The classification are executed using both regularized ELM and standard support vector machine (SVM). For the training process, 8100 samples of 5 categories and 912 samples are prepared for test.

Method	Overall accuracy
ELM	93.3%
SVM	70.2%

Table 4.2: Overall accuracy on classification of the dataset

The detection method mentioned in the previous sections are combined and implemented on the Singapore traffic dataset. The test image contain normal scenarios which do not have irregular shadows on the road surface, and images of rough situations, for example road with cars or covered with shadows. The test is to evaluate the robustness of the detector in multiple conditions. The result is shown in Fig.Fig. 4.8.

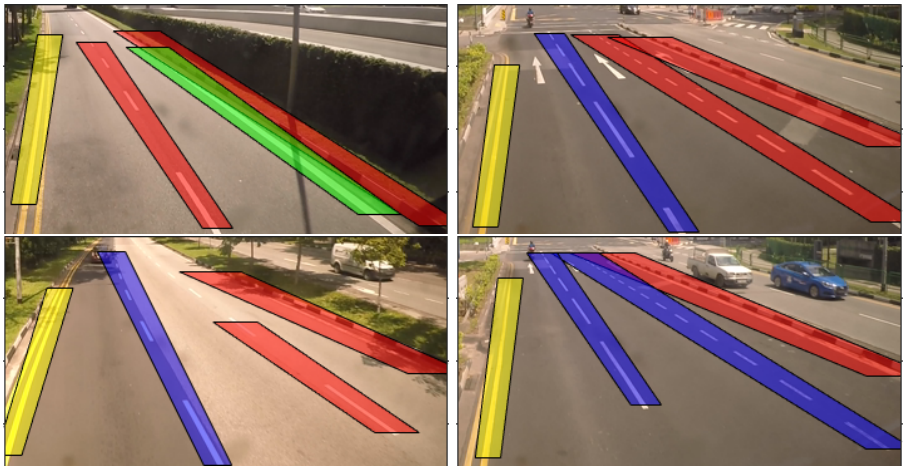

Figure 4.8: Examples of detection. Red: Road edge; Yellow: Double solid lane; Blue: Dashed lane; Green: Solid lane.

4.5 Performance Analysis

According to the methods introduced above, some experiments are performed and the results have been made. In the result, most of the lane marks are correctly detected and classified.

Method	Road edge	Single solid	Dashed	Double solid	Zig-zag
Road edge	99.32%	0	0%	0.68%	0
Single solid	NaN	NaN	NaN	NaN	NaN
Dashed	0.46%	0.23%	99.3%	0	0
Double solid	1.12%	0%	0	98.88%	0
Zig-zag	0	0	0	0.93%	99.07%

Table 4.3: Accuracy on each category in the dataset. (This test set does not contain single solid lanes)

Method	Road edge	Single solid	Dashed	Double solid	Zig-zag
Road edge	100%	0	0%	0	0
Single solid	6.19%	88.57%	5.24%	0	0
Dashed	8.20%	0.35%	89.88%	1.40%	0.17%
Double solid	0	0	0	100%	0
Zig-zag	NaN	NaN	NaN	NaN	NaN

Table 4.4: Accuracy on each category in the dataset. (This test set does not contain Zig-zags)

In the experiment, the test scenarios differ in road surface situations. In normal scenario where road surface is not covered with shadow from other objects or filled with vehicles, the detection result shows good detection effect (shown in left-top in Fig 4.8). In some situations as shadow covers the lane marks (shown in bottom left in Fig.4.8, where the road edges and double solid lanes are shadowed), The method preserve good performance on recognizing the lane marks. As introduced in the previous sections, fitting lines for representing the lane marks is the key to following steps. By adjusting related parameters of RANSAC model, the solution could fit desired lines in spite of the noises. Obviously, the edge maps may contain sample points from other objects, for example, vehicles and shadows. However, such sample points have little chance to form a line that can be fitted by RANSAC. Therefore the line fitting step can hardly be interrupted by the noise and the fitting results are robust for further processes.

The classification result on each category is shown on table 4.3 and table 4.4, indicating the accuracy of classifying each category and the error on mis-classification. The results show that most samples are correctly classified, but some samples of certain categories have obvious tendency of being mis-classified. The first is the samples of dashed lane marks. From the table there are 8.20% of dashed marks classified as road edges, and 6.15% of single solid lanes are classified as road edges, 5.24% as dashed marks. The reason for this mis-classification may comes from the method of feature

extraction. The classifier receives HOG features as input. The HOG feature is considered as description of image structure. One common feature of these three categories is that they are all single-lined mark representations, which means that the feature samples from both sides of the marks take major partition of the whole feature maps, while the ones of detailed structures take minority of the whole features. Therefore features of the samples from the three categories share feature samples of fairly high similarity. Although the structure of these samples are distinctive, if the road surface condition is not ideal, for example the illumination is too high to distinguish detailed structures of the marks, the corresponding feature samples are hard to acquire. In addition, HOG features does not store color or texture information, which in turn discards an significant difference between dashed marks and road edges. Consequently the mis-classification occurs. Based on these reasons, there are some incorrect classification appearing on the detection results. In the part of detection, the results face the different situations. In normal conditions, road marks are clearly detected and classified. In the situations which are rough or complexed, some errors may occur that some marks may not be detected correctly. For example, if illumination are very low, the detector may not clearly filter the edges which leads to failure of corresponding marks detection.

The method for detector is designed specifically for recognizing the marks. It approximately depicts the position and shape of the lane marks, shaping a general structure of current road. However, the detector faces several disadvantages. First, the segmentation patches do not precisely shape the marks' structure, in which every patch are segmented as straight lines. When curve marks are detected, the segmentations cannot show the bending. Second, since the classification solution only sets the labels on categories of road marks, it does not build a criterion of discriminating straight lanes and curved lanes. Therefore, the detector is not capable of discriminating the specific direction of the road.

4.6 Potential Improvement

4.6.1 Adaptive detection region

The methods mentioned in the literature review all have a region limiting step before actual detection. The step has an advantage of erasing invalid region and preserve road regions for the ease of feature extraction, since the texture and color of road lane marks are possibly too similar to distinguish. Limiting detection region could considerably reduce the occurrence of wrong detection.

Previous methods use rectangle regions for limitation, which is fast to implement but not robust. The rectangle regions are mostly fixed in the input image. When the road surface changes its location in the image, for example, as the vehicle is turning or leaving main road for branches, the fixed region may not contain road surface, thus the lane marks are unable to be detected.

In this solution, the semantic segmentation, or scene recognition methods are considered as a solution for determining the regions. Based on the solution presented in Chapter 3, multiple approaches are available for the mission. An advantage of the solution is that, it can recognize road surface region of arbitrary location in the input images, which benefits the processing of lane marks locating.

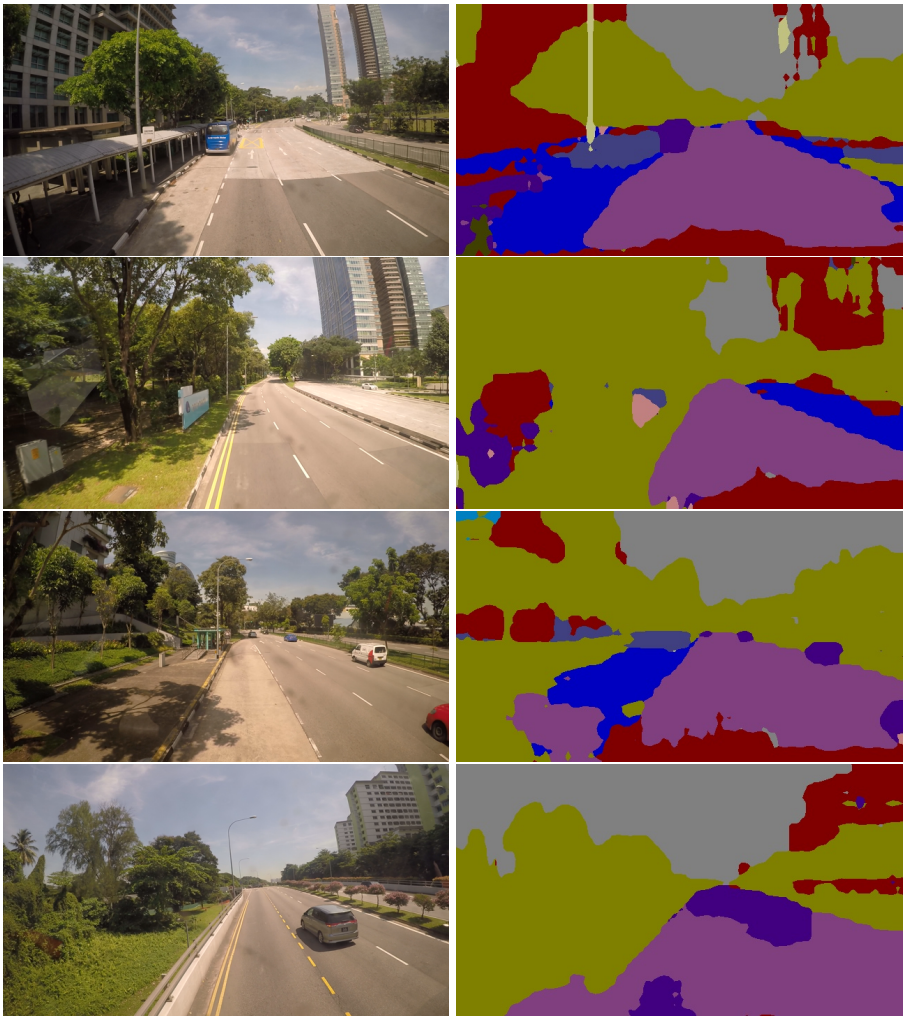

Figure 4.9: Examples of the road surface segmentation. Dark pink: Road; blue: Sidewalk, red: building; purple: vehicles; gray: sky; yellow: plantations

As mentioned in Chapter 3, FCN is designed for the segmentation problem and further optimization is needed. In this solution dilated convolution is considered as a proper optimization method. Some samples of road surface segmentation are shown in Fig.4.9. The results indicate that the segmentation method is capable of locating and shaping the road surface of multiple situations.

4.7 Conclusion

In this chapter, a solution of detecting road lane marks is presented. The solution contains data collection part and recognition part. The data is collected from Singapore traffic roads from different scenarios. Then the lane marks on the roads are extracted and labeled. In the recognition part, a RANSAC method is used for fitting lane marks representation lines, then the lane marks patches are extracted and classified. The results show good performance on recognizing each categories.

Chapter 5

Conclusion and Future Works

5.1 Scene recognition

In this report, the development of scene recognition is reviewed. From non-deep-learning method to deep learning method, the technique has grown into a field that both satisfies recognition accuracy and speed. Traditional methods are reviewed using two typical methods, Conditional Random Fields and auto-context. These methods still have their value for applications and new learning method development. Developed from traditional methods, the deep learning methods take advantages of modern computation devices and multiple kinds of networks are developed to complete scene recognition tasks. As a spatial dependency model, RNN and its integrations are reviewed. FCN, a fundamental framework for general semantic segmentation, is reviewed and a project is done based on its improvement. The report also reviewed corresponding dataset for scene recognition tasks, in which the mechanic of dataset is presented and some advantage and shortcomings are analyzed. Besides, the FCN framework is extended to perform multi-task processing. The method predict images' pixel-wise segmentation (scene recognition) and edge maps simultaneously. The main purpose of the method is to use information from edge map as structure information to enhance pixel-wise segmentation performance. This method needs further improvement for better edge de-

tection to enhance the whole network's performance on scene labeling.

Solving typical challenges in scene recognition

The scene recognition tasks still face challenges. As mentioned above, there are two main problems for general scene parsing, rare class recognition and similar classes confusion.

Rare classes recognition[45] is based on two phenomenons.First, a certain class rarely appears in the dataset. Second, some classes frequently appear in the images but occupy very small regions. Two factor make these classes hard to be segmented. In some tasks, these classes are very crucial for the whole image representation, or they are the elements that researchers concern. Therefore, an efficient method for rare classes recognition in semantic segmentation is required.

The difficulty of distinguishing similar image regions is that although given plenty data for training, the feature maps between these two kind of regions are still confusing for classifiers. For example, general FCN networks are difficult to distinguish walk streets and roads for vehicles, which is crucial for traffic vision applications.

One possible way is extract features with significant difference. Image regions with similar classes are globally similar, but different in detailed textures. The key to the problem is finding proper representations for them. In the view of CNNs, features that contain detailed texture information is mostly stored in low-level feature maps, which is close to image input. Using certain method to distinguish such slight difference may result in significant classification.

Current research reveals that constructing rich feature of multi-scale structure could solve the problem in some degree. The method extracts features with different kernel and pooling sizes, upsamples them and concatenates them into one final feature map. This method collects rich information, and stores them in a pyramid shape, which reflects image textures' features from global scale to tiny region scale. In the view of receptive fields, such network has stored as much context information as possible. Con-

sequently, the method archived 80% overall accuracy on PASCAL VOC dataset. The advantage of this method is that its rich information provides robust support for further computation. However, it occupies huge space, and if the computation device is not strong enough, it may not be so efficient as the computation procedure requires huge resources.

Therefore, a method that costs less computation resources as well as keeps ideal segmentation performance is needed.

5.2 Road lane marks detection

Based on the solution presented in this chapter, the basic structure of the road lane marks detector is completed. Each step of the solution is relatively independent and fully connected. To achieve the goal, high pass filter is used to extract the feature map that could clearly capture the road lane marks' edges. Then the RANSAC algorithm is used to unite this edges as representative line of each marks. Based on the fitted lines, marks patches are extracted for further classification. In the classification step, Regularized Extreme Learning Machines is utilized for quick classification.

Although the solution could approximately recognize marks, each of the steps are so basic that available for improving. First is the method of detector. The measurement indicates that the detection method is naive. The extracted patches are coarse, and the final detection is not precise. The main reason is the way of extracting fitted lines. When facing line fitting problem, RANSAC specifically solves straight line fitting problem, which means it is capable of fitting curves. The results show that some detected patches are short. This is because the result of line fitting does not cover the points from smaller points in the feature maps. Two factors contribute to the reason. One is the points that should be of one line is too far from the previous cluster that cannot be covered by RANSAC algorithm, and the other is that the number of points is small when the components of the feature map is far from the camera. Based on these factors, finding a way for modeling curves and lines of wider coverage based on current features maps is an

expecting direction of improvement. Second, in the processing of classification, a simple feature called HOG feature is used. The method is efficient for classification based on structure but weak in the one on colors. As discussed above, a significant difference between dashed lane marks and road edge marks is color or grayscale. Utilizing the difference of these features could potentially improve the classification performance. In addition, some common road lane marks also contain colors different from pure white, for example double solid lane marks. Using color information could improve the accuracy on classifying these marks.

Bibliography

[1] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *arXiv preprint arXiv:1606.00915*, 2016.

[2] Z. Tu and X. Bai, "Auto-context and its application to high-level vision tasks and 3d brain image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 10, pp. 1744–1757, 2010.

[3] Z. Zuo, B. Shuai, G. Wang, X. Liu, X. Wang, B. Wang, and Y. Chen, "Convolutional recurrent neural networks: Learning spatial dependencies for image representation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2015, pp. 18–26.

[4] B. Shuai, Z. Zuo, B. Wang, and G. Wang, "Dag-recurrent neural networks for scene labeling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3620–3629.

[5] 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*, 2015, pp. 3431–3440.

[6] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," *arXiv preprint arXiv:1612.01105*, 2016.

[7] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, "The role of context for object detection and semantic segmentation in the wild," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[8] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Semantic understanding of scenes through the ade20k dataset," *arXiv preprint arXiv:1608.05442*, 2016.

[9] T. M. Hoang, H. G. Hong, H. Vokhidov, and K. R. Park, "Road lane detection by discriminating dashed and solid road lanes using a visible light camera sensor," in *Sensors*, 2016.

[10] A. Parajuli, M. Celenk, and H. B. Riley, "Robust lane detection in shadows and low illumination conditions using local gradient features," *Open Journal of Applied Sciences*, vol. 3, no. 01, p. 68, 2013.

[11] J. Hur, S.-N. Kang, and S.-W. Seo, "Multi-lane detection in urban driving environments using conditional random fields," in *Intelligent Vehicles Symposium (IV), 2013 IEEE*. IEEE, 2013, pp. 1297–1302.

[12] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[13] V. Koltun, "Efficient inference in fully connected crfs with gaussian edge potentials," *Adv. Neural Inf. Process. Syst*, vol. 2, no. 3, p. 4, 2011.

[14] A. Wipf, *Mean Field Approximation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 119–148. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-33105-3_7

[15] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, "Conditional random fields as recurrent neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1529–1537.

[16] T. Mikolov, "Recurrent neural network based language model."

[17] 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*, 2016, pp. 2818–2826.

[18] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1915–1929, 2013.

[19] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li, "Imagenet large scale visual recognition challenge," *CoRR*, vol. abs/1409.0575, 2014. [Online]. Available: http://arxiv.org/abs/1409.0575

[20] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012.

[21] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[22] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ade20k dataset," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[24] 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 (NIPS)*, 2015.

[25] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Hypercolumns for object segmentation and fine-grained localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 447–456.

[26] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "Lsd: A fast line segment detector with a false detection control," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 4, pp. 722–732, 2010.

[27] C. Kreucher and S. Lakshmanan, "Lana: a lane extraction algorithm that uses frequency domain features," *IEEE Transactions on Robotics and automation*, vol. 15, no. 2, pp. 343–350, 1999.

[28] C. Kreucher, S. Lakshmanan, and K. Kluge, "A driver warning system based on the lois lane detection algorithm," in *Proceedings of IEEE International Conference on Intelligent Vehicles*, vol. 1. Stuttgart, Germany, 1998, pp. 17–22.

[29] X. Wang and M. Han, "Online sequential extreme learning machine with kernels for nonstationary time series prediction," *Neurocomputing*, vol. 145, pp. 90–97, 2014.

[30] G.-B. Huang and C.-K. Siew, "Extreme learning machine with randomly assigned rbf kernels," *International Journal of Information Technology*, vol. 11, no. 1, pp. 16–24, 2005.

[31] W. Deng, Q. Zheng, and L. Chen, "Regularized extreme learning machine," in *Computational Intelligence and Data Mining, 2009. CIDM'09. IEEE Symposium on*. IEEE, 2009, pp. 389–395.

[32] J. Dai, K. He, and J. Sun, "Instance-aware semantic segmentation via multi-task network cascades," *CoRR*, vol. abs/1512.04412, 2015. [Online]. Available: http://arxiv.org/abs/1512.04412

[33] R. Caruana, "Multitask learning," in *Learning to learn*. Springer, 1998, pp. 95–133.

[34] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[35] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[36] J.-J. Hwang and T.-L. Liu, "Contour detection using cost-sensitive convolutional neural networks," *arXiv preprint arXiv:1412.6857*, 2014.

[37] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang, "Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3982–3991.

[38] A. B. Hillel, R. Lerner, D. Levi, and G. Raz, "Recent progress in road and lane detection: a survey," *Machine vision and applications*, vol. 25, no. 3, pp. 727–745, 2014.

[39] M. B. de Paula and C. R. Jung, "Real-time detection and classification of road lane markings," in *Graphics, Patterns and Images (SIBGRAPI), 2013 26th SIBGRAPI-Conference on*. IEEE, 2013, pp. 83–90.

[40] J. Fritsch, T. Kuhnl, and A. Geiger, "A new performance measure and evaluation benchmark for road detection algorithms," in *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*. IEEE, 2013, pp. 1693–1700.

[41] T. Veit, J.-P. Tarel, P. Nicolle, and P. Charbonnier, "Evaluation of road marking feature extraction," in *Proceedings of 11th IEEE Conference on Intelli-*

gent Transportation Systems (ITSC'08), Beijing, China, 2008, pp. 174–181, http://perso.lcpc.fr/tarel.jean-philippe/publis/itsc08.html.

[42] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 12, no. 7, pp. 629–639, 1990.

[43] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," vol. 24, pp. 381–395, 06 1981.

[44] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.

[45] J. Yang, B. Price, S. Cohen, and M.-H. Yang, "Context driven scene parsing with attention to rare classes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3294–3301.