人工知能を用いた次世代IoTアプリケーションに関する研究

Artificial Intelligence for Emerging Internet-of-Things Applications

Liangzhi Li

Department of Information and Electronic Engineering
Muroran Institute of Technology

This dissertation is submitted for the degree of
Doctor of Philosophy of Engineering

March 2019

Declaration

I hereby declare that this thesis is my own work and effort and that it has not been submitted anywhere for any award. Wherever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature, and acknowledgement of collaborative research and discussions.

Liangzhi Li
March 2019

Acknowledgements

It seems like yesterday when I first step into the campus of Muroran Institute of Technology. Everything is beautiful. Even now, I can still hardly believe that I can study in such a picturesque place. This really has been one of the happiest time in my life. I appreciate all of you who made these years such a wonderful journey.

I am especially grateful to Prof. Mianxiong Dong, for putting a lot of effort into my research. You gave me numerous opportunities to realize my self-worth. I really appreciate it.

I have been privileged to have Prof. Kaoru Ota in the same laboratory. She was always generous with her time and energy to help her students.

I have learned so much from Prof. He Li. You have always been my teacher and my friend. Thank you for accompanying me, through all the happy times and all the hard times.

I acknowledge all the help from the thesis committee. Your valuable comments and suggestions are of great significance for me.

I owe a debt of gratitude to Xiaoyi, Chaofeng, Zhaola, Jianwen, Xiting, Zujun, Luyao, Axita, Songhe, Caijuan, Zhenzhen and all my friends and colleagues in Muroran IT. Thanks for everything you have done.

I would like to express my sincerest appreciation to my parents for their invariable understanding and support. They have tried their best to give me everything I need and have sacrificed so much in this process.

I would also like to thank my exceedingly patient and supportive wife, Zhaofeng, who is by my side right now, watching me writing this document. For you, I have countless words treasured in my heart.

I dedicate this dissertation to all of you. Thank you.

Liangzhi Li
March 2019

Abstract

There have been more and more Internet-of-Things (IoT) applications emerged in the recent few years, e.g., connected robots, Internet-of-Vehicles, IoT-enabled Smart Grid, IoT-based Sensing, etc. These new-generation IoT applications are potential to totally revolutionize people's everyday life. However, before that, a series of new artificial intelligence (AI) methods are needed to efficiently analyze the big, heterogeneous IoT data, and automatically give reliable decisions, accurate predictions, or quick yet correct feedbacks. In the dissertation, the author focused on the design and implementation of some novel deep learning approaches, in order to address various challenging problems in several emerging IoT applications.

First, the author proposes a view-invariant Convolutional Neural Network (CNN) Model for the scene understanding tasks of connected disaster-handling robots. In this system, two individual CNNs are used to, respectively, propose objects from input data and classify their categories. The author attempts to overcome the difficulties and restrictions caused by disasters using several specially-designed multi-task loss functions. The most significant advantage in this work is that the proposed method can learn a view-invariant feature with no requirement on RGB data, which is essential for harsh, disordered and changeable environments.

Second, the author adopts AI methods to implement intelligent decision-making for autonomous vehicles. A human-like driving system is proposed to give autonomous vehicles the ability to make decisions like a human. In this method, a decision-making system calculates the specific commands to control the vehicles based on the abstractions. The biggest advantage of this work is that the author implements a decision-making system which can well adapt to real-life road conditions, in which a massive number of human drivers exist.

Third, the author focuses on the electrical load forecasting task for the Internet-of-Energy. An IoT-based deep learning system is introduced to automatically extract features from the captured data, and ultimately, give an accurate estimation of future load value. One significant advantage of this method is the specially designed two-step forecasting scheme, which significantly improves the forecasting precision. Also, the proposed method is able to quantitatively analyze the influences of some major factors.

At last, the author introduces a new AI method to address the efficiency and privacy problems in the Internet-of-Sensors. The author adopts the state-of-the-art edge computing method to solve the crowdsensing problem with the real-time sensing data. A distributed deep learning model is adopted to extract features from the captured data, which is not only a compression process to reduce the communication cost, but an encryption procedure for privacy protection.

Table of contents

List of figures

List of tables

Nomenclature

Roman Symbols

$\mathscr{A}$	An input abstraction set
$\mathscr{B}$	Biases of proposed network, $\mathscr{B} = \{B_1, B_2, ..., B_\eta\}$
F	The repulsive force
L	The loss functions of deep models
$\mathscr{P}$	Set of ground-truth speed labels, $\mathscr{P} = \{p_{x_1}, ..., p_{x_n}\}$
p	Parameter of Minkowski distance
$\mathscr{T}$	Set of ground-truth 3D-box labels $\mathscr{T} = \{t_{x_1}, t_{x_2}, ..., t_{x_n}\}$
$\mathscr{T}$	Set of ground-truth steering labels, $\mathscr{T} = \{t_{x_1}, ..., t_{x_n}\}$
U	The repulsive potential
$\mathscr{W}$	Weights of proposed network, $\mathscr{W} = \{W_1, W_2, ..., W_\eta\}$
$\mathscr{X}$	Set of Input data
$\mathscr{Y}$	Set of ground-truth category labels, $\mathscr{Y} = \{y_{x_1}, ..., y_{x_n}\}$

Greek Symbols

η	Positive scaling factor
λ	Term weights used in functions
θ	A layer of the deep learning network

Subscripts

i subscript index

j subscript index

k subscript index

Other Symbols

B_θ Bias matrix of layer θ, $B_\theta = (b_\theta^1, b_\theta^2, ..., b_\theta^m)$

c_θ Neuron number of layer θ

$h_\theta(x_i)$ Output of layer θ

$J(\mathscr{W}, \mathscr{B})$ Loss function of proposed network

W_θ Weight matrix of layer θ, $W_\theta = (w_\theta^1, w_\theta^2, ..., w_\theta^m)$

Acronyms / Abbreviations

5G 5th Generation wireless systems

AI Artificial Intelligence

AMI Advanced Metering Infrastructure

CNN Convolutional Neural Network

cvRMSD coefficient of variation of the Root Mean Square Deviation

DCEN Daily Consumption Estimation Network

DMN Decision-Making Network

DPA Direct Perception Approach

ECD Edge Computing based Deep model

FoV Field of View

GPU Graphics Processing Unit

HW Holt-Winters method

ILFN Intra-day Load Forecasting Network

IoT Internet of Things

LiDAR Light Detection And Ranging

MAPE Mean Absolute Percentage Error

NBV Next-Best-View

ODLN Object Detection and Localization Network

PCL Point Cloud Library

PLC Power Line Communication

PMU Phasor Measurement Unit

RCUN Toad Condition Understanding Network

RGB A color model where the Red, Green, and Blue light are combined to represent a broad range of colors.

RLM Reinforcement Learning Method

RMSD Root Mean Square Deviation

ROS Robot Operating System

RPF Repulsive Potential Field

SGD Stochastic Gradient Descent

SIC Scale-Insensitive Convolutional

TCRE Total Consumption Relative Error

TORCS The Open Racing Car Simulator

VI View-Invariant

VTCN View-invariant Three-dimensional Classification network

WSN Wireless Sensor Network

Chapter 1

Introduction

1.1 Background and Motivation

1.1.1 Internet-of-Things (IoT)

IoT, just as its name implies, means the internet of all things, including refrigerators, bulbs, air-conditioners, vending machines, and some emerging concepts, such as robots, vehicles, smart grid, smart sensors, etc. All of them can be connected to and organized as a complicated yet efficient network, where the data are shared among the participants. According to some statistic results[55], currently, there are 7 billion connected IoT devices in the world, and the number is predicted to be almost 21.5 billion by 2025.

Connection brings more captured data and more automatic control. If we can fully utilize this trend for future, IoT can bring a revolutionary change to our everyday life. However, the mapping from various, heterogeneous, and extremely complicated data to specific control commands is hard to clarify and need lots of research efforts. This is especially challenging in the newly-emerged areas, e.g., Internet-of-Robots, Internet-of-Vehicles, Internet-of-Energy, and Internet-of-Sensors, as they bring about much more complex data than ever before. The existing automatic analysis methods, such as artificial intelligence tools, are not yet ready for these new problems. The author focuses on the artificial intelligence area and attempts to use deep learning methods to address a series of problems in the emerging IoT applications, which will be detailed in this dissertation.

1.1.2 Artificial Intelligence (AI) and Deep Learning

One huge challenge in AI is that, the output result can be influenced by numerous factors. As we know, a simple three-layered neural network with only one hidden layer can approximate

any continuous nonlinear functions with any precision. Therefore, a deep neural network having multiple hidden layers has a strong ability in modeling the load forecasting problem under complicated conditions. Deep learning [27] is one of the deep neural networks. It belongs to unsupervised learning methods, which are able to determine which features are essential and which features can be omitted without manpower. As a result, it has a satisfactory robustness and an excellent generalization ability. Layer-by-layer data mapping and feature extraction are adopted in deep learning methods to improve the precision of forecasting and classification, which brings huge success in various research fields, e.g. pattern recognition, classification, clustering, dimensionality reduction, forecasting, recommendation, and information retrieval [43].

Traditional neural networks learn new patterns by calculating and tuning the weights. The algorithms for adjusting weights in shallow networks is usually very simple and limited in learning ability. Hinton and David Rumelhart proposed the back-propagation algorithm, adding a hidden layer in neural networks, which addressed the representation problem of perceptron and achieved much better performance. However, with the increase of network layers, deep networks frequently encounter overfitting problems, which result in poor generalization ability and bad performance, even compared with some shallow networks. Moreover, the vanishing gradient is also a severe problem in deep networks [17]. When sigmoid is used as the activation function, the back-propagation gradient decays to one quarter of original value with every propagation between two layers. Therefore, the weights near input end receive little back-propagated training signals. Neural networks with multiple hidden-layers are easily trapped in local optima, instead of global optima. And with the increase of network layers, non-convex objective function become progressively complex and local optima increase exponentially, which makes it extremely difficult to train deep networks [17].

To address the training problem of deep networks, Hinton proposed the strategy to pretrain the network layer-by-layer in advance and conduct fine-tuning to the existing weights [27]. But this method cannot fundamentally solve the training problem of deep networks. The main reason is, when the sigmoid activation function is close to saturation region, the function changes too slowly and its derivative approaches zero, which results in the vanishing gradient. In 2010, Glorot Bengio used ReLU function to train a neural network, and found it can not only reach a lower error rate, but have a better sparsity [21]. This finding meets the rules of biological brain that there are only one to four percent of total neutrons activated at a time [44]. To solve the overfitting problem, Hinton proposed a novel training strategy calling "dropout" in 2012 [28]. Its main principle is, in each iteration, each neutron is forbided to perform the forward and backward calculation with a certain probability p. So the network structure changes in every iteration of the training process, which brings a better robustness.

Deep networks can, to a great extent, avoid overfitting with "dropout". The incidence of some unexpected situations that the final output changes significantly with a small fluctuation of the input data, which is common for the traditional training method, decreases greatly. In addition, ReLU also zeroes the outputs of several neurons, which results in sparse networks, and, as a result, decreases the interdependence between parameters and reduces the incidence of overfitting.

With the further research in the pre-process of the raw data and several other novel approaches, such as ReLU and Dropout, both the vanishing gradient problem and the local optima problem can be effectively avoided [17]. Furthermore, the Graphic Processing Units (GPU) which have powerful parallel computing ability are widely used in network training, as a result, even a network with huge amount of data can get trained in an acceptable time.

As a result, deep learning becomes one of the most effective and efficient AI tools for various tasks, due to its strong feature extraction and analysis ability, especially for extremely-huge data. In this research, the author exploits a series of different designs and implementations of deep learning models, in order to adapt the intelligent IoT systems to various areas, which be introduced below.

1.2 AI for Emerging IoT Applications

1.2.1 AI for Robotic Vision

As an emerging area and a new IoT application, connected robots have became a great substitute for human in a lot of tasks. However, robotic will still face huge problems without good enough visual abilities. In this research, the author aims to enhance their powers in vision using AI methods. This work is focused on the disaster-handling robots as they are in most dangerous and complicated environments, and need much more efforts in their vision module.

Disaster is a persistent challenge to the human world. In recent years, lots of new technologies have emerged to improve our responses to various disasters. Although many encouraging researches have been made, one grave problem remains unsolved, i.e., how to safely explore damaged architectures or other dangerous areas. In fact, there are so many tragedies caused by secondary disasters that people cannot overlook this problem any more. Robotic, which is a great substitute for humans to carry out dangerous tasks, becomes the first choice in these scenarios.

Since robots require a sufficient understanding of the surrounding environment before any movements, scene understanding ability is of great importance for robots to be competent

for these tasks. However, traditional 2D methods struggle in such situations, because they give little space information, which is very important for robots to move around and interact with the world. Therefore, 3D perception will become a vital topic in robotic for disaster management. While many approaches have been proposed in the field of 3D object detection and recognition, all these researches cannot be applied to the disaster-handling robots directly, due to some limitations existing in such situations. To our best knowledge, few researchers are working on this problem.

The obstacles lying between the disaster-handling robots and existing 3D scene understanding methods mainly include the following facts. Firstly, most of the existing approaches require the RGB information as well as the depth image. Although depth sensors are able to output stable and robust 3D images regardless of illumination conditions, it is very difficult for RGB cameras to take a picture in lower light situations [31], which, however, is very common in many disaster scenarios. Also, RGB cameras can hardly obtain consistent images in different illumination conditions. Therefore, all the 3D scene understanding methods requiring RGB data are not suited for disaster-handling robots. Secondly, the Field of View (FOV) of robots is very limited when exploring the damaged house. Many objects, e.g. furniture and electronics, maybe tilted or even overturned due to some external forces. In this situation, the on-board sensors may get the objects' image from some uncommon or unanticipated view. Since it is very difficult to recognize the objects from some specific views [39], the performance of traditional methods is usually unsatisfied. Another problem is, learning-based perception systems require a great number of data in their training process, which is a huge challenge for traditional standalone approach [2].

Therefore, to solve these problems, the author desires to propose a Convolutional Neural Network (CNN) based robotic 3D scene understanding method to improve robots' feasibility and adaptability in disaster scenarios.

1.2.2 AI for Vehicle Control

Internet-of-vehicles makes it possible to share information among the connected vehicles. A new, eye-catching concept has emerged, i.e., autonomous vehicles. In recent years, autonomous vehicles have become a popular topic, not only in the field of research but also in the application domain. Many encouraging approaches and prototypes have been made. However, the existing self-driving strategies focus too much on the "correctness", and, to some extent, overlook the human personality and social intelligence [77]. For example, during the road testing on February 14th, 2016, a Google autonomous car crashed into a municipal bus, which does not give way to the car but is predicted to slow or stop by the self-driving system. Google believes this accident is due to some misunderstanding and can

be used as a valuable experience for the self-driving system [103]. They also state that their cars will have the knowledge that the larger vehicles are more unwilling to yield, and some software adjustments will be made to avoid this type of collision in the future.

In some complex conditions, autonomous vehicles should have the ability to perform *human-like* decisions and judgments, in which both correctness and social intelligence are essential. Undoubtedly, traffic regulations must be observed. But beyond that, predictions and understandings are also required. From determining whether the car behind is going to yield to avoiding a driver who seems drunk or tired, human drivers do these speculations all the time behind the wheel. Since human drivers will exist for a long time to come, self-driving systems with poor human understanding ability may struggle in the road testing, not to mention practical applications. Indeed, in most accident reports of self-driving vehicles, it is the human drivers that should bear the main responsibility. Therefore, to improve the self-driving performance in real-life road conditions, where most vehicles are in the hands of human drivers, the author desires to design an autonomous driving system which can understand complicated road conditions and, based on that, make human-like decisions.

1.2.3 AI for Smart Grid

Smart grid, as a newly-emerged IoT application and a power system for the future, has recently received lots of attentions. Although many encouraging research works have emerged in the relevant area, one grave problem remains unsolved, i.e., electrical load forecasting. An accurate estimation of future load variation is of great significance for competitive and deregulated electricity markets, where the load prediction is an important guidance, both for power companies and electricity consumers, to make decisions and operations [99].

The major obstacle in load forecasting is the numerous impact factors. There are so many possible influences that it is extremely difficult to find a meaningful relationship between load variation and these factors. In fact, even the acquisition of necessary data is not an easy case until quite recently. The emerging of smart meter infrastructures [1], efficient sensing methods [47], and IoT technologies [7] give us a chance, for the first time, to record and analyze possible influences on a large scale. With several equipped sensors, smart meters can be used to independently capture various environment data. Also, they can obtain the shared data from IoT-enabled devices. All these data will be uploaded to the central controller. Then a massive number of data can be accumulated for further analysis. However, it is still a challenging problem to handle the data, due to the complex and variable influences, especially the diverse weather conditions. Indeed, most existing time-series forecasting approaches [26] have some limitations when applied to electrical load prediction. The

classical statistical methods are criticized for their limited abilities in handling non-linear data; and the computational intelligence methods are facing problems like inappropriate hand-crafted features, limited learning capacity, inadequate learning, inaccurate estimation, insufficient guiding significance, etc.

To solve these problems, the author desires to utilize the state-of-the-art deep learning methods [43] to automatically extract features from the historical data, and give an accurate estimation of future load value.

1.2.4 AI for Crowdsensing

With the rapid development of IoT, many applications have emerged to process the real-time sensing data, analyze its contents, find the hidden patterns, and ultimately, give the right labels or predict the future trends. In the other hand, crowdsensing is an economic and efficient approach to collect data on an extensive scale, and can be used as a scalable and stable method for some costly and complicated tasks in the IoT research. It has become a hot topic in recent years. The progressive development of miniaturized sensing and computing devices, especially the explosive growth of mobile phones, tablets, and wearables, gives significant prominence to the crowdsensing [104]. Both the academia and the industry have recognized its values, for example, to conduct the data collection with no large-scale investment. Many companies have utilized crowdsensing to acquire data at relatively low cost, in order to support their online service based on the captured data, some of the most notable examples being Facebook, Google and Uber. However, the boom of crowdsensing also brings about two grave problems.

First, the crowdsensing applications create a huge number of data, which poses significant communication and computational costs for the existing cloud infrastructure [32]. Compared with the centralized servers, the user-end devices have limited batteries and computing abilities. Therefore, existing crowdsensing systems usually put the heavy computation tasks into the centralized servers, such as the data processing and analysis. As a typical scenario in current cloud applications, the major part of the computational burden is usually shifted to the centralized servers, resulting in the rapid increase of communication frequency and server calculation load. The former brings significant traffic to the cloud infrastructure; and the latter one leads to overwhelming, sometimes unbearable, load to the centralized servers.

The other concern is about the respect of crowdsensing contributors. In fact, it has been a long time since the users are excluded in the crowdsensing process. Although most service guarantees the right to know and to decide, it is very difficult for users to truly get involved in the processing loop, e.g., to what extent the privacy should be protected, to what amount the device power can be consumed, etc. The compromise between data uploading,

which may cause the privacy leaks [65], and local computing, which will result in the energy consumption of the mobile devices, is merely decided by the service providers, rather than the actual device users.

To address these problems, the author desires to adopt the state-of-the-art edge computing and deep learning methods to balance the workloads in the cloud, and give the control of crowdsensing process back to the users. Edge computing pushes calculation tasks away from the central points to the logical boundaries of the cloud; and deep learning is a good choice to conduct the data processing task, simultaneously considering the user privacy and communication cost.

1.3 Originality and Contributions

The main innovations and contributions of this dissertation include:

- The author proposes a scene understanding method requiring only depth information instead of unstable RGB data, which gives robotic the ability to work in extreme conditions. Two different 3D encodings are used for object proposal and classification, respectively, to further improve the performance of scene understanding. A view-invariant module, which can extract similar features from different views on a same object and adapt robotic for disordered environments, is designed. The author works out a multi-task learning method. In addition, the author also designs a fast optimization method to accelerate the optimization process of complex loss functions.
- The author works out an autonomous decision-making system to imitate human drivers' social intelligence, which can better adapt the self-driving vehicles to the real-life road conditions. An efficient training scheme is designed for self-driving systems, which can significantly improve the quality and speed of data collection, and alleviate the tedious and time-consuming manual labeling process. The author finds an approach to analyze the reason that the deep learning system makes specific decisions. As far as I know, this is one of the first attempts in the self-driving area. In our opinion, this kind of analysis is of big significance to testing and validating autonomous driving systems.
- An IoT-enabled load forecasting system based on the state-of-the-art deep learning technologies is proposed. Compared to the traditional time-series analysis methods, the proposed method can perform accurate prediction without hand-crafted features. The author designs an ingenious two-step forecasting scheme, which forecasts the daily total consumption at first, and based on that, predict the intra-day load variation. This method can significantly improve the forecasting precision, which is demonstrated in the experiment section. The author works out an analysis method of possible

influence factors. To my best knowledge, this is the first attempt to gain insight into the relationship between the factors and the actual load, which, we believe, will play a tremendous role in selecting attribute combination and deploying smart meters, especially for the smart grids in some countries with vast territory and varied climates.

- A distributed deep model is proposed for the sensing problem. The proposed model can fully utilize the edge computing resource, and reduce the calculation load of the cloud servers. The author transforms the crowdsensing task into a hierarchical computing problem, and allocate different layers to different computing nodes. With the proposed method, crowdsensing contributors obtain the right to decide the balance between the privacy protection and energy saving. The author works out a dynamic learning model for the changing sensing tasks. In this model, the higher layers can be flexibly modified in the centralized servers, while keeping fixed lower layers in the edge nodes and devices.

1.4 Organization

The rest of the dissertation is organized as follows. Chapter 2 gives the fundamentals and related works of several intelligent IoT applications, including robotic vision, autonomous vehicles, smart grid, and crowdsensing. Chapter 3 proposes a CNN-based robotic 3D scene understanding method for the Internet-of-Robots, as well as several loss functions to perform multi-task learning. An optimization algorithm is also presented to improve the learning speed. Chapter 4 gives a human-like autonomous driving system for the Internet-of-Vehicles, which can imitate human drivers' social intelligence, in order to better adapt the self-driving vehicles to the real-life road conditions. This chapter also includes an efficient training scheme to improve the quality and speed of data collection, and a feasible visualization approach to analyze the possible influence factors in the decision-making process, which can help in the testing and validating of autonomous driving systems. Chapter 5 shows an IoT-based electrical load forecasting method for the smart grid, which can significantly increase the prediction precision for daily total consumption. In this chapter, the author also proposes an analysis method to find the relationship between the influences and the electrical load, and design a heatmap generation method to show the specific impacts of each attribute on forecasting results. Chapter 6 introduces an edge computing based crowdsensing method for the Internet-of-Sensors, which can adopt the available computing resources in the whole network, both in the cloud and in the edge side, to ensure the load balance and reduce communication cost. The author also gives a specially-designed deep model to transform the

crowdsensing problem into a hierarchical task for better data security. Finally, conclusions are drawn in Chapter 7.

Chapter 2

Fundamentals and Related Work

2.1 Scene Understanding

2.1.1 Object Detection and Localization

This is an old topic emerging with the boom of 2D image processing, detection, segmentation and classification. It is a prerequisite of the object classification in images and scenes, where lots of objects exist and mix together in a complicated way. A simple and effective method to solve this problem is the exhaustive search, i.e. searching all possible locations with all possible scales, which is called sliding window. Dalal and Triggs [15] use a conventional non-maximum suppression to run with detection windows to detect object instances. Harzallah et al. [24] apply SVM for each sliding window to select a few candidate regions, and use a score function on these regions to give out the final results.

However, the traditional methods working well in 2D images cannot be directly adopted in 3D images. Much more work needs to be done for 3D object proposal applications. Gupta et al. [22] extends Multiscale Combinatorial Grouping (MCG) framework [3] to 3D, and use RGB-D contours to calculate object candidates using features of depth and color images. Song and Xiao [83] present a CNN model taking a 3D volumetric scene as input to predict the object boundaries.

The author desires to develop a CNN based object detection and localization network, and adopts a single model for both two tasks. The focus of this research is a multi-task loss function, which makes the proposed method significantly different from existing ones.

2.1.2 Object Classification

With the rise of artificial intelligence, object classification has become a booming research area in recent years. Lots of encouraging approaches [53] have been proposed in this field, especially for 2D images. Krizhevsky et al. [40] design a deep CNN model and achieve amazing classification accuracy. They use a novel regularization method called "dropout" to reduce overfitting which frequently happens in deep learning. Their model is proved very effective and becomes one pioneer in CNN based object classification methods. Szegedy et al. [89] present a large CNN architecture and set a new state-of-the-art for object classification. They introduce a carefully crafted design which can increase the size of network and keep the computational budget constant.

Again, traditional 2D methods meet frustrations in 3D world, due to the unique structure and characteristics of 3D data. Many researchers try to address this problem from various angles. Socher et al.[80] adopt two CNN models to extract low-level features from RGB and depth data separately, and an RNN model to combine these two parts and learn high-order features. The research of Shi et al. [78] is the most similar work with the proposed method. They also achieve a rotation invariant representations for 3D shape. The difference is they adopt a row-wise max-pooling layer to change the low layer features extracted by the convolutional layers, while the author implements view-invariant during learning process using specially-designed loss functions. The view-invariant ability improves the precision and adaptability of the proposed method in some extreme conditions.

2.2 Autonomous Decision-making

With the rapid development of computer vision, engineering, networking, etc. [10, 8, 85, 86, 45, 46, 64, 63], the newly-emerged autonomous vehicle has become a feasible and promising technology. However, the driving decision-making is still a research area under development. Although many researchers have proposed their theories and implementations [4, 33, 19, 81], there are lots of problems to be resolved. Currently, most of the driving decision-making systems can be categorized into two major paradigms, i.e., behavior reflex approaches and abstraction calculation approaches.

Behavior reflex approaches directly map the input images to several pre-defined driving commands. LeCun et al.[60] propose an obstacle avoidance system for mobile robots. This system is trained from end to end to map raw input images to control commands. A CNN model is used in this system to learn the mapping relationship between images and driver's steering angles. And the mobile robot achieves an excellent performance in obstacle detection and path navigation. Hadsell et al.[23] work out a self-supervised learning method for mobile

robots with long-range vision. In this system, a deep CNN model is trained to extract informative and meaningful features from captured images, and predict the transferability of the input scene. The classifier is real-time and can obtain obstacles and paths from 5 to 100 m.

Abstraction calculation approaches abstract the road condition to some representations for better understanding and calculation. Chen et al.[9] map the input scene to several key perception indicators which are directly related to the affordance of current road condition, and adopt a simple controller to control the self-driving vehicles. Simulation results demonstrate their system can generalize well to real driving images. However, their controller logic is oversimplified for real-life road conditions. Xiong et al.[100] combine deep reinforcement learning and safety-based control, and propose a self-driving and collision avoidance system, which can learn the driving policy in a stable and familiar environment. This is an interesting attempt, but their research is not complete and sufficient enough for a truly feasible autonomous driving system.

The proposed decision-making system belongs to the abstraction calculation approaches. And the most significant difference between our method and these existing approaches is, we put our emphasis on the human-like thinking ability. The proposed system has a complete and sophisticated control logic. Also, we combine the decision-making with the security control to make an efficient and secure self-driving system.

2.3 Electrical Load Forecasting

2.3.1 IoT-enabled Smart Meter Systems

Smart meter is a modernized electrical device that records the energy consumption and uploads it to the utility for billing and further analysis. The most cutting-edge smart meters not only have a two-way communication ability, but are equipped with real-time sensors which can gather the data of relevant factors. This kind of electricity meter is a vital part of advanced metering infrastructure (AMI), a system keeping the whole smart grid connected and informed. With AMI, the utility can obtain necessary data from the client-side, and push notifications and recommendations to the clients. Connected smart meter is a fundamental component for the future smart grid, and also a cornerstone of this research. Many researchers are working on this area [54].

IoT, as a hot topic in recent years, is a good approach to implement connected AMI systems. However, like any other applications using IoT technologies, the underlying network to connect smart meters must be carefully investigated. Some researchers find it necessary to

clarify the exact capacities of existing wireless networks for the upcoming smart metering traffic [62]. A few preliminary conclusions have been drawn, including decreasing the communication interval and equipping phasor measurement units (PMUs).

In addition, an efficient distributed communication architecture has been proposed for the connection of smart meters [35], which can leverage data processing locally. Besides, with a carefully selected control centers [48], the cost of deployment and communication can be significantly decreased.

2.3.2 Time-series Forecasting Methods

Although time-series forecasting is a topic with a long history, it is still an open problem due to its complexity. The existing approaches are of two main types, namely, statistical methods and computational intelligence methods.

The statistical method is an obvious and natural solution when dealing with a series of numbers, including many algorithms with different design principles [67]. There is a famous exponential smoothing method called Holt-Winters (HW). HW is a good choice when the time-series shows both trend and seasonality. Two sub-models are included in HW, i.e., additive model for data with additive seasonality, and multiplicative model for data with multiplicative seasonality.

Although many classical statistical methods have emerged over the past few decades, they are currently disfavored due to their limited abilities in handling complicated nonlinear relationships. Computational intelligence, one of the hottest topic in current academia, becomes a key technology to accurately analyze and forecast time-series data. And, among all these computational intelligence methods, deep learning is in evidence [14]. Deep learning is a newly developed and fast-growing class of machine learning algorithms. A deep network has multiple hidden layers between input and output layers, in order to model complicated non-linear relationships. With enough training materials, which usually are labeled data, the parameters in a deep network can be well trained to extract complex features from large data. Therefore, deep learning methods have been successfully applied in lots of fields, including scene understanding, natural language processing, self-driving, audio recognition, etc. Because of its strong automatic feature extraction and pattern recognition ability, deep learning based methods are extremely suitable for electrical load forecasting, where lots of influences exist. The most similar approach with the proposed method is short-term deep neural network (SDNN) model [72]. This model contains three steps, i.e., data preprocessing, network training and forecasting. The historical weather conditions and load values are used as the network input. Compared to the aforementioned deep learning based approaches, the

proposed method adopts a specially designed two-step forecasting scheme, and takes into account various influences to analyze their impact.

2.4 Intelligent Sensing and Analysis

2.4.1 Current Crowdsensing Approaches

Crowdsensing is a technology and a trend to utilize the rapidly-developing sensing and computing ability of Internet-of-Sensors, to gather, process, calculate, and analysis the data generated by the environment, society, and other sources. Crowdsensing is usually for the public affairs or some wide-range commercial applications.

The first problem in crowdsensing is how to persuade users to participate in the projects, and encourage them to make contributions positively on data collection and sharing. Several incentive mechanisms are proposed for this purpose [56, 47]. The prime principle is to design a trading model, in which all captured data has a value and can be sold to some companies. There are two mainstream mechanisms, i.e., auction and lottery. The former one is the most common mechanism in crowdsensing. The bidders are the users who have the mobile sensing devices, and the auctioneers attempt to buy the data from the bidders. The latter one focus on the even distribution of the winning positions, and the winner is decided by a probability.

Another important concern is the sensing cost. There have been some researchers working in this area to reduce the overall sensing cost while ensuring data quality. A novel way is to leverage the temporal and spatial relationship between the data captured in different areas, in order to decrease the essential number of allocated sensing tasks [98]. A prediction framework is proposed to predict the data of unsensed area with the captured data in some selected areas.

Communication cost is also a consideration. The rapidly growing crowdsensing applications impose heavy burdens on the existing network. Therefore, the performance of these applications may deteriorate in some scenarios because of the overwhelming communication requests. A congestion-ware paradigm is worked out for dense crowdsensing to ensure load balancing and reliable communication among mobile devices [88].

2.4.2 Opportunities in Edge Computing

In the era of big data and mobile computing, an extensive number of applications have emerged and become important solutions in a lot of areas, such as taxi sharing, mobile payment, social networks, etc. More and more services become computation-intensive and cloud-based, which, however, gives a heavy workload to both the network infrastructure

Table 2.1 Features Comparison of Crowdsensing Methods.

	Existing Approaches	Proposed Approach
Load	Concentrated in servers	Distributed among cloud
Traffic	Raw data uploading	Extracted features
Analysis	Need further processing	Already processed
Privacy	Sensitive data	Irreversible data
Controllability	Controlled by provider	Controlled by user
Adaptability	Fully applicable	Available in 5G network

and the cloud servers. As a good solution, edge computing becomes popular due to its ability to offload the computational load from central servers to the devices near the user end [68, 92, 52, 97, 73, 5, 38, 74]. One of the most obvious advantages of edge computing is that it can empower the edge nodes with some essential computing abilities, which can provide lower response latency and better resource utilization rate. Another advantage is that it can balance the workload of the cloud servers [93], decreasing the possibility that they are overloaded. These features of edge computing structure can largely alleviate the aforementioned problem. Several instructive examples have been presented [70, 69, 90, 58, 18, 36, 91, 25, 49]. Bilal and Erbad [6] present an approach to improve users' experience of video generation and streaming with mobile edge computing. The authors work on the "edge based video generation" concept, and desire to implement a robust and efficient video generation and streaming approach for gaming interactive videos, multi-view videos, 360-degree videos, etc. The proposed method makes sure that the system can conduct the computation with the edge computing framework. Therefore, it can significantly improve the QoS of video service and give the users better experiences when using the edge computing based applications and services. Tran et al. [94] give another example using the edge computing approach for the video streaming. They use the edge servers to transcode the desired videos into several different versions, which have different resolutions or bit rates, and, therefore, differ in data size. This design can adapt the raw videos to various devices with different internet connection speed and video playing abilities.

2.4.3 Deep Learning for Crowdsensing

Although edge computing can partly solve the aforementioned problems in crowdsensing, another approach is needed to process the captured data, including the compression, encryption, and analysis. The compression is for the further decrease of communication cost, the

encryption is used for protecting the users' privacy, and the analysis is essential to make the captured data into full play and utilization.

The compressed sensing [87] is a representative case for data compression. Based on a deep learning method, a binary autoencoders scheme is designed for compressed sensing, in which a binary sensing matrix and a recover solver are jointly optimized in the network training. Results show that the compressed sensing performs well in efficiency, and is preferable for real-time wireless applications.

Will deep learning revolutionize crowdsensing? [42] gives preliminary answers by implementing a prototyping deep learning engine on a mobile device SoC. Compared with other approaches in several typical crowdsensing tasks, the deep learning based methods show significant advantages on inference accuracy, without overburdening the mobile hardware.

Valerio et al. combine the deep learning with edge computing and crowdsensing-like tasks [96]. They consider three scenarios, i.e., calculating all the tasks on the local devices, calculating all the tasks on the remote cloud, or calculating the tasks on the devices and cloud at the same time. They model a network cost function to measure the approximate total cost for the deep learning computing. Given a desired accuracy for a specific task, their model can be used to compute the trade-off between performance and network cost and to find the minimum network cost.

Compared to the aforementioned deep learning based approaches, the proposed method adopts a specially-designed hierarchical deep model, which can be well adapted to the edge computing framework. The difference between the proposed method with other deep learning based approach is shown in Table. 2.1. The proposed method can solve many problems existing in the traditional approaches, such as the computing distribution, the communication cost, and the user controllability. The main differences between this approach and the existing ones include the following aspects. First, in the traditional solutions, the centralized servers take all the burden of computation task, while in the proposed approach, most of the computing resource in the cloud can be utilized, leading to a better balanced cloud environment. Second, in cloud-based approaches, each device sends the captured data to the centralized servers in the cloud, resulting in significant traffic to the network infrastructure, while in this method, only the extracted features are uploaded to the upper layer, instead of the raw data, which contains lots of redundant data. Third, as one major drawback of the existing approaches, the user devices have to upload the sensitive data to the centralized servers for further process, which may cause privacy leakage, while in this method, users can decide which level of abstraction can be uploaded. The higher level the abstractions are, the more security the user data has. However, there also have some disadvantages, for example , the network adaptability. Unlike the traditional methods which have few requirements

on the network infrastructure and, therefore, are fully applicable on existing devices, the proposed method need the basic edge computing structure which needs upgrading on the cloud infrastructure, especially on the base stations.

Chapter 3

Artificial Intelligence for Internet-of-Robots

In the proposed approach, robots equipped with depth sensors are sent to explore dangerous environments, as shown in Fig. 3.1. With the ability of 3D scene understanding, robots can deal with various conditions and give great help to disaster preparedness, relief and recovery. Every robot is self-contained, but can also share its information and understandings with the Internet-of-Robots. Each robot has a CNN network installed in its firmware. Robots can upload its captured data to the nearby server for further fine-tuning, and the updated parameters will be transferred back to the robots to improve their adaptability to their current environments.

3.1 System Overview

3.1.1 Notation and Problem Definition

Boldface uppercase letters are used to denote matrix, e.g. W_θ, lowercase letters for vectors, e.g. $w_\theta^{(m)}$, and calligraphic-font letters represent sets, e.g. $\mathscr{W}$.

Given a collection of 3D images $\mathscr{X}$, the goal of scene understanding can be described as follows. For each input image x_i, the proposed method should detect and localize as many as possible objects $\mathscr{O}$, and classify them with correct labels $\mathscr{L}$.

3.1.2 System Framework

Fig. 3.2 presents an overview of the proposed method. For better understanding, a brief introduction on the main concepts is given in this section.

Fig. 3.1 Robotic for Disaster Scenarios.

For 3D scenes, two individual models are used to conduct the object proposal and classification task, respectively. The objective of object proposal is to find any potential objects in the images, and, if any, localize them with precise locations and boundaries. This is an essential procedure for the following classification process, as classification model can only deal with a single object and is easily confused if there are multiple objects in the classification area. Therefore, the classification network will use the output of object proposal to clarify what the object is. Before the introduction of these two models, an explanation on the 3D data representations will be given, because input data must be encoded before it is used.

3.1.3 3D Data Encoding

The first challenge of 3D computer vision is the data encoding. Generally, the raw data captured by 3D sensors is in the form of point cloud, as shown in Fig. 3.3(a). This original scene comes from the RGB-D Object Dataset [41], which is also one of the datasets used in the learning and testing process. One common encoding method is to map the 3D point into RGB space, depending on the distance between the sensor and each point, as shown in Fig. 3.3(b), which is called the 2D colored representation. Another method is to transform the point cloud data into 3D volumetric representation, as shown in Fig. 3.3(c).

Currently, both methods have their advocates. However, according to [84], a CNN model trained with 2D-encoded data can obtain a significantly better performance in classification tasks than the one trained with 3D representation. This is mainly because that the advantages of 2D representation is very useful in classification tasks, e.g., the availability of well-trained network and the high resolution of 2D images. Therefore, input images are encoded

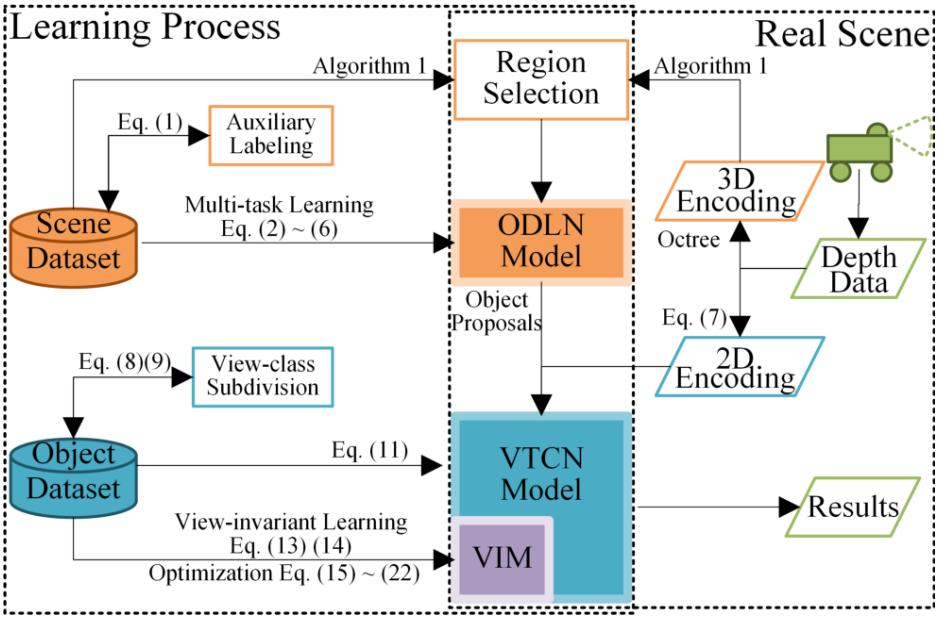
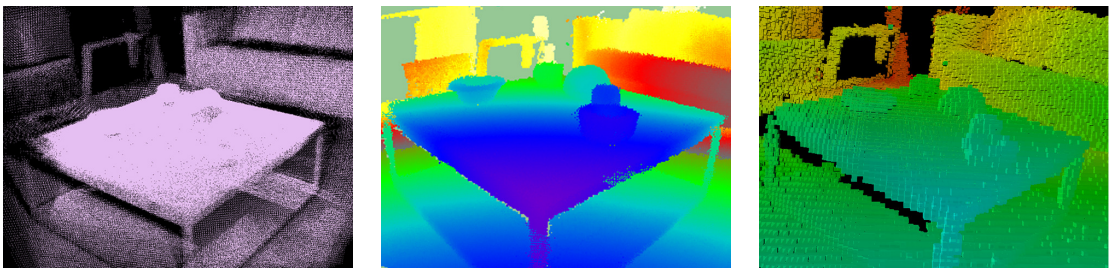


Fig. 3.2 Framework of the Proposed Scene Understanding Method.



(a) (b) (c)

Fig. 3.3 3D Scene Representation[41]. (a) Original point cloud. (b) 2D colored representation. (c) 3D volumetric representation (resolution = 0.02 m).

to 2D representation in the proposed classification model. However, in object detection and localization task, the author finds 3D volumetric representation is more preferable, as introduced in [51], owing to its excellent ability of preserving 3D spatial information. For this reason, 3D volumetric encoding is used in the detection and localization task. It is another difference between this job and existing approaches that the author selects different representations for different tasks, instead of using one same representation in all applications.

3.2 3D Object Proposal

3.2.1 ODLN Model

Object proposal is very important for the whole scene understanding process, because it can greatly decrease the recognition area and accelerate the classification network. A CNN-based Object Detection and Localization Network (ODLN) is proposed in this section to accurately propose potential object candidates for subsequent calculations. As previously mentioned, the author desires to propose a scene understanding method without the requirement of RGB data which is full of uncertainty. As a result, it is very difficult to work out a selective search scheme like [95] and [83], in which color information is indispensable for similarity calculation. Therefore, the author designs a region selection method which is very similar with sliding window. By constantly selecting all possible regions in FOV, all objects could be included in at least one region, and get imported into ODLN for further calculation. The author also adopts an auxiliary labeling method to help people mark a massive number of raw data and improve the practicability of supervised learning. The region selection algorithm and auxiliary labeling method are presented in Algorithm 1. Several hand-crafted features [11] are used to perform auxiliary labeling on initial dataset, these features are learnt with SVM. Fig. 3.5 shows the effects of the auxiliary labeling method and the proposed ODLN model. Although the hand-crafted features are outperformed by self-learning ODLN, it can save people lots of time by labeling the scene data automatically, especially when a large dataset is used for learning. The templates used for region selection include 10 cuboids in different sizes.

The network architecture of ODLN is presented in Fig. 3.4. There are two specially-designed modules and three ordinary convolutional layers behind the input layer, following by two individual sub-networks which will output detection and localization results respectively.

The input of ODLN is the volumetric representation of 3D objects in each region with their labels. The objective of ODLN is to judge whether there is a single object in the input region, and its exact boundaries and orientation.

Inspired by [76] and [89], the author designs the Scale-Insensitive Convolutional (SIC) module. There are three convolutional kernels with different size in one SIC module, which can extract the features from the input data in different scale. SIC module gives ODLN the ability to cope with 3D objects in various sizes.

Two sub-networks are behind the convolutional networks for detection and localization. The detection network outputs a value between $[0,1]$ representing the possibility y that there is a single object in current region. The localization network is to predict the object's center point $\{x,y,z\}$, dimensions $\{\ell,w,h\}$ and Euler angles $\{\alpha,\beta,\gamma\}$, as shown in lower right corner

Algorithm 1: Region Selection and Object Proposal

Input: 3D Scene Data $\mathscr{S}$, Region Templates $\mathscr{T}$
Output: Object Proposals $\mathscr{P}$
proposal_list = null;
for $v \in \mathscr{S}$ **do** `/* Traversal of voxels. */`
  for $t \in \mathscr{T}$ **do** `/* Try all Templates. */`
    if *is_auxiliary_labeling* **then**
      `/* Perform auxiliary labeling */`
      Compute

$$\begin{aligned} \text{Score}(v) = \lambda_1 \psi_{point_cloud} + \lambda_2 \psi_{free_space} \\ + \lambda_2 \psi_{height_prior} + \lambda_2 \psi_{height_contrast} \end{aligned} \tag{3.1}$$

      if Score$(v) >$ threshold **then**
        Append v to proposal_list.
    else `/* Imported to ODLN Model. */`
      Score(v), Location(v) = ODLN $\leftarrow v$;
      if Score$(v) >$ threshold **then**
        Append Location(v) to proposal_list.
return proposal_list;

of Fig. 3.4. Both sub-networks have two fully-connected layers for further analysis and abstracting on the features originate from the lower layers. Specially, the author designs a multi-task loss function to train two sub-networks at a same time, which is detailed in 3.2.2.

In short, 3D data in every possible region is imported into ODLN with their labels, which have the information including whether the region has an object and its locations. After forward propagation, ODLN can use the calculated values, their labels and the proposed loss function to adjust weights and optimize the network. During this process, ODLN can be gradually trained into a desirable network.

3.2.2 Learning with Multi-task Loss Function

Given n input data $\mathscr{X} = \{x_1, x_2, ..., x_n\}$ in ODLN, there are several different labels for each input data x_i, i.e., ground-truth labels $\mathscr{Y} = \{y_{x_1}, y_{x_2}, ..., y_{x_n}\}$ and 3d-box labels $\mathscr{T} = \{t_{x_1}, t_{x_2}, ..., t_{x_n}\}$, where $t \in \{x, y, z, \ell, w, h, \alpha, \beta, \gamma\}$ standing for locations, dimensions and orientations respectively. In regard to the possible labels, $y \in \{y_1^*, y_2^*, ..., y_m^*\}$ and t should be real values. Specially, in the detection subnetwork of ODLN, $m = 2$ and $y \in \{0, 1\}$. $h_{\text{cls}}(x_i)$ is

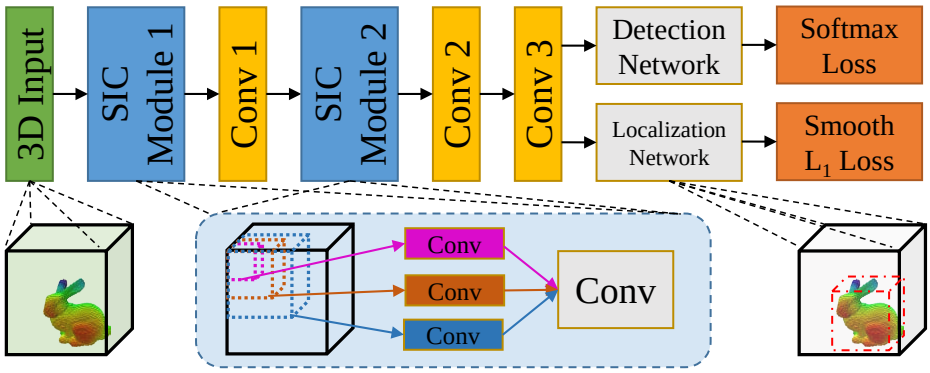
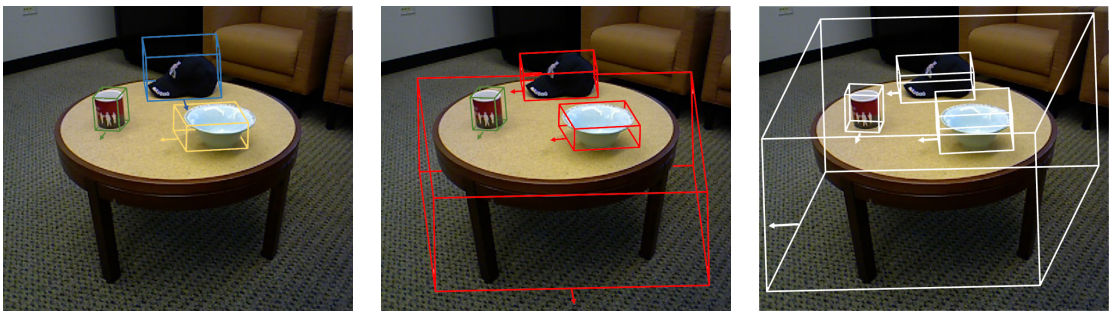


Fig. 3.4 Network Architecture of ODLN.



(a) (b) (c)

Fig. 3.5 3D Scene[41] Labeling(resolution = 0.02 m). (a) Auxiliary labeling. (b) Manual amendment. (c) Actual proposal result using ODLN.

the output of Softmax classification layer, and $h_{\text{loc}}(x_i)$ is the one of Smooth L_1 Localization layer.

Inspired from the Multi-task Loss in [83] and [20], the author designs a loss function for ODLN as follows.

$$\begin{aligned} J(\mathscr{W},\mathscr{B},\mathscr{X},\mathscr{Y},\mathscr{T}) =& \lambda_1 J_{\text{cls}}(\mathscr{W},\mathscr{B},\mathscr{X},\mathscr{Y}) \\ &+ \lambda_2 J_{\text{loc}}(\mathscr{W},\mathscr{B},\mathscr{X},\mathscr{T}) + \lambda_3 J_{\text{dec}}(\mathscr{W}) \end{aligned} \tag{3.2}$$

where J_{cls}, J_{loc} and J_{dec} are sub-loss functions for different objectives, and λ_1, λ_2 and λ_3 are weights to custom the relative importance of each term.

J_{cls} is the softmax loss function for measuring the accuracy of the final detection result, i.e. whether a region owns one single object or not. It can be expressed as

$$\begin{aligned} & J_{\text{cls}}(\mathscr{W},\mathscr{B},\mathscr{X},\mathscr{Y}) \\ & = -\frac{1}{n}\left[\sum_{i=1}^{n}\sum_{j=1}^{m}\mathbb{1}\{y_{x_i}=y_j^*\}\log\frac{e^{h_{\text{cls}}^{(j)}(x_i)}}{\sum_{\phi=1}^{m}e^{h_{\text{cls}}^{(\phi)}(x_i)}}\right] \\ & = -\frac{1}{n}\left[\sum_{i=1}^{n}\sum_{j=0}^{1}\mathbb{1}\{y_{x_i}=y_j^*\}\log p(y_{x_i}=y_j^*|x_i;\mathscr{W};\mathscr{B})\right] \\ & = -\frac{1}{n}\left[\sum_{i=1}^{n}y_{x_i}\log(h_{\text{cls}}(x_i))+(1-y_{x_i})\cdot\log(1-h_{\text{cls}}(x_i))\right] \end{aligned} \tag{3.3}$$

where $h_{\text{cls}}^{(j)}(x_i)$ varies from 0 to 1, according to the possibilities that there is one single object in this region.

J_{loc} is utilized to predict the object's localization. Essentially, it is a regression of objects' boundary. According to the smooth L_1 loss in 2D bounding-box regression method of Fast R-CNN[20], J_{loc} is defined as

$$\begin{aligned} & J_{\text{loc}}(\mathscr{W},\mathscr{B},\mathscr{X},\mathscr{T}) = \\ & \sum_{t\in\{x,y,z,\ell,w,h,\alpha,\beta,\gamma\}}\frac{1}{n}\sum_{i=1}^{n}\text{smooth}_{L_1}(h_{\text{loc}}^{(t)}(x_i)-t_{x_i}) \end{aligned} \tag{3.4}$$

and

$$\text{smooth}_{L_1}(z)=\begin{cases}0.5z^2, & |z|<1\\ |z|-0.5, & |z|\geq 1\end{cases} \tag{3.5}$$

where t represents the coordinates of the object's center point $\{x,y,z\}$, dimensions $\{\ell,w,h\}$ and Euler angles $\{\alpha,\beta,\gamma\}$.

$J_{\text{dec}}(\mathscr{W})$ is a weight decay term which penalizes large values of the parameters to decrease the magnitude of the weights $\mathscr{W}$ and help prevent over-fitting. It is computed by the following function.

$$J_{\text{dec}}(\mathscr{W}) = \| \mathscr{W} \|_2^2 \tag{3.6}$$

The proposed loss function takes object's detection and localization into account at a same time, which is a huge advantage compared to the common-used loss functions. As a result, the output of ODLN can be directly used as the input of object classification network. The performance of ODLN is evaluated in 3.4.1.

3.3 3D Object Classification

3.3.1 VTCN Model

A View-invariant 3D Classification network (VTCN) is proposed in this section. As mentioned above, 2D encoded image is adopted as the input of network. The depth value can be transformed to RGB data using the following formula.

$$\text{depth} \mapsto \underset{(0\sim360,1,255)}{(h,s,v)} \mapsto (r,g,b) \tag{3.7}$$

Therefore, VTCN has a similar structure with common 2D image classification CNN models, e.g. AlextNet [40], which is used as the prototype of the proposed VTCN model. To develop a classification model used in disordered and extreme scenarios, the author desires to develop a view-invariant CNN model for feature extraction. The author gains inspiration from [12] and proposed a new method which can be used with 3D data. A View-invariant (VI) Module consisting of two fully-connected layers is added between the 6th and 7th layer of AlexNet model. The author also presents a new object category subdivision method to give VTCN the ability to deal with multi-view images. With the well-learnt low-layer features brought by AlexNet, the high-layer features can get fine-tuned using 3D data in 2D representation. This learning process is conducted using specially-designed loss functions.

The class subdivision method and learning process will be detailed below, respectively.

3.3.2 View-subclass Definition

One of the most significant differences between VTCN and other methods is, one object category is not regarded as one single class. As is known to us, even a same object looks very different in different views. For example, it is very difficult to recognize a can if the viewpoint is directly below it. There would be even greater difficulties for robots with only depth information. Given these facts, the author subdivides the existing object categories into several "view-subclasses", which will be the ground-truth label imported into VTCN model, as shown in Fig. 3.6.

So how to definite a view-sub-class? Interestingly, this is very similar to robotic view planning problem. In view planning methods, a next-best-view (NBV) is generated during each iteration of model reconstruction, then the depth sensor is moved to NBV to conduct the next scan. In the same way, several NBVs can be generated for each object category and each of them is used as a cluster center for one view-sub-class. Fortunately, the author has worked

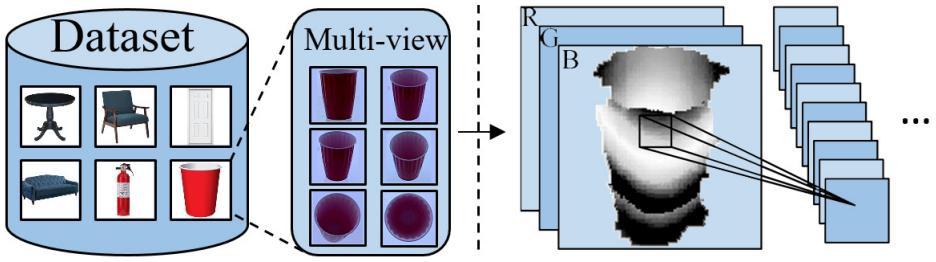


Fig. 3.6 View-subclass Division and the Input of VTCN.

out an efficient robotic view planning method in [50], which can be directly used in VTCN. Because this algorithm is not the main topic, it will merely be given a brief introduction.

At first, NBV candidates are generated at all possible viewpoints in the whole working space, using the hierarchical generation and filtration algorithm proposed in the author's past work [50]. And they are evaluated using the score function presented below.

$$N(p) = \omega_1 \,\text{volume}(p) + \omega_2 \,\text{overlap}(p) \tag{3.8}$$

where $\text{volume}(p)$ represents the number of newly-observed voxels, $\text{overlap}(p)$ is the coincidence region between the new view and existing views. These two terms can control the number and orientation of the view-sub-class for each object category. $N(p)$ is the view point's score. And, the view with highest $N(p)$ score is selected as a new view-subclass.

After the sub-class centers are generated, they are extend to the whole working space. All of the viewpoints will be subjected a view-sub-class, depending on their similarities with the class centers. Given the points set $\mathscr{P}$ obtained in view v, the similarity function can be described as

$$\text{Similarity} = \frac{\mathscr{P}_v \bigcap \mathscr{P}_{\text{center}_v}}{\mathscr{P}_v \bigcup \mathscr{P}_{\text{center}_v}} \tag{3.9}$$

Then every view belongs to a center v, which owns a view-sub-class, with the highest similarity score.

3.3.3 Learning View-invariant Features

Suppose there are n object view-subclasses, then the input data of VTCN can be defined as

$$\begin{cases} \mathscr{X} = \{\mathscr{X}_1, \mathscr{X}_2, ..., \mathscr{X}_n\} \\ \mathscr{X}_i = \{x_i^{(1)}, x_i^{(2)}, ..., x_i^{(i_m)}\} \\ \Rightarrow \mathscr{X} = \left\{ \{x_1^{(1)}, x_1^{(2)}, ..., x_1^{(1_m)}\}, ..., \{x_n^{(1)}, ..., x_n^{(n_m)}\} \right\} \end{cases} \tag{3.10}$$

where $\mathscr{X}$ is the set of all input samples, and $\mathscr{X}_i$ is the set of all the views of one single view-subclass. Each view-subclass has i_m elements. Then the total number of input data SIZE$(\mathscr{X}) = \sum_i^n i_m$. The labels of input images can be represented by $\mathscr{Y} = \{y_{\mathscr{X}_1}, y_{\mathscr{X}_2}, ..., y_{\mathscr{X}_n}\}$.

Like Eq. (3.2), the loss function can be written as

$$\begin{aligned} J(\mathscr{W}, \mathscr{B}, \{\mathscr{X}_i\}, \mathscr{Y}) = \quad & \lambda_1 J_{\text{cls}}(\mathscr{W}, \mathscr{B}, \{\mathscr{X}_i\}, \mathscr{Y}) \\ & + \lambda_2 J_{\text{view}}(\mathscr{W}, \mathscr{B}, \{\mathscr{X}_i\}) \\ & + \lambda_3 J_{\text{dec}}(\mathscr{W}) \qquad (0 \leq i \leq n) \end{aligned} \tag{3.11}$$

The difference is that Eq. (3.11) needs $\{\mathscr{X}_i\}$ as the view-subclass information to learn view-invariant features.

Similarly, $J_{\text{cls}}(\mathscr{W}, \mathscr{B}, \{\mathscr{X}_i\}, \mathscr{Y})$ can be described as

$$\begin{aligned} & J_{\text{cls}}(\mathscr{W}, \mathscr{B}, \{\mathscr{X}_i\}, \mathscr{Y}) \\ & = -\frac{1}{n}\left[\sum_{i=1}^{n}\sum_{r=1}^{i_m}\sum_{j=1}^{c_{cls}} \mathbb{1}\{y_{x_i} = y_j^*\} \log \frac{e^{h_{\text{cls}}^{(j)}(x_i^{(r)})}}{\sum_{\phi=1}^{m} e^{h_{\text{cls}}^{(\phi)}(x_i^{(r)})}}\right] \\ & = -\frac{1}{n}\left[\sum_{i=1}^{n}\sum_{r=1}^{i_m}\sum_{j=1}^{c_{cls}} \mathbb{1}\{y_x = y_j^*\} \log p(y_x = y_j^* | x_i^{(r)}; \mathscr{W}; \mathscr{B})\right] \end{aligned} \tag{3.12}$$

where c_{cls} is the neuron number of final layer, i.e., the total number of output features. And, $J_{\text{dec}}(\mathscr{W})$ is same with Eq. (3.6).

Then, the key problem is how to design a view-clustering function to give VTCN the ability to extract similar features from different views of a same object. Therefore, the differences in a view-sub-class $\mathscr{X}_i$ should be calculated and deteriorated during the learning process. Based on these assumptions, we define

$$\begin{aligned} & J_{\text{view}}(\mathscr{W}, \mathscr{B}, \{\mathscr{X}_i\}) \\ & = \frac{1}{pn}\left[\sum_{i=1}^{n}\sum_{r=1}^{i_m} \parallel h_{\text{vim}}(x_i^{(r)}) - \overline{h_{\text{vim}}(\mathscr{X}_i)} \parallel_p^p\right] \\ & = \frac{1}{pn}\left[\sum_{i=1}^{n}\sum_{r=1}^{i_m}\sum_{k=1}^{c_{cls}} (h_{\text{vim}}^{(k)}(x_i^{(r)}) - \overline{h_{\text{vim}}^{(k)}(\mathscr{X}_i)})^p\right] \end{aligned} \tag{3.13}$$

and

$$\overline{h_{\text{vim}}^{(k)}(\mathscr{X}_i)} = \frac{1}{i_m}\sum_{r=1}^{i_m} h_{\text{vim}}^{(k)}(x_i^{(r)}) \tag{3.14}$$

where h_{vim} is the output of VI Module.

So the main objective is to optimize the following function

$$(\hat{\mathscr{W}}, \hat{\mathscr{B}}) = \arg\min_{\mathscr{W},\mathscr{B}} J(\mathscr{W}, \mathscr{B}, \{\mathscr{X}_i\}, \mathscr{Y}) \tag{3.15}$$

It is extremely difficult to perform this function on the entire dataset. Therefore, a batch Stochastic Gradient Descent (SGD) strategy is adopted in network learning, i.e. perform the optimization on a batch sampled from the whole dataset in each iteration. Although SGD has been proved successful on the functions like J_{cls} and J_{dec}, some more work is needed before it can be applied to the newly proposed J_{view} function. The stochastic and back-propagation method of J_{view} should be figured out to improve the computational efficiency.

Define *bt* is one batch and

$$H_{\theta,\text{bt}} = \begin{pmatrix} h_\theta(x_1) \\ h_\theta(x_2) \\ \vdots \\ h_\theta(x_{n_b}) \end{pmatrix} \tag{3.16}$$

is a matrix consisting of all the outputs of layer θ in a batch. So the right-hand side of Eq. (3.13) can be expressed as

$$J_{\text{view}}(\mathscr{W}, \mathscr{B}, \mathscr{X}_{\text{bt}}) = \frac{\| H_{\text{vim,bt}} - \overline{h_{\text{vim}}(\mathscr{X}_{\text{bt}})}\mathbb{1}_{1\times n_b} \|_p^p}{p n_b} \tag{3.17}$$

where

$$H_{\text{vim,bt}} = W_{\text{vim}} H_{\text{vim-1,bt}} + B_{\text{vim}}\mathbb{1}_{1\times n_b} \tag{3.18}$$

w.l.o.g., assume that the average of feature is consistent during one iteration, and define $\text{AVG} = \overline{h_{\text{vim}}(\mathscr{X}_{\text{bt}})}\mathbb{1}_{1\times n_b}$ Then

$$\begin{aligned} \nabla_{J_{\text{view}}} &= \begin{bmatrix} \frac{\partial J_{\text{view}}(\mathscr{W}, \mathscr{B}, \mathscr{X}_{\text{bt}})}{\partial W_{\text{vim}}} \\ \frac{\partial J_{\text{view}}(\mathscr{W}, \mathscr{B}, \mathscr{X}_{\text{bt}})}{\partial B_{\text{vim}}} \end{bmatrix} \\ &= \begin{bmatrix} \frac{(H_{\text{vim,bt}} - \text{AVG})^{p-1}}{n_b} \frac{\partial H_{\text{vim,bt}}}{\partial W_{\text{vim}}} \\ \frac{(H_{\text{vim,bt}} - \text{AVG})^{p-1}}{n_b} \frac{\partial H_{\text{vim,bt}}}{\partial B_{\text{vim}}} \end{bmatrix} \end{aligned} \tag{3.19}$$

Finally, the derivative of weights W and biases B of the last layer in VI Module can be expressed as

$$\begin{aligned}&\frac{\partial J_{\text{view}}(\mathscr{W},\mathscr{B},\mathscr{X}_{\text{bt}})}{\partial W_{\text{vim}}}\\&=\begin{cases}\dfrac{\text{sign}(H_{\text{vim,bt}}-\text{AVG})}{n_b}H_{\text{vim-1,bt}} & ,p=1\\ \dfrac{(H_{\text{vim,bt}}-\text{AVG})^{p-1}}{n_b}H_{\text{vim-1,bt}} & ,otherwise\end{cases}\end{aligned} \tag{3.20}$$

and

$$\begin{aligned}&\frac{\partial J_{\text{view}}(\mathscr{W},\mathscr{B},\mathscr{X}_{\text{bt}})}{\partial B_{\text{vim}}}\\&=\begin{cases}\dfrac{\text{sign}(H_{\text{vim,bt}}-\text{AVG})}{n_b} & ,p=1\\ \dfrac{(H_{\text{vim,bt}}-\text{AVG})^{p-1}}{n_b} & ,otherwise\end{cases}\end{aligned} \tag{3.21}$$

The weights and biases are updated as

$$\begin{aligned}W_{\text{vim}} &:= W_{\text{vim}}-\text{lr}\frac{\partial J_{\text{view}}}{\partial W_{\text{vim}}}\\ B_{\text{vim}} &:= B_{\text{vim}}-\text{lr}\frac{\partial J_{\text{view}}}{\partial B_{\text{vim}}}\end{aligned} \tag{3.22}$$

where lr is the learning rate. Then the stochastic is back-propagated to the lower layers. After a number of iterations, the loss function can convergence and give VTCN the ability to extract view-invariant features.

In addition, $\| h_{\text{vim}}(x_i^{(r)}) - \overline{h_{\text{vim}}(\mathscr{X}_i)} \|_p^p$ is the Minkowski distance of order p between $h_{\text{vim}}(x_i^{(r)})$ and $\overline{h_{\text{vim}}(\mathscr{X}_i)}$. When $p=1$, Minkowski distance degenerates into Manhattan distance, and when $p=2$, it becomes Euclidean distance. For better learning efficiency, the p value should be carefully selected. Therefore, an experiment is conducted in 3.4.3 to explore the relationship between p value and the training performance.

3.3.4 Proof of the Training Efficiency

Since Eq.(3.17) is the core component of the proposed loss functions, a formal proof should be provided in order to guarantee its training efficiency on different experiment datasets. The derivation of Eq.(3.17) has been presented in Section 3.3.3, however, the p value still has uncertain effect on the learning speed. Therefore, a detailed proof on the relationship

between the p value and the training efficiency will be given. First, the following three lemmas are needed to prove the final theorem.

Lemma 1. *The convergence rate of the gradient descent method using different types of distances will follow*

$$\begin{aligned} ManhattanDistance > & EuclideanDistance \\ & > ChebyshevDistance \end{aligned} \tag{3.23}$$

Proof. This lemma can be expressed as the following statement. For two arbitrary points $P(x_0, y_0)$ and $Q(x_1, y_1)$, when $P \to Q$,

$$d_M(DP, DQ) \geq d_O(DP, DQ) \geq d_Q(DP, DQ) \tag{3.24}$$

where D is an arbitrary point.
Without loss of generality, we can assume D is the origin point $O(0,0)$. Define $x_0 > y_0 > y_1$ and $x_0 > x_1$, then

$$\begin{aligned} d_M(OP, OQ) &= x_0 + y_0 - x_1 - y_1 \\ d_O(OP, OQ) &= \sqrt{x_0^2 + y_0^2} - \sqrt{x_1^2 + y_1^2} \\ d_Q(OP, OQ) &= x_0 - x_1 \end{aligned} \tag{3.25}$$

First, compare $d_M(OP, OQ)$ with $d_O(OP, OQ)$. We can start with the comparison between $\sqrt{x_0^2 + y_0^2} - \sqrt{x_1^2 + y_1^2}$ and $\sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}$. Then we can get

$$\begin{aligned} \sqrt{x_0^2 + y_0^2} - \sqrt{x_1^2 + y_1^2} < & \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2} \\ & < |x_0 - x_1| + |y_0 - y_1| \end{aligned} \tag{3.26}$$

Therefore, $d_M(OP, OQ) \geq d_O(OP, OQ)$.
Second, compare $d_O(OP, OQ)$ with $d_Q(OP, OQ)$

$$\frac{x_0 - x_1}{\sqrt{x_0^2 + y_0^2} - \sqrt{x_1^2 + y_1^2}} = \frac{\sqrt{x_0^2 + y_0^2} + \sqrt{x_1^2 + y_1^2}}{x_0 + x_1 + \frac{(y_0 - y_1)(y_0 + y_1)}{x_0 - x_1}} \tag{3.27}$$

Since $P \to Q$, we can get $x_0 \to x_1$ and $y_0 \to y_1$ with a same approaching speed. Therefore, $Eq.(3.27) \leq 1$. Therefore, $d_O(OP, OQ) \geq d_Q(OP, OQ)$ ■

Lemma 2. *There exists an intermediate point that enables us to express the Minkowski distance with the Euclidean distance. In other words, for two arbitrary points $P(x_0, y_0)$ and $Q(x_1, y_1)$, there exists a point $M(x_m, y_m)$ to make*

$$d_p(P,Q) = d_O(P,M) + d_O(M,Q) \tag{3.28}$$

Proof. Without loss of generality, we can assume Q is the origin point $O(0,0)$. Then

$$d_p(O,P) = (x_0^p + y_0^p)^{\frac{1}{p}} \tag{3.29}$$

When $1 < p < 2$, we can get $x_0^p + y_0^p < (x_0 + y_0)^p$. Therefore, $d_p(O,P) < |x_0| + |y_0|$.
We need to prove that there always exists a point $M(x_m, y_0)$ which makes

$$x_m + y_0 = (x_0^p + y_0^p)^{\frac{1}{p}} \tag{3.30}$$

Because

$$(x_m + y_0)^p = \sum_{i=0}^{p} C_p^i x_m^i y_0^{p-i} \tag{3.31}$$

so it can be expressed as

$$x_1^p + \sum_{i=1}^{p-1} C_p^i x_m^i y_0^{p-i} = x_0^p \tag{3.32}$$

With the Intermediate Value Theorem, we can know Eq.(3.30) and (3.32) are valid.
When $p > 2$, we have $(x_0^p + y_0^p)^{\frac{1}{p}} < (x_0^2 + y_0^2)^{\frac{1}{2}}$.
Similarly, we can prove that there exists a point $M(x_0, y_m)$ to make $(x_0^p + y_0^p)^{\frac{1}{p}} = (x_0^2 + y_m^2)^{\frac{1}{2}}$. ■

Lemma 3. *Lemma 2 is valid for spaces with more than two dimensions. In other words, the distances can be projected into lower-dimension spaces.*

Proof. We have $x = (x_1, \cdots, x_n)^T$, $y = (y_1, \cdots, y_n)^T$ and $d_p(x,y) = [\sum_{i=1}^n (x_i - y_i)^p]^{\frac{1}{p}}$.
We need to prove that there exists a point $\xi = (\xi_1, \cdots, \xi_n)^T$ to make

$$d_p(x,y) = d_0(x,\xi) + d_o(y,\xi) \tag{3.33}$$

i.e.,

$$\Big[\sum_{i=1}^{n} (x_i - y_i)^p\Big]^{\frac{1}{p}} = \Big[\sum_{i=1}^{n} (x_i - \xi_i)^2\Big]^{\frac{1}{2}} + \Big[\sum_{i=1}^{n} (y_i - \xi_i)^2\Big]^{\frac{1}{2}} \tag{3.34}$$

which can be expressed as

$$\sum_{i=1}^{n}(x_i-y_i)^p=\left\{\left[\sum_{i=1}^{n}(x_i-\xi_i)^2\right]^{\frac{1}{2}}+\left[\sum_{i=1}^{n}(y_i-\xi_i)^2\right]^{\frac{1}{2}}\right\}^p \tag{3.35}$$

Similarly, using the Intermediate Value Theorem, we know that there exists $\xi=(\xi_1,\cdots,\xi_n)^T$ to make this equation true. ■

Lemma 4. *Lemma 1 is also valid for spaces with more than two dimensions.*

Proof. It can be deducted using Lemma 2 and 3. ■

Theorem 1. *For the proposed loss function $J_{\text{view}}(\mathscr{W},\mathscr{B},\{\mathscr{X}_i\})$, the training efficiency can be improved by decreasing the p value, i.e.,*

$$\nabla_{J^p_{\text{view}}}\geq\nabla_{J^{p+1}_{\text{view}}} \tag{3.36}$$

Proof. It can be deducted using Lemma 1, 2, 3 and 4. ■

According to the theorem, a smaller p value will increase the back-propagated stochastic and improve the optimization efficiency, which is also demonstrated in the experiment 3.4.3. On the other hand, a larger p value, resulting in a smaller stochastic, may also stable the learning process. Therefore, it is reasonable to set a small p in the early stage of training to get higher efficiency, and gradually increase p for stable convergence. The above proof guarantees that the proposed method can get a good learning speed with different datasets.

3.4 Performance Evaluation

To fully reach the potentiality of the proposed networks, several existing datasets combined with the newly-scanned 3D data are utilized in the training and testing process. The proposed networks are complicated enough to contain more than one dataset without under-fitting. The adopted datasets include *NYU Dataset v2* [61], *A large dataset of object scans*[13], *RGB-D Object Dataset* [41], *Bigbird dataset* [79] and *SceneNN dataset* [30].

3.4.1 ODLN Performance

The proposed ODLN not only detect objects in images, but localize them with exact position, dimension and orientation. To compare ODLN with some existing approaches, the following

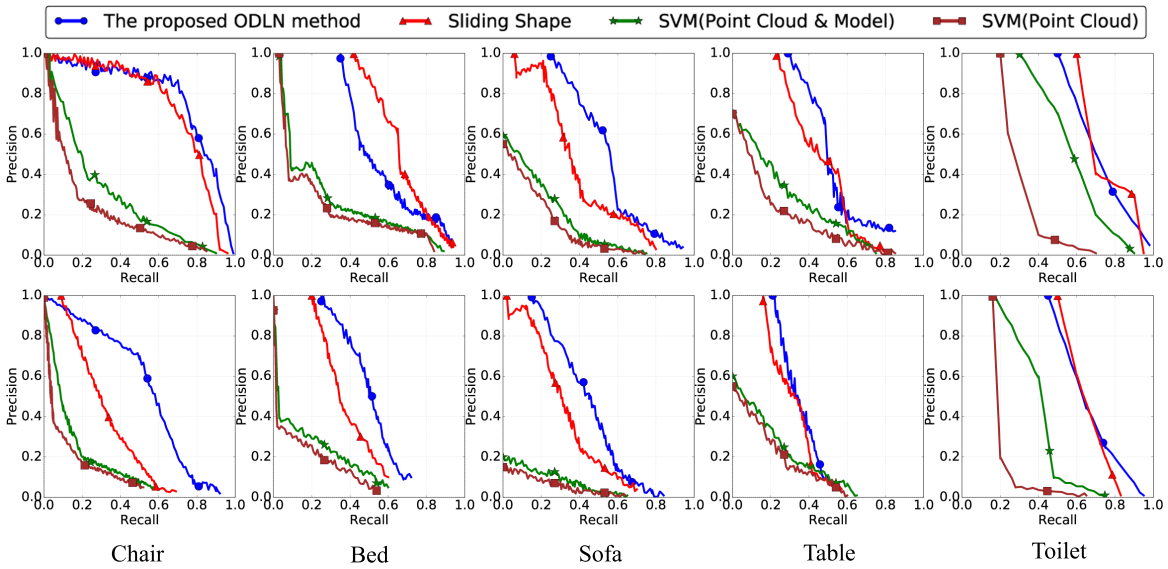


Fig. 3.7 Comparison of 3D Object Proposal Methods.

formulas are adopted to evaluate the performance.

$$\text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}} \tag{3.37}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} \tag{3.38}$$

where TP, FP and FN mean true positive, false positive and false negative, respectively. It is common for the performance experiments of object proposal methods to present the relationship between Recall and Precision, because both Recall and Precision change with the settings of proposed object number.

In this experiment, all of the detection methods take 950 labeled 3D scenes, which contains 53 object categories, as the training material, and 50 raw point cloud data as the test set. Fig. 3.7 shows the comparison of ODLN and representative existing approaches, e.g. Sliding Shapes [82], SVM method [57] trained on Point Cloud & 3D Model Dataset and [57] trained on only Point Cloud. To demonstrate the adaptability for disaster scenarios, there are two versions of dataset prepared for the experiment: one normal dataset and its variant in which the brightness of RGB images is turned down and a part of depth information is removed manually. The first row in Fig. 3.7 presents the results of the normal dataset, and the second row presents the results of the variant dataset. Five categories are selected in the experiment, i.e., chair, bed, sofa, table and toilet.

As shown in Fig. 3.7, ODLN method performs better in the variant dataset. It is mainly because the author desires to design a method not relying on instable RGB data from the

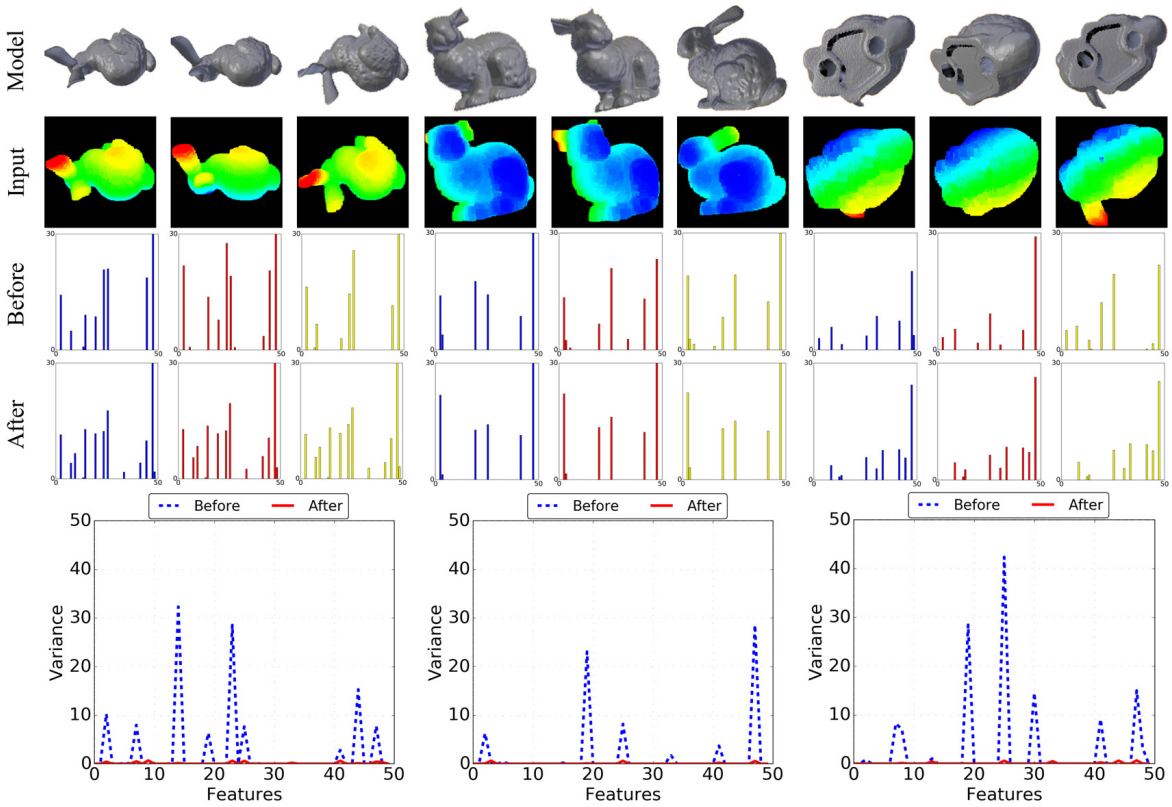
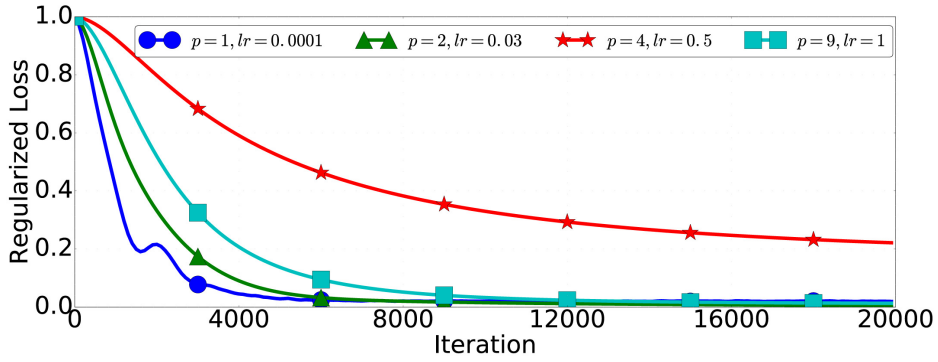


Fig. 3.8 Results of View-invariance Experiment.

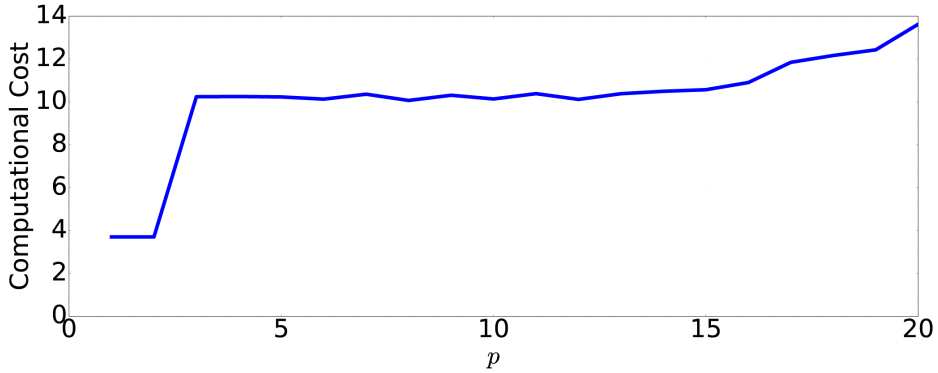
beginning, and the proposed ODLN is not affected by the missing of RGB data. However, the incomplete depth data do give some impact on its performance. But still, it can outperform others in these difficult situations. Of course, ODLN does not show much advantage in the normal dataset, especially compared with Sliding Shapes method. But in my opinion, the normal dataset is too idealized and have little practical significance, considering the disaster scenarios.

3.4.2 View-invariance Test

A demonstrating test is conducted to validate the effect of VI Module in VTCN model. As shown in Fig. 3.8, nine views of bunny model are selected in this experiment. They belong to three different view-subclasses. Therefore, there will be nine features extracted from these views, and each set of three should be similar. The first row of Fig. 3.8 shows each scan view of the bunny model. The second row is the 2D representation of obtained 3D data, which is also the input of VTCN model. The third and fourth row present their features before and after the process of VI Module, respectively. The features are sampled out from the 4096 outputs of VTCN's last layer. It can be seen that, without VI Module, the features looks



(a)



(b)

Fig. 3.9 Results of p Value Selection Experiment. (a) The regularized loss curves with different settings of p and learning rate. (b) The computational cost of different p values.

very different even in a same view-subclass, while with VI Module, views in one subclass become consistent and generate fairly alike features. To further figure out the differences, their variance is calculated using the following formula.

$$\sigma = (F_a - F_b)^T (F_a - F_b) \tag{3.39}$$

The last row of Fig. 3.8 shows the variance curves of the features before and after the process of VI Module. Compared with the blue dotted line, the red one is smooth and steady, which gives a good proof to the proposed view-invariance method.

3.4.3 p Value Selection

A set of experiment is designed to find the appropriate p value in the Minkowski distance for the loss function of VI Module. As discussed in 3.3.3, the setting of p value is very important for the training efficiency. To clarify the relationship between them, several tests

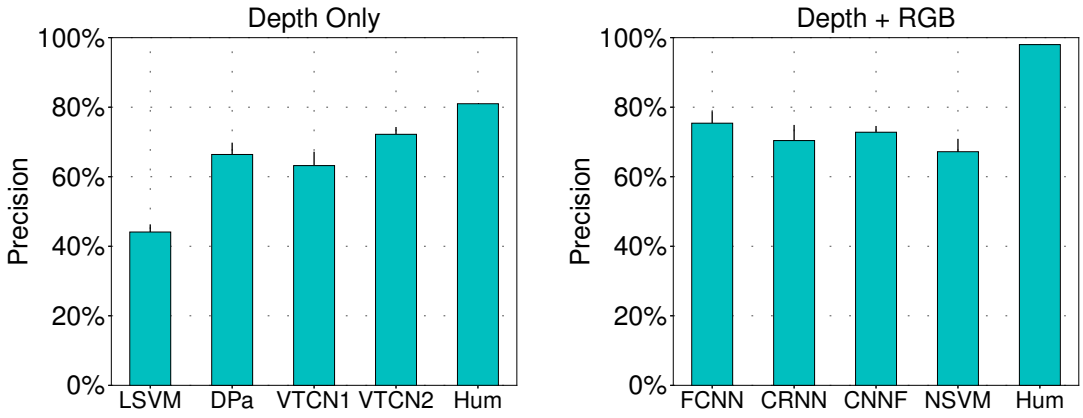


Fig. 3.10 Comparison Results of Classification Experiments. (a) Input data only contains depth information. (b) Input data contains both depth and RGB information.

are conducted using different settings. As expected, their results follow a specific rule. Fig. 3.9(a) presents four representative curves. A regularized loss is adopted in this experiment to measure the learning speed with different settings fairly.

$$J_{i,\text{reg}} = \left| \frac{J_i^{(1)}}{J_i^{(1)}}, \frac{J_i^{(2)}}{J_i^{(1)}}, \cdots \right| \tag{3.40}$$

where i represents the identifier number of each experiment with different p values. It can be seen, with the increase of p value, the learning process slows down significantly even with a larger learning rate. However, on the other hand, they are also more stable with a larger p when close to the convergent point.

Fig. 3.9(b) shows the relationship between computational cost and p values. The computational cost is defined as the complexity of calculation. Several unit operations are added in optimization algorithm to give out a uniform measure for estimated computational cost, whose value is defined in 0$\sim$50. As shown in Fig. 3.9(b), the estimated cost increases with the p value, but not changes drastically as supposed. Therefore, the selection of a larger p is feasible because of the acceptable extra computational cost.

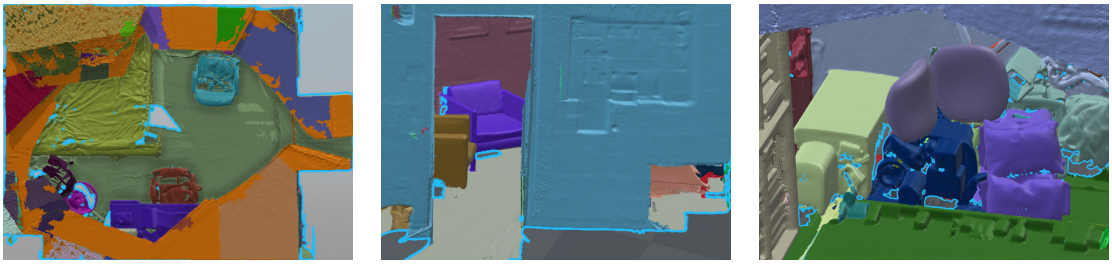
The results in this section are consistent with the author's theories and give us a helpful guide to select p values. A mathematical proof is presented in the appendix, in order to find the precise relationship between p value and training efficiency, and guarantee the adaptability of the proposed system in various datasets.

3.4.4 Classification Performance of VTCN

A comparison experiment is performed to evaluate the classification performance of the proposed VTCN with several state-of-the-art methods. In this experiment, there are 53 categories of 3D labeled objects in the training and testing dataset. For each category, there are 100 instances in the training set, and 10 instances in the testing set. The results are shown in Fig. 3.10. Two set of experiments are designed. Although one same dataset is adopted in all experiments, one set is conducted without RGB information. The left figure presents the performance of depth-only methods, including Linear SVM [41], DPano[78], and the proposed VTCN method; the right one shows the RGBD methods, i.e., Fus-CNN [16], CNN-RNN [80], CNN-Features [75], and Kernel SVM [41]. The proposed VTCN method is tested without VI Module (VTCN1) and with VI Module (VTCN2), respectively, in order to present the performance difference between with and without the view-invariant ability. For better understanding, a manual classification test is also performed. It can be seen that the proposed method improves a lot with the view-invariant ability of VI Module, and outperforms other depth only methods significantly. Although with similar view-invariant feature, VTCN can still outperform DPano method. However, different from the proposed approach, DPano adopts a row-wise max-pooling layer to change the low layer features extracted by the convolutional layers, which needs a semi-panoramic view of the objects. This solution can lead to serious problems in some actual scenes, which will be discussed in Section 3.4.5. But in this test, DPano has been given the necessary views to perform the recognition. Still, its precision is lower than the proposed methods, because VTCN further refines the object category using the view-subclass algorithm. The performance of VTCN is slightly inferior to RGBD methods, however, the author believes, the actual conditions of disaster scenarios do limit the acquisition of RGB data. It is worthwhile to leave out the unstable color data and work on the 3D information for a scene understanding method, which is more robust and adaptable.

3.4.5 Disaster Adaptability

Although the above evaluations have indicated the performance of four different approaches and clearly demonstrate the advantage of the proposed method in common scene understanding tasks, the author finds it is necessary to conduct an additional experiment for the disaster scenarios, because the extreme conditions of the disaster scenes may be a huge challenge to the perception methods. In order to simulate the disaster scenarios, the author works out a new dataset consisting of more than 300 scenes. These scenes are selected from the adopted datasets, which have been introduced in Section 3.4. All of these images are made highly



(a) (b) (c)

Fig. 3.11 Simulated Disaster Scenarios for System Evaluation.

disordered and broken, and only contains the depth information, as shown in Fig. 3.11, which is very similar to some dark, extreme disaster situations.

The author adopts the complete scene understanding system, including the trained ODLN and VTCN presented in Section 3.4.1 and 3.4.4, without extra training process. The new dataset is merely used as the test data. The objective is to evaluate the system performance in simulated disaster scenes. The system will first use ODLN to find possible objects, and then use the VTCN to recognize them. The classification precisions are recorded as the final result. With VI Module, the proposed system can achieve 55.61% in this test, while without VI module, it is 47.71%. As comparison, the author also combines ODLN, VTCN with several other approaches for the same test data. The ODLN&Linear SVM[41] is 30.39%, the ODLN&DPano[78] is 37.16%, the SVM[57]&VTCN(with VI module) is 18.33%. Due to the difficult situation, the results are generally not very high. However, the combination of VTCN, ODLN and VI module does bring a significant improvement to the precision, which can give a boost to the development of disaster robotic. In addition, since the system cannot obtain the complete depth information of the objects, DPano method has no sufficient view to perform panoramic recognition, therefore, its performance on this dataset is not very good, if compared to its precision in Section 3.4.4.

3.4.6 Robotic Simulation

Robot Operating System (ROS) [66] is adopted as the simulation environment in this experiment. As shown in Fig. 3.12, a mobile robot moves along a path and takes three scans. The understanding results are generated by a system using the proposed method. To validate the feasibility of the proposed method in dark scenes, only depth data is imported into the system. It can be seen, the proposed method is able to output correct labels for most objects. In Fig. 3.12(b), the proposed system successfully recognizes the table with a confidence of 0.849, and the lamp with 0.424. It also generates several candidates with different confidences. For

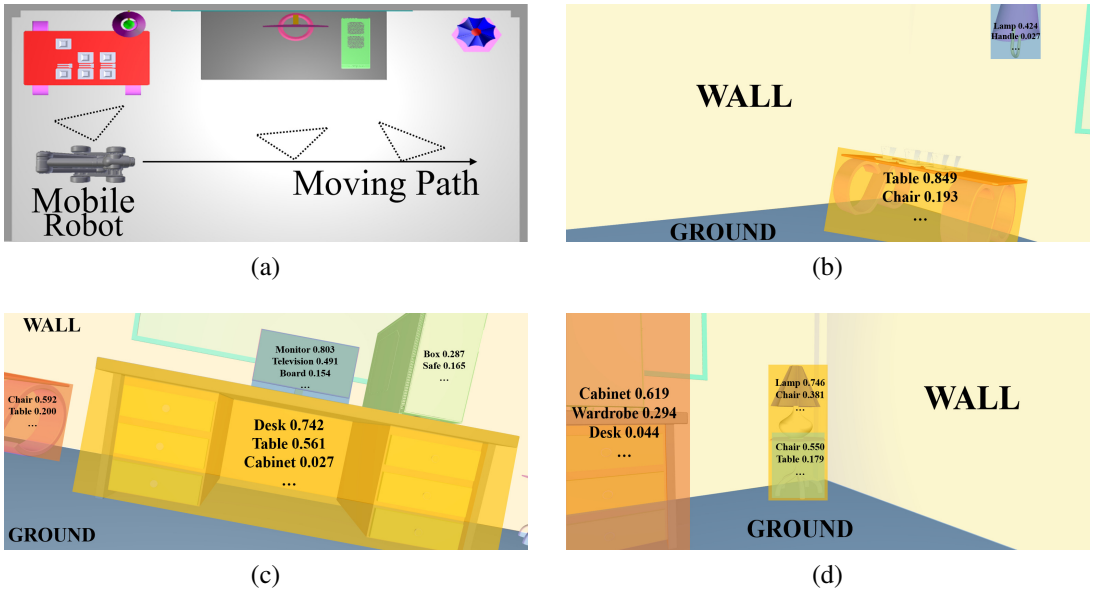


Fig. 3.12 Robotic Scene Understanding Simulation.

example, in Fig. 3.12(c), the system gives some labels to the desk in yellow zone including desk, table, cabinet, etc. Among them, desk is with the highest confidence, and becomes the final output label. The total precision is 91.0% with 10 runs. The results demonstrate that the proposed method is stable and robust, and can be well applied to complicated indoor scenes. In addition, the author also uses some existing approaches to perform the same task under the same condition. In fact, this is a similar test, compared to the one in Section 3.4.5, with more ordered environment and lower difficulty. The results are listed below. The ODLN&Linear SVM[41] is 35.62%, the ODLN&DPano[78] is 67.64%, the SVM[57]&VTCN(with VI module) is 41.81%.

Chapter 4

Artificial Intelligence for Internet-of-Vehicles

The autonomous vehicle, as an emerging and rapidly growing field, has received extensive attention for its futuristic driving experiences. Although the fast developing Internet-of-Vehicles and AI methods have given a huge boost to self-driving research, existing autonomous driving vehicles do meet with several avoidable accidents during their road testings. The major cause is the misunderstanding between self-driving systems and human drivers. To solve this problem, a new decision-making system is proposed in this work for autonomous vehicles. As shown in Fig. 4.1, the proposed system prefers to thinking like a human, while traditional approaches make decisions according to the software settings.

In order to simplify the complexity of the decision-making system, a scene understanding subsystem is adopted to generate the abstractions from the raw input. This system only takes the 3D data from the light detection and ranging (LiDAR) sensor, which is able to generate stable and robust images in both light and dark environments. With the work experience in Chapter 3, a LiDAR-only scene understanding and abstraction method is used as a component of the proposed system. In this method, RGB data is only used in the detection of nearby lane markings, which can always be covered by the headlight. And only depth information is needed in the detection and classification of other objects, e.g. cars, buses, trunks, etc.

In this chapter, the author focuses on the autonomous decision-making system for the Internet-of-Vehicles and implement it on the basis of a deep learning model.

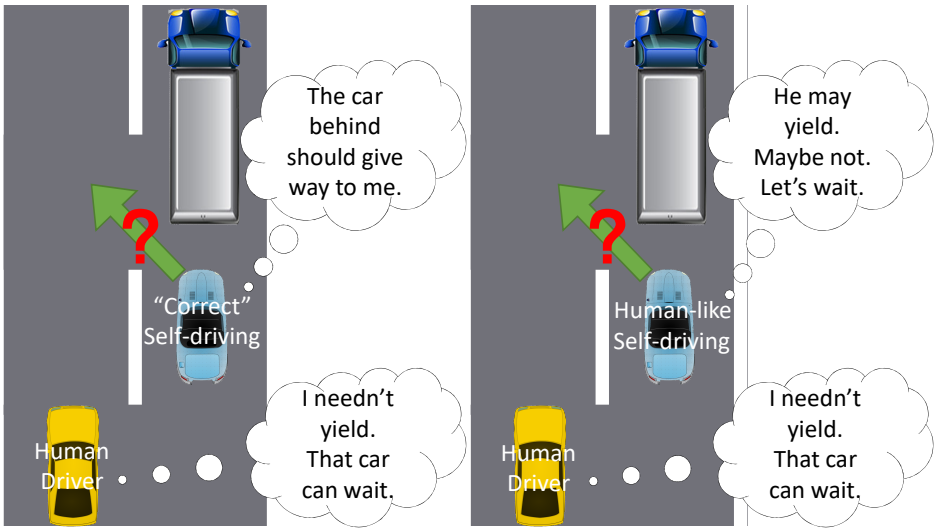


Fig. 4.1 Differences between traditional "correct" self-driving and the proposed human-like driving.

4.1 System Overview

4.1.1 Problem Definition and System Framework

Given a collection of road scene images $\mathscr{X}$, which are captured by on-board sensors during driving and consist all necessary information regarding the road condition at a certain moment, the goal of the proposed self-driving system can be described as below. For each input scene $\mathscr{X}$, the perception subsystem will recognize as many as possible vehicles on the road and give them accurate labels. Then the decision-making subsystem should give out some advice $\mathscr{D}_h$ regarding the speed and steering, according to the abstraction $\mathscr{A}$ of current road condition, and, ultimately, generate the precise driving command d_{fin}, considering the security policy.

We introduce the system framework in Fig. 4.2 to give a brief description of the proposed system. As can be seen, the vehicles use the CNN models to respectively perform the detection and recognition task, then the abstraction results will be transferred to the decision-making system. As for the markings, they are processed by several simple edge detection methods. For brevity, they are omitted in the dissertation.

After the perception process, the decision-making network calculates the driving decisions based on these abstractions. The decision-making model is the focus of this research, and it will be given a detailed introduction in Section 4.2. In addition, an influence analysis module is included in the model to clarify the specific relationship between the road conditions and the output decision.

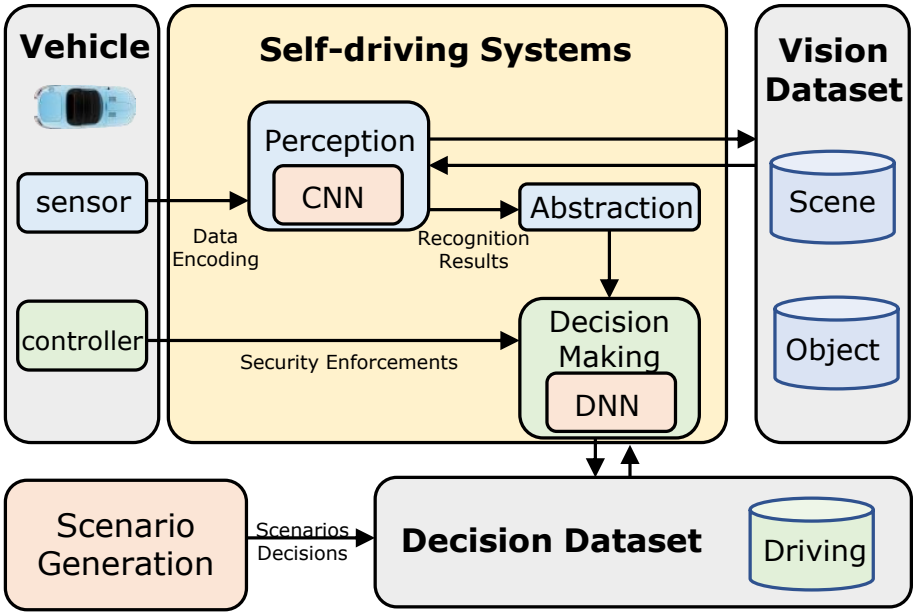


Fig. 4.2 Framework of the proposed self-driving system.

4.1.2 3D Perception and Abstraction

One major characteristic of the abstraction calculation approaches is that the abstraction of the road condition is generated before making driving decisions. In this procedure, a robust and efficient 3D perception method is essential to obtain the accurate understanding of the input images, which is captured by the sensors equipped on the self-driving vehicles. This task is also called scene understanding, one hot topic in computer vision area. However, as mentioned above, the traditional methods struggle in the perception and abstraction of road images, due to their dependences on unstable RGB data. We propose a specially-designed perception method to understand the input scenes, in order to improve the performance of vehicle perception, and work out an abstraction approach for simplified road representation. The road data comes from two sources: the famous vision dataset for autonomous driving KITTI [59] and the open racing car simulator (TORCS). KITTI provides a great number of real-world road scenes, which is captured by a real car equipped with a laser scanner and a GPS localization system. And TORCS is a fully customizable car racing simulation environment, where lots of cars and road types are available. Both KITTI and TORCS are adopted in the benchmarking and simulation tests.

In short, the main purpose of the proposed perception and abstraction method is to understand the captured road images and generate a simplified representation for the following decision-making process. As shown in Fig. 4.3, the ego car in TORCS is equipped with depth sensors and can send the depth images to the scene understanding network in real time, and

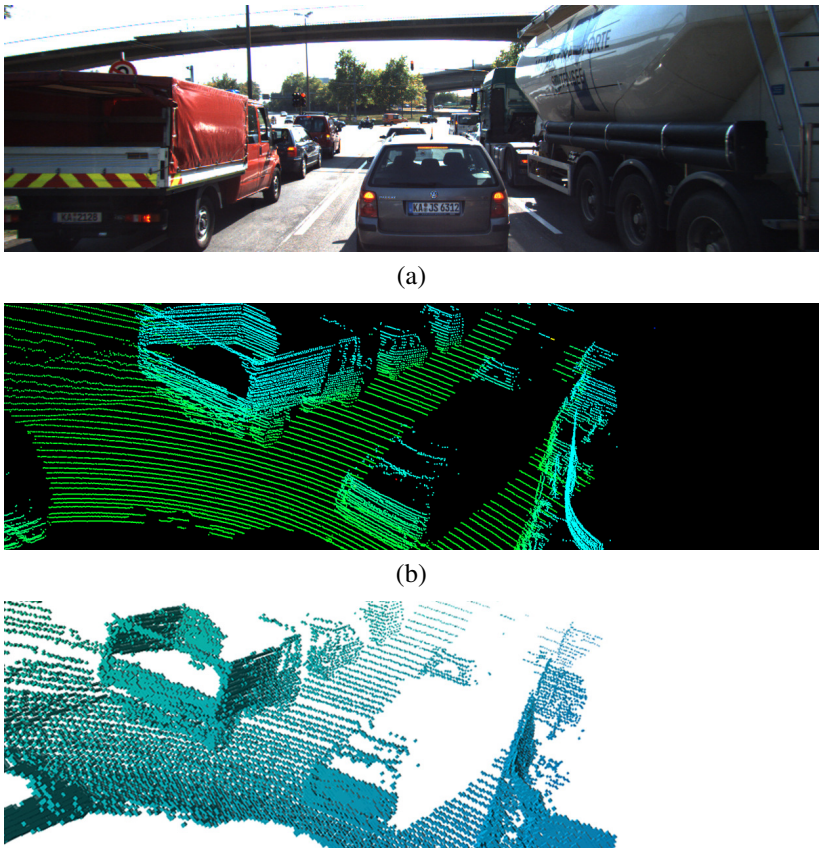
(a) (b)

Fig. 4.3 The perception and abstraction of road condition. (a) Road simulation using the open racing car simulator (TORCS). (b) Generated abstraction using the proposed road perception method.

then the current road condition is abstracted by the proposed scene understanding method for the planar representation, which can be directly imported into the decision-making model. The perception and abstraction method is detailed below.

We propose the road condition understanding network (RCUN) to understand the road scenes, and adopt the famous Alex model [40] as the recognition network, and its modified version, which is compatible with 3D input data, as the vehicle proposal network. The major difference between RCUN and other perception methods is that the author utilizes different data encodings in these two models. The original raw data generated by the depth sensors is in the point cloud format, which is shown in Fig. 4.4(b). There are two ways to process these data. The first one is to map these data into RGB space, using the Point Cloud Library (PCL) [71], and obtain the 2D colored representation. The other one is the 3D representation, in which the point cloud data is transformed into volumetric encoding, which could be imported into computational models with 3D spatial information preserved, using Octomap [29], as shown in Fig. 4.4(c). The reason for this design is based on the following facts.

These two approaches have their advantages and disadvantages. One huge superiority of 2D representation is that there have been extensive research works regarding 2D image descriptors and feature engineering, which is of great help in the designing of 2D-encoded scene understanding models. And also, tremendous 2D image datasets have been published, and even more well-trained models relying on these datasets have been made available, on whose basis the author can establish own models. According to [102], low-layer features are not specific to one particular task or dataset and can be applied to many other networks. Similar to the physiological visual mechanism of the human, neurons of lower layers merely



(a)

(b)

(c)

Fig. 4.4 Road scene encoding and representation [59]. (a) The original scene captured by the RGB camera. (b) The point cloud captured by the depth sensor. (c) 3D encoding.

deal with the most specific and simplest tasks, which are usually homogeneous between many different applications. Therefore, the existing models are adopted as the basis of the proposed model with further fine-tuning, learning abstract features in high-layers once again and improving the training speed and network precision of the newly-designed model. On the contrary, works on 3D images are yet to be developed and improved. Few large-scale 3D image databases or effective trained models have been proposed. Another advantage of 2D representation is its high resolution. Due to some restrictions in principle and implementation, the volumetric representation must be encoded in a lower resolution (0.02 m in Fig. 4.4(c)), whereas 2D representations can preserve image details well. On the other hand, the volumetric representation also has some advantages. Unlike 2D encoding, 3D representation can indicate the 3D spatial information of the original scene, which is highly valuable for some applications. Therefore, the author designs the system with two data encoding types, which are respectively for the 3D detection and recognition.

After training, these two networks are able to perform the proposal and recognition tasks.

4.2 Decision Making

After obtaining the abstraction of road conditions, the system can work on the corresponding driving decisions. Compared to behavior reflex approaches, which directly map the input images to several pre-defined driving commands, the proposed method, using the abstractions instead of original images to calculate reactions, is much simplified in the inherent logic and structure, and as a result, is much more trainable and can achieve higher network precisions.

The generated abstraction is shown in Fig. 4.3(b), however, the abstraction data is still a bit complicated to be inputted into the decision-making network, due to the massive possibilities of relative vehicle positions. For further simplification, the author defines lots of grids all over the necessary road area, and then each vehicle can be subjected to one grid, which can be represented by one numerical value. This method gives a huge convenience for data encoding and network input because only a fixed number of girds exist in the interested zone of the road. We give these grids two attributes to reflect the type and speed of the vehicle on the road and adopt the girds as the input matrix of the proposed network. More precisely, each grid has two attributes, including one discrete attribute value, representing the specific vehicle type including sedan, bus, trunk, ego vehicle, none, unobservable, etc., and one continuous value, representing the vehicle current speed. If there is no vehicle in a grid, then its speed attribute is set to zero. As shown in Fig. 4.7(a), the blue car is the ego vehicle, and all the grids are labeled with a specific vehicle type and corresponding vehicle speed. Notably, although the on-board sensors improve a lot in the visual performance nowadays, it is not possible to obtain accurate road conditions in all grids. Especially, the vehicle perception system may fail to output an exact result, due to the occlusion. In this situation, the unknown or indefinite grids are labeled with "unobservable" and painted blue in the figure.

In this section, the author will detail the proposed decision-making system, and also introduce an analysis method to clarify the relationship between the road factors and the final decision.

4.2.1 Decision Model

We propose a six-layer decision-making network (DMN) to learn human decision-making behaviors, as shown in Fig 4.5. The input is the generated abstractions of road conditions. And the main objective is to make driving decisions based on these input data. The proposed model contains five hidden layers and one output layer. There are 1028 neurons in each of the first two layers, and 512 in each of the following three layers. This is a simple design because the author focuses on the data engineering and network learning process. We believe,

using an effective training scheme, the proposed method can extract sufficient information from the well-processed input data, even with a simple model. Due to the superior feature extraction ability of deep model, the author demonstrates that six layers are enough to obtain the relationship between road condition and driving decisions.

The distinctiveness is that the author designs the DMN as a two-part structure. Two individual sub-networks are adopted to respectively calculate the speed and steering commands. Since the low-level features are usually similar in neural networks, the first three fully-connected layers are shared between two sub-networks for training efficiency. There are two more layers in each sub-network to extract the high-level features, which is highly correlated to the target of each sub-network.

There are n input abstractions $\mathscr{A} = \{a_1, a_2, ..., a_n\}$ in DMN, with their corresponding labels a_i, including the speed labels $\mathscr{P} = \{p_{a_1}, p_{a_2}, ..., p_{a_n}\}$ and steering labels $\mathscr{T} = \{t_{a_1}, t_{a_2}, ..., t_{a_n}\}$. $t \in \{t_1^*, t_2^*, t_e^*\}$ are the possible labels, representing left and right lane changing or keeping the current lane, and p should be real values representing the possible vehicle speed. $h_{\text{spd}}(a_i)$ and $h_{\text{str}}(a_i)$ are the results of each layer.

According to the multi-task loss introduced in [83] and [20], the loss function is defined as follows.

$$L = \lambda_1 L_{\text{spd}} + \lambda_2 L_{\text{str}} \tag{4.1}$$

where the loss functions, including L_{spd} and L_{str}, are adopted for different goals, and their parameters, including λ_1 and λ_2, are pre-defined to give them respective weights according to their importance.

L_{spd} is utilized to calculate the acceleration or deceleration commands and it is a regression of desired vehicle speed. Using the smooth L_1 loss [20], L_{spd} can be defined as

$$L_{\text{spd}} = \frac{1}{n} \sum_{i=1}^{n} \text{smooth}_{L_1}(h_{\text{spd}}(a_i) - p_{a_i}) \tag{4.2}$$

L_{str} is calculated using the softmax function. It is adopted to compute the steering commands, i.e. changing lanes or not. It can be expressed as

$$L_{\text{str}} = -\frac{1}{n} \left[\sum_{i=1}^{n} \sum_{j=1}^{3} \mathbf{1}\{y_{a_i} = y_j^*\} \log \frac{e^{h_{\text{str}}^{(L)}(a_i)}}{\sum_{\phi=1}^{3} e^{h_{\text{str}}^{(\phi)}(a_i)}} \right] \tag{4.3}$$

where $h_{\text{str}}^{(j)}(a_i)$ varies from 0 to 1, according to the momentum to change to different lanes.

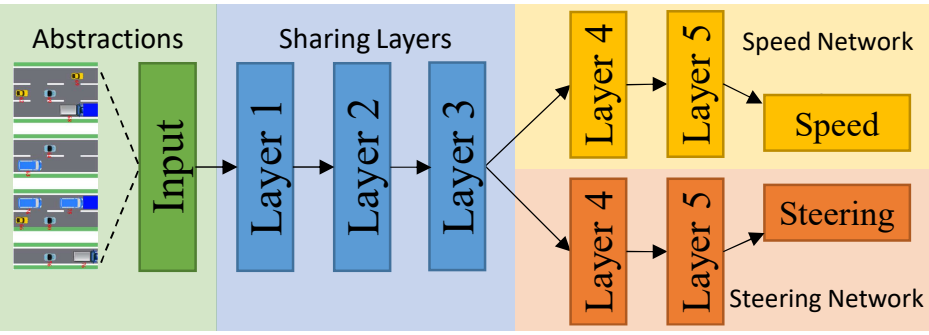


Fig. 4.5 Architecture of the decision-making network.

Algorithm 2: Decision-making Logic

Input: Current Road Condition c
Output: Driving Decisions d_{fin}

```
/* Make decisions constantly.                                    */
while IsDriving do
    A = GenerateAbstractions(c)
    D_h = ComputeAdvices(A)
    D_s = CollisionAvoidance(A)
    /* Examine D_h in confidence order                           */
    for d_h^spd, d_h^str ∈ D_h do
        d_fin^spd = d_h^spd + d_s^spd,   d_fin^str = d_h^str + d_s^str
        d_fin = (d_fin^spd, d_fin^str)
        d_fin = RegulationsChecking(d_fin)
        /* Final Security Check.                                 */
        if isSafeDriving(d_fin) = True then
            output(d_fin)
            break
    KeepSafeDriving()
```

Also, the SGD strategy is adopted in this work to perform network training and the output results of the proposed network are regarded as driving advice $\mathscr{D}_h$ for the final decision-making strategy, which is detailed in the next section.

4.2.2 Decision-making Strategy

The whole process of the proposed decision-making system is presented in Algorithm 2. As mentioned above, at first, the RCUN model generates abstractions of the inputted images regarding current road conditions. And then DMN model can calculate and output the

human-like decisions. Although these decisions are already very reasonable, they are still not yet fully adaptable for actual driving situations, due to the security reasons. We combine the driving advice generated by DMN with a safety enforcement method, namely the repulsive potential field (RPF) [100]. RPF is a famous method used for robotic path planning. Its main objective is to control the robots to safely reach destinations while avoiding collisions. RPF mainly relies on the repulsive force of possible obstacles, which is very suitable for driving safety control. In detail, the author regards all other vehicles and both road edges as the obstacles and then calculate their repulsive forces. The repulsive potential formula can be expressed as

$$U_{\text{rep}}(o) = \begin{cases} \frac{1}{2}\eta(\frac{1}{d(o)} - \frac{1}{d}), & d(o) \leq d \\ 0 \quad , & d(o) > d \end{cases} \tag{4.4}$$

where o represents the obstacle object, $d(o)$ is the distance between the obstacle and ego vehicle, η is a positive scaling factor. And, d is a pre-defined constant value representing for the maximum obstacle distance. Any obstacles beyond the distance d are neglected. Having the repulsive potential formula, the repulsive force can be expressed as

$$\begin{aligned} F_{\text{rep}}(o) &= -\nabla U_{\text{rep}}(o) \\ &= \eta(\frac{1}{d(o)} - \frac{1}{d})(\frac{1}{d^2(o)})\nabla d(o) \\ &= \eta(\frac{1}{d(o)} - \frac{1}{d})(\frac{1}{d^2(o)})\frac{o-e}{\parallel o-e \parallel} \end{aligned} \tag{4.5}$$

As shown in Fig 4.6, the objects in red color are regarded as the obstacles, including two edges and two vehicles in front of the ego car. The orange lines represent the repulsive force, and the yellow coordinate represents the driving decisions. It can be seen that the repulsive force and the ego car form an angle θ, and the forces are respectively exerted on the steering and speed axis. Then, the RPF security enforcement can be written as

$$d_s^{spd} = -\sum_{o}^{\text{objects}} \parallel F_{\text{rep}}(o) \parallel sin(\theta_o) \tag{4.6}$$

$$d_s^{str} = -\sum_{o}^{\text{objects}} \parallel F_{\text{rep}}(o) \parallel cos(\theta_o) \tag{4.7}$$

In addition, since the steering command is a discrete value, several thresholds are set to discretize d_s^{str}.

We design the final decision-making equation as below.

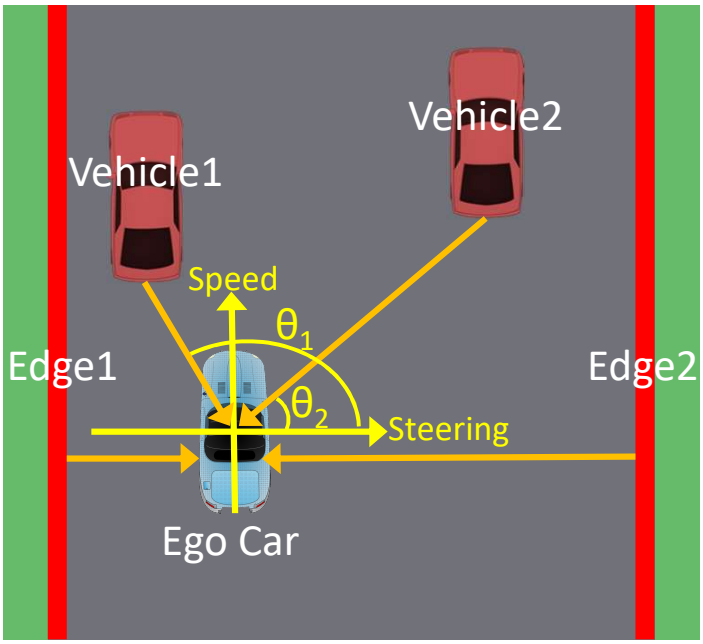


Fig. 4.6 Repulsive force and security enforcement.

$$\begin{aligned} d_{fin} &= \omega_1 d_h + \omega_2 d_s \\ s.t. \quad & \omega_1 + \omega_2 = 1 \end{aligned} \tag{4.8}$$

where d_{fin} is the calculated final decision, d_h and d_s represent the human-like driving advice and safety enforcement, respectively, and ω_1, ω_2 are the custom weights. In addition,

$$\begin{aligned} d_h &= [d_h^{spd}, d_h^{str}] \\ d_s &= [d_s^{spd}, d_s^{str}] \end{aligned} \tag{4.9}$$

are the output of DMN and RPF. As shown in Algorithm 2, after the final decision is generated, it is regularized to follow the traffic regulations, for example, driving along the lane, and examined with some final security check, which is defined according to local situations. Ultimately, the system can give out a safe and human-like driving decision.

4.2.3 Network Training Scheme

As the proposed decision-making system is based on a machine learning model, an important problem is how to obtain sufficient learning materials for the underlying neural network. One traditional way is, obviously, to record actual human driving behaviors, using either a real car

Algorithm 3: Scenarios Generation

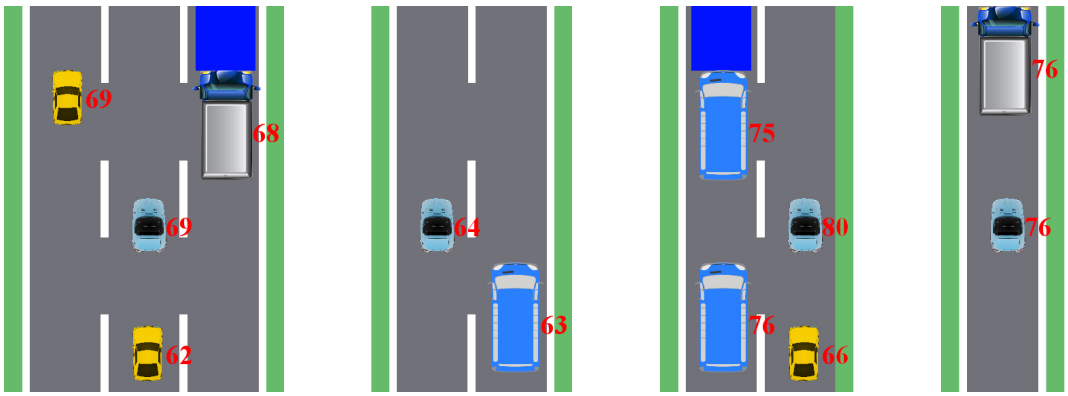
Input: Desired Scenarios Number n; Vehicle Type (including "none") $\mathscr{V}$ and their Propotions $\mathscr{P}_{\mathscr{V}}$; Vehicle Type $\mathscr{R}$, their Propotions $\mathscr{P}_{\mathscr{R}}$ and Speed Distribution in each road type $\mathscr{D}_{\mathscr{S}}$

Output: Simulation Scenarios $\mathscr{S}$

```
scenario_list = null
while length(scenario_list) < n do
    /* Generate a road r.                                          */
    r = RandomRoad(R, P_R)
    /* Try all available positions.                                */
    for grid ∈ r do
        /* Generate a new vehicle v.                               */
        v = RandomVehicle(V, P_V)
        if ExistCarInFront(v) is False then
            /* Assign vehicle speed v.                             */
            v.speed = RandomSpeed(v, r, D_S)
        else
            /* Adjust v according to the speed of vehicles in front.
            */
            v.speed = SpeedAdjust(r)
        r.append([v,grid])
    scenario_list.append(r)
return scenario_list
```

or a custom gaming simulation environment. However, it is very inefficient to conduct such data acquisition task, which usually needs several months to get adequate road scenarios and driving decisions, even for a small-scale dataset.

We implement a novel generation strategy of training material to efficiently create driving scenarios for human reaction recording, as shown in Algorithm 3. At first, the road type is randomly selected from several templates according to the pre-defined occurrence possibilities. Then the vehicles are also generated in a similar way and get assigned to random grids of the newly generated road. Each vehicle will be set with a specific speed value, and be adjusted to be as realistic as possible. Some generated scenarios are shown in Fig. 4.7 as an example. The type and relative position of the vehicles are all randomly selected, on the basis of some common situations, traffic rules, driving habits, etc. The last step is to import these scenarios into some driving environments, such as the TORCS, and record the driving decisions of test drivers. To improve the sense of reality and interactivity, some peripheral equipment can be used to emulate the actual driving experience, e.g. steering



(a) (b) (c) (d)

Fig. 4.7 Generated driving scenarios for network training.

wheel and pedals. Before each scenario, a brief buffer time is available for the test drivers to get familiar with the current road condition.

Compared to some commonly used data recording methods, the proposed scheme has a significant advantage at the efficiency, and more importantly, the representativeness of the generated scenarios. In fact, there are a large number of unnecessary testing scenarios in the continuous driving process. Therefore, the proposed scheme can avoid this problem by generating discontinuous scenarios.

4.2.4 Influence Analysis

We find it essential to figure out why the proposed neural network can make human-like reactions to various road conditions. Which parts of the input abstractions play a key role in the final decision? What is the network focused on among all the input values including the vehicle types and speed in every road grid? Indeed, it is not easy to accurately visualize the specific influences of the input units in a neural network. In this section, the author implements an ingeniously-designed visualization method to analyze the mechanisms behind the proposed decision-making network. There are two major benefits of this analysis method. The first one is that, with an intuitive heatmap, self-driving researchers can easily find the specific areas and information which the decision-making system is really interested in. And as a result, researchers can find ways to improve the accuracy or clearness of these valuable data, e.g., increasing the resolution of the back depth sensor, refining the vehicle category of the road perception system, improving the accuracy of velometer, etc. The other benefit is, the generated heatmap is an important criterion reflecting whether the decision-making network is overfitting or not. Since the network architecture is defined according to subjective

Algorithm 4: Influence Analysis

Input: Trained Network N, Specific Road Abstraction $\mathscr{A}$
Output: Visualized Unit Influences

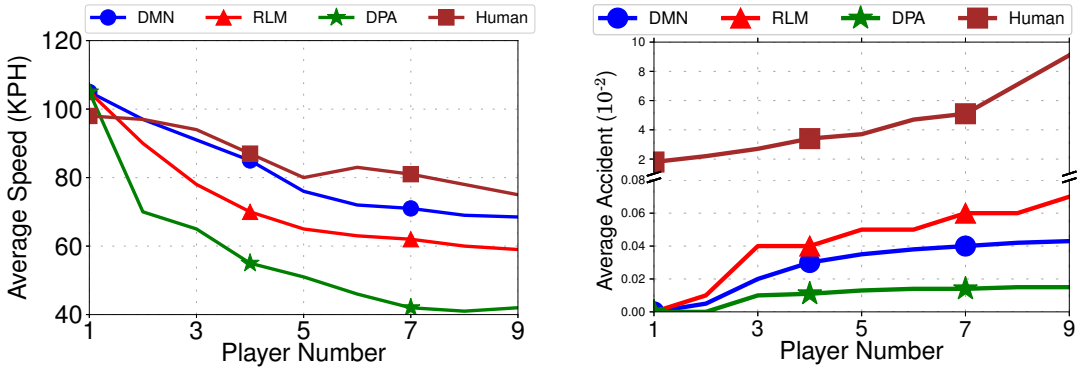
```
/* Check all the unit values in both two input channels.          */
for att ∈ [vehicle_type, vehicle_speed] do
    influences_vector = null;
    original_output = h^N(𝒜)
    for u ∈ 𝒜_att do
        /* Modify the unit value.                                 */
        if att is vehicle_type then
            𝒜_new = ChangeType(𝒜, u)
        else
            𝒜_new = ChangeSpeed(𝒜, u)
        new_output = h^N(𝒜_new)
        /* Compute the impact of the changed unit value.          */
        influence = new_output − original_output
        influence = Normalize(influence)
        influences_vector ← influence
    /* Generate the visualization.                                */
    Visualize(influences_vector)
```

experience, it is possible that the deep network is designed with too many layers or neurons. In this situation, if no enough training material is available, the network will suffer the overfitting problem. Although an overfitted network performs well from the perception of training loss, it may easily fail in some other road conditions. It is very difficult to judge the overfitted networks using simple precision tests. However, the overfitted networks show inexplicable and meaningless heatmaps in most cases. By examining the generated influence maps carefully, researchers can find some symptoms of the overfitting problem. This algorithm is introduced below.

In a trained network, the influence of one input u, which is a single value in the input vector, to the output of layer θ is presented below

$$h_\theta^N(u) = \begin{cases} W_\theta h_\theta(u) + b_\theta, & \theta \geq 2 \\ W_\theta u + b_\theta \quad , & \theta = 1 \end{cases} \tag{4.10}$$

Notice that, with a fixed W_θ and b_θ, the output of every layer in the neural network, including the final output, is only related to the input u. Therefore, the contribution of each input values can be inferred using Algorithm 4. In the beginning, each vector unit of both two



(a) (b)

Fig. 4.8 Performance evaluation in driving simulation. (a) Comparison results of average driving speed. (b) Comparison results of average driving accidents.

attributes, i.e. grid type and vehicle speed, will be checked for the influence calculation. Its value is modified with a minor adjustment, and the difference between the original and new output results is calculated to reflect the contribution of this grid. The numerical results are normalized and transformed into heatmaps.

An experiment is conducted in Section 4.3.2 to demonstrate this analysis method.

4.3 Performance Evaluation

Several experiments are conducted in this section to evaluate the performance of the proposed self-driving system. First, the author carries out a driving simulation to test the abilities of decision-making method. Then a visualization of unit influences is presented, using the strategy described in Section 4.2.4.

4.3.1 Decision-making System Evaluation

The best way to evaluate the performance of driving systems is to conduct the road tests in a simulation environment. As mentioned above, TORCS is a common choice for the testing platform in driving simulations. Therefore, the author also adopts TORCS in this experiment for better comparability. In this section, the focus problem is the rationality and security of generated driving decisions, while the scene understanding ability is not the emphasis. Therefore, the exact abstractions of the road conditions are directly passed to the self-driving methods by the TORCS platform, rather than the scene understanding methods. Then, the

key problem is the experiment design, which should give a comprehensive test on both the advance speed and driving security.

The testing details are introduced below. First, the author selects and modifies an existing roadmap in TORCS, which is a two-lane street with a total length of $15,000$ meters, making clearer lane markers and richer roadside views. Then, an autonomous vehicle, controlled by different self-driving methods, is added to the track as player Ego. Up to eight human players are also included in the same track, so the self-driving vehicle must take other players into account when driving on the road. There are nine test settings with different human numbers for one self-driving method, and ten tests are conducted for each test setting. The vehicle speed and total accident number are recorded during these tests. Fig. 4.8 presents the average results of vehicle speed and accident number. Four driving methods are adopted for player Ego, i.e., the proposed DMN method, the reinforcement learning method (RLM) [100], the direct perception approach (DPA) [9], and manual driving (Human), which is set for better understanding.

It can be seen that, in Fig. 4.8(a), with the increase of player number, all the testing vehicles show slower average speed. Among the self-driving methods, the proposed DMN shows an obvious advantage beyond other two methods, due to its intelligent decision-making ability. The RLM method is implemented by us according to the principle introduced in the original paper. It is a novel design, but does not consider the human driving behaviors and social intelligence. Therefore, it is slightly inferior in this test with human participants. DPA is focused on the driving perception, and only has a simple driving logic, which mainly cares about the security rules. So it does not perform well in the speed testing. Unexpectedly, the increase of player number has little influence on the human driver, possibly because the human drivers have the social intelligence ability and prefer to aggressive driving strategies. Fig. 4.8(b) presents the results of security testing, i.e., the average accident number. Contrary to the speed testing results, DPA outperforms other methods because the security regulations are carried out rigorously in DPA. And the proposed DMN has the second best security performance, due to the safety enforcement and, more importantly, the human-like decision-making ability. RLM also considers the safety requirement, but is outperformed by DMN due to the lack of social intelligence. Because the road conditions are directly sent to self-driving systems, they can easily beat the human driver in this testing, which, of course, is different with the real-life driving environment, where the road conditions must be understood and abstracted by the perception system. However, as a performance evaluation, these results can sufficiently demonstrate the feasibility and stability of the proposed decision-making method.

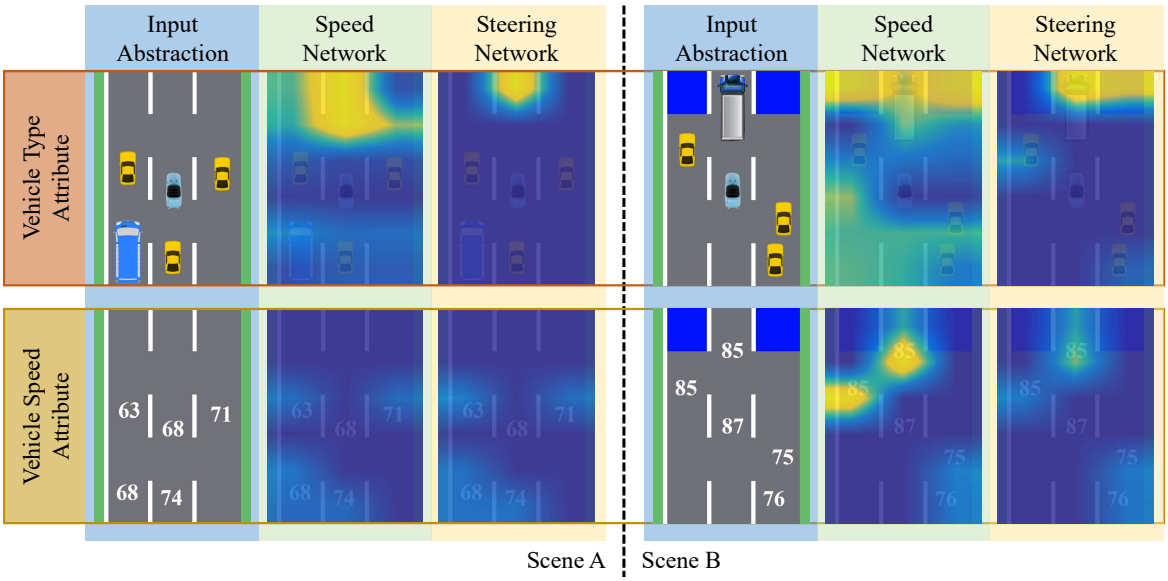


Fig. 4.9 The visualization results of unit influence.

4.3.2 Influence Analysis and Visualization

As it has been demonstrated that the proposed self-driving system is able to perform sound and safe decisions, an additional experiment is conducted to clarify the relationship between the road conditions and the driving decision. Fig. 4.9 shows the visualization results of two randomly generated scenarios. The input abstractions are presented in the first column of Scene A and B in Fig. 4.9, and the second and third column give out the unit contributions to the speed decision network and steering decision network. The output decision of Scene A is to keep the current lane and pick up speed; the decision of Scene B is to deceleration and move away from the front truck. It can be seen that, in Scene A, DMN notices there is an extensive free area in front of ego-car. DMN also judges that the cars in neighbor lanes are unlikely to change lanes, and, therefore, have little impact on the final decision. In Scene B, DMN decides not to change lane because of the unknown area in the top-right corner, which is reflected in the vehicle-type channel in the steering network. Instead, DMN prefers to slow down and keep the speed slightly slower than the front truck.

Interestingly, the visualized unit contributions give us deep insights into the mechanism of the decision-making neural network. We find several interesting results after the unit contributions are transformed into heatmaps. These results are reasonable and explainable. The way DMN considers its decisions is so similar to the human mind that, the author believes, it can achieve more amazing performance with a larger dataset. The experiment results demonstrate that the proposed method does have the ability to perform human-like decisions and adapt well to various road conditions.

Chapter 5

Artificial Intelligence for Internet-of-Energy

Electrical load forecasting is still a challenging open problem due to the complex and variable influences, e.g. weather and time. Although, with the recent development of Internet of Things (IoT) and smart meter technology, people have obtained the ability to record relevant information on a large scale, traditional methods struggle in analyzing such complicated relationships for their limited abilities in handling non-linear data. In this chapter, the author introduces an IoT-based deep learning system to automatically extract features from the captured data, and ultimately, give an accurate estimation of future load value.

The IoT-enabled system is implemented in an urban area of south China, as shown in Fig. 5.1. Smart meters are adopted to record and upload electrical and background data, with specially-designed sensors and IoT-enabled devices. They are deployed in every electricity consumption unit, and share their information with the control center. Ultimately, a large dataset is obtained in seven years from 2010 to 2016, including all conceivable factors. Then the data is used for the network training and influence analysis in this work. As IoT and smart meters both have the ability of bidirectional communication, the recommendations and decision made by the control center can be sent to the demand side. The IoT infrastructure in Fig. 5.1 is an important part of the proposed system. The smart meters can connect to all the IoT-enabled sensors, gadgets and appliances in the electricity unit, for example, the smart homes. The proposed system can deal with all these data, and based on that, perform accurate load forecasting.

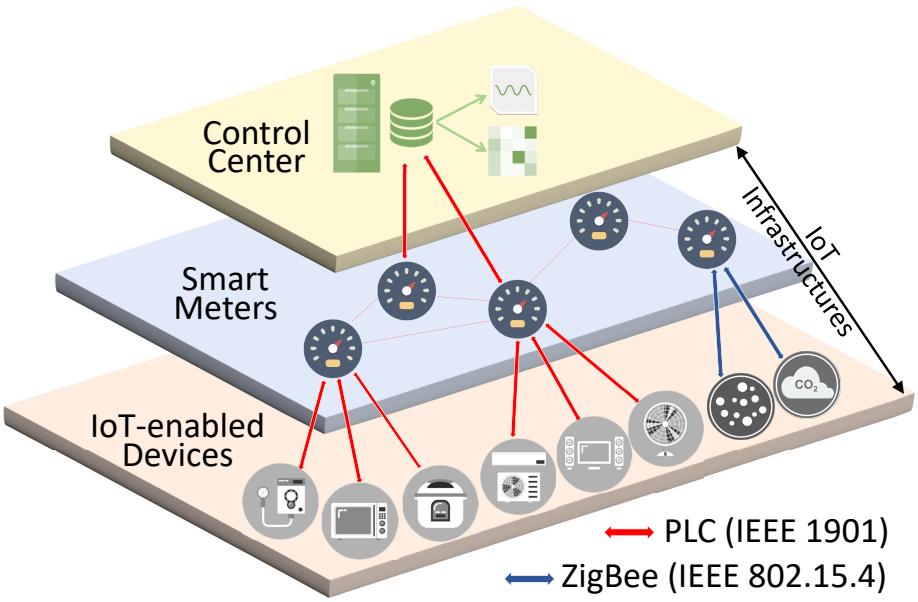


Fig. 5.1 The load forecasting and analysis system based on IoT-enabled sensors and devices.

5.1 Forecasting System: Concept and Design

When starting with the research of electrical load, the author wonders what on earth are the possible influences. And which factors do have a role in the load variation? The author decides to begin with the analysis of historical record, and tries to find some inspirations. Fig. 5.2 presents an electrical load record of a large city in south China. The data is sampled every five minutes, from January 2014 to June 2016. As can be seen, there are some obvious patterns in the load variation. On one hand, these records reflect an annual periodicity. The power load peaks between June and October every year, and hits bottom around February, which has significant seasonal characteristics. On the other hand, there is also an obvious daily periodicity, i.e., the load value keeps high in the daytime, and drastically drops down at night.

Although with some simple data analysis like this, the presented load patterns can lead to a few preliminary conclusions, it is very difficult to truly understand the complex relationship between the power consumption and influence factors. In fact, weather and some other factors play much more parts in electrical load variation, and also in more complicated ways, which is far beyond the capacity of humanity and traditional load forecasting methods. Besides, the author empowers the smart meters with the ability to communicate with other IoT-enabled devices in the system, leading to more extensive input attributes. As shown in Fig. 5.1, the IoT infrastructure is a fundamental component, because it monitors the factors and sends

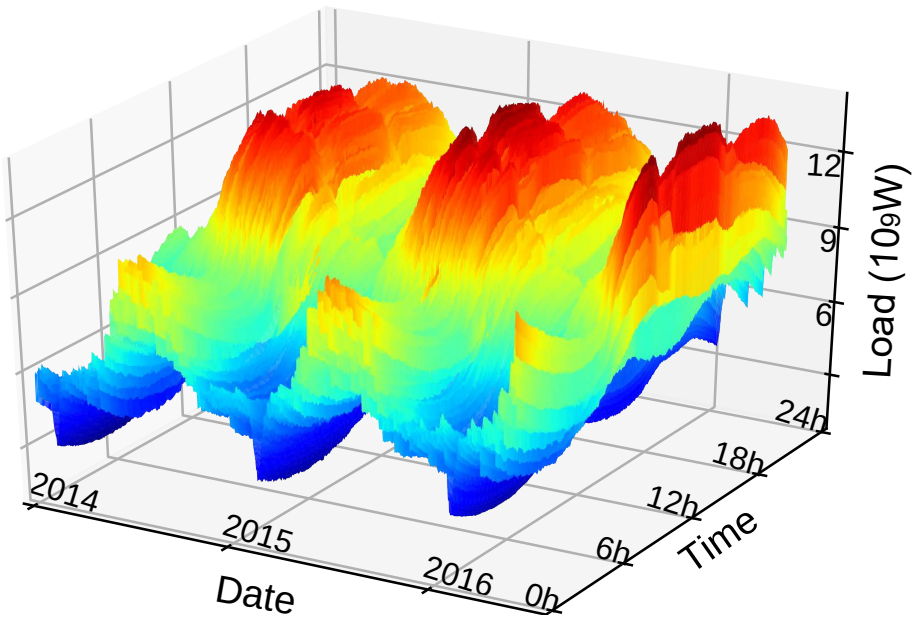


Fig. 5.2 Urban electrical load in China (sampled every five minutes).

the data to the control center. The IoT infrastructure consists of the IoT-enabled devices, including the smart meters, gadgets and appliances, and the communication network. For economy and reliable communication, the author adopts Power Line Communication (PLC), which can transfer low bit-rate data with low costs [101], and ZigBee, which can exchange data wirelessly within a 100m range such as in a home or building, as the communication network. And smart meters, just as the sink nodes in the wireless sensor network (WSN), are responsible for collecting data from the household devices and sensors, and uploading the data to the control center every 30 seconds. The captured data is extremely complicated and contains lots of useful information. The author desires to learn these patterns with a deep learning based system to give out an accurate estimation of the future electrical load.

As mentioned above, the author notices that several researchers have attempted to utilize deep learning in load forecasting, but many of them are facing a problem of low precision. Frequently, the existing approaches give inaccurate daily consumptions even when they have the ability to predict short-term load precisely. It is a serious problem because many participants of electricity markets regard the *daily* consumption value as an important reference for decision-making. A too large estimation may lead to energy waste, while a too small value can possibly cause an insufficient supply. The author adopts a specially designed two-step forecasting scheme to address this problem.

The framework of the proposed load forecasting system is presented in Fig. 5.3. In the proposed method, two individual models are used to respectively predict the daily

consumption and intra-day variation, i.e., daily consumption estimation network (DCEN) and intra-day load forecasting network (ILFN). There are two major reasons that the author designs the two-step forecasting scheme. One reason is, the estimation value of DCEN is not only a helpful guidance for electrical companies and consumers, but an important input to ILFN model, which takes the daily consumption value as a reference and also a limitation. Therefore, with the proposed scheme, the estimated variation can be much closer to the actual load values. The other reason is that electrical load is influenced by various factors, which is usually in unit of days, such as the daily maximum or minimum temperature, daily precipitation, daily sunshine duration and, of course, the date. And, the relevant data is also most often obtained in the unit of days. Based on these facts, the author concentrates all the possible factors at the DCEN model to accurately predict the daily consumption, and only adopts several basic factors to support the intra-day forecasting, in order to simplify the network structure.

There are ten hidden layers in the proposed DCEN model. Layer one and layer two each has 4096 neurons, layer 3$\sim$5 each has 2048 neurons, and layer 6$\sim$10 each has 1024 neurons. The author envisaged implementing DCEN as an extremely complicated model to hold and analyze the super large data. However, the author found that the data engineering is a more efficient way for this task. With well-selected input, even a common deep model can extract sufficient features and give meaningful information for the load forecasting. This will be demonstrated in the experiment section.

As mentioned above, the key problem in DCEN model is the selection and preprocessing of the input data. A massive number of data is acquired by the proposed IoT system. Among them, the following data are picked as the input. As the most instructive reference, the daily consumption of past 7 days is selected; to reflect any periodic characteristics, the time relevant attributes are also adopted, including the date, the Chinese lunar date, day of the week; as the most important and complicated data, weather relevant attributes are of great significance to the DCEN model, including the temperature, air pressure, vapor pressure, precipitation, evaporation, wind speed and sunshine duration. These data are preprocessed to give out the maximum, minimum and average values, and then normalized to generate the final inputs, which include 7 electrical attributes, 3 time relevant attributes and 22 weather relevant attributes.

After obtaining the daily consumption data, ILFN model is adopted to estimate the intra-day load variations. ILFN is also a classic deep model with five hidden layers. And each layer has 512 neurons. The difference is that ILFN needs fewer input attributes, compared to DCEN model, because all the possible influences have been handled by DCEN, and most of them can be neglected in ILFN. The input only includes several basic factors and the recent

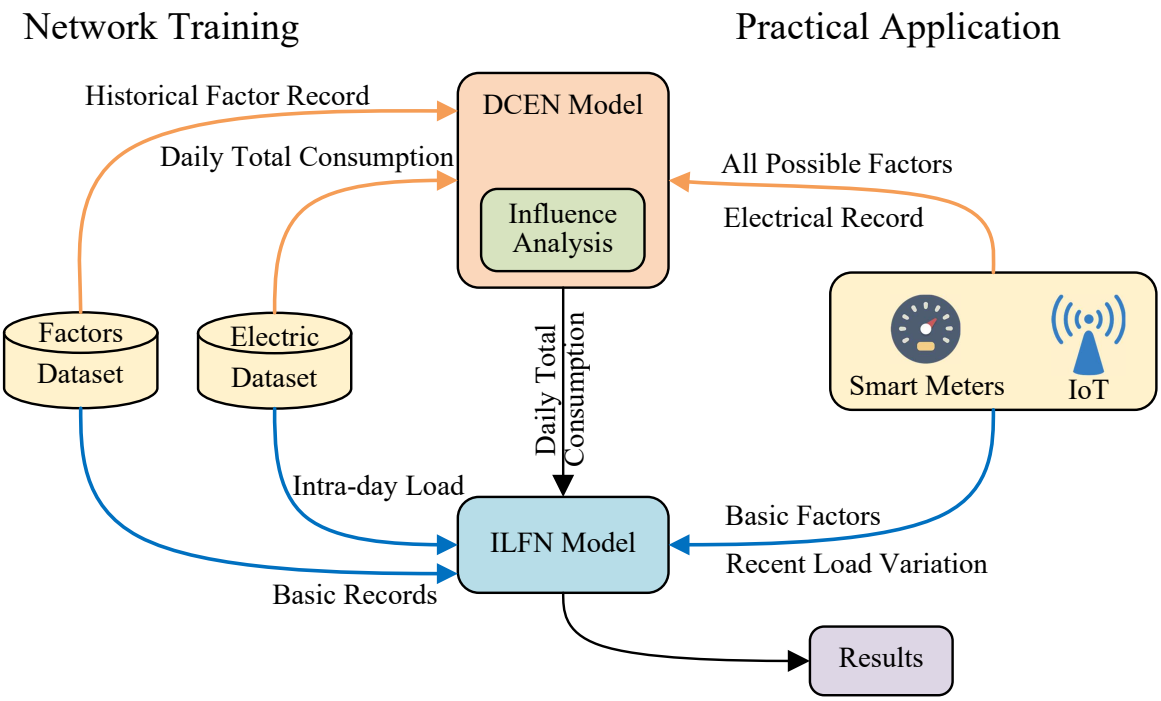


Fig. 5.3 The framework of the proposed load forecasting system.

Table 5.1 Features Comparison of Forecasting Systems.

	Existing systems	IoT-enabled system
Data source	On-board sensors	IoT devices
Granularity	Community	House / Room
Data scope	Limited sensors	Extended by devices
Controllability	Controlled by provider	Controlled by residents
Deploy cost	Expensive sensors	Low-cost sharing
Adaptability	Fully applicable	Available in IoT network

load variation. In detail, the input data includes the estimated daily consumption value, the time relevant attributes, the load values in the last five time units and some relevant readings. DCEN and ILFN are performed for each electrical consumption unit for more nuanced and accurate forecasting. This is mainly benefited by the lower-granularity data from IoT-enabled system. Table 5.1 gives the comparison between traditional systems and the IoT-based system. It can be seen that, the proposed system is able to monitor the detail information of the residents' house, and give solid data support for the forecasting system. These valuable data can be a useful addition to the records captured by the on-board sensors, such as the

operation log of smart appliances, which can be important reference for the residents' energy usage habits. As one of the most important factor, some detailed weather condition data can only be captured by the households sensors, such as the indoor temperature, sunshine duration and indoor air quality, which differentiate in every house but have a strong effect on the energy consumption.

5.2 Influence Analysis

We need to go a step further than merely implementing a forecasting system. Although the proposed DCEN and ILFN model can make accurate predictions, it is no doubt necessary to figure out the mechanism behind the network structure, rather than simply leaving it as a black box. The author starts with clarifying the focus of the system, in other words, what the system really concerns among all the input attributes, including the historical load, weather factors, and time relevant information. This is very important not only for this research, but for other load forecasting applications in different area, because the analysis of forecasting mechanism can serve as a useful guidance for system design. For example, the 32 attributes the author selects in the proposed instance are probably not suitable for other smart grids, especially the Chinese lunar date, which is only of significance to some area in China. So how to find the "right" attributes for a specific area? Influence analysis is an efficient way to perform this task. In the stage of system design, researchers can push all the possible factors into a prototype system, and after adequate training, analyze the contributions of each attribute. Ultimately, the researchers are able to obtain the accurate combination of possible factors. It is an economy solution to know the factors that truly matters before the large-scale deployment of smart meters and sensors. Besides, following the trend of IoT, an increasing number of devices will be IoT-enabled. Therefore, the smart meters will get much more complicated input data in the future, and the influence analysis will play a key role in discriminating the value of various data source. In addition, influence analysis can also be used as a technical measure for the network overfitting, which is a common and serious problem in the training process, but with few effective measuring means for a long time. Overfitting frequently occurs when the deep model is too complicated while having insufficient input data. An overfitted model may have good statistical results on the training materials, but usually perform poorly on actual applications, due to its overreacts to the minor fluctuations. Through influence analysis, researchers can obtain some information regarding whether the overfitting occurs or not. It is mainly because that, an overfitted network usually extracts meaningless features from the raw data, which are impossible to comprehend in most cases. On the contrary, well-trained networks analyze the data in a human-like way.

This difference can become an effective standard of distinguishing the overfitted networks from normal ones.

For these purposes, the author designs a novel visualization method to analyze the contributions from each input attribute to the final output. The author notices, a trained network have fixed parameters, including weights and biases. Therefore, the final output is merely related to the input. And if we change one input unit of an input attribute, the output result will also be changed, which gives a way to infer the contribution of one single input attribute. The analysis algorithm is explained below. First, each attribute in each input sample is fine-tuned to generate new output results. Then each new output value is compared with the former results to show their own contributions. At last, the normalized differences are presented in heatmap form.

An example of the proposed influence analysis is shown in Fig. 5.4. The analysis is conducted with a well-trained network. For simplicity, only some relevant attributes, which have significant influence on the final forecasting result are presented in the figure, including the date, the Chinese lunar calendar date, the day of the week, the temperature and the air pressure. The influences are shown in color. And the red areas have bigger influences than the blue areas. Since all the attributes are normalized and get changed at a same extent, the generated heatmap can give an intuitive representation regarding which parts of the attributes give the most influence to the forecasting results.

Among all the presented heatmaps, the temperature attribute has the most significant effect on the final output, according to the highlighted zone around 25 Celsius degree in the temperature channel. As can be seen, the highlighted zone is converged around 25 degree, which is mainly because 25 degree is a sensitive cut-off point to determine whether to use the air conditioner. When the temperature is lower than 25 degree, there is no cooling needs. On the other hand, when the temperature is much higher than 25 degree, the cooling needs always exist, and a minor temperature change has little influence on the power consumption. Traditionally, there is no demand for heating in south China. Therefore, low temperature also has little influence on the electrical load. The author is very surprised at the rationality and interpretability when the author sees the visualization results for the first time. And, not only the temperature but other attributes show meaningful heatmaps. For example, the date channel and lunar date channel both have highlighted zones around several legal holidays, when the factories are usually closed and, as a result, the electrical load drops down. Lunar date is a traditional calendar in China, and many holidays are based on lunar date. Therefore, the author sets the attribute of lunar date to reflect some specific periodical patterns in China. In the week channel, the influence value of weekends is higher than the one of weekdays, because the weekends are also rest days for many industries. As for the air

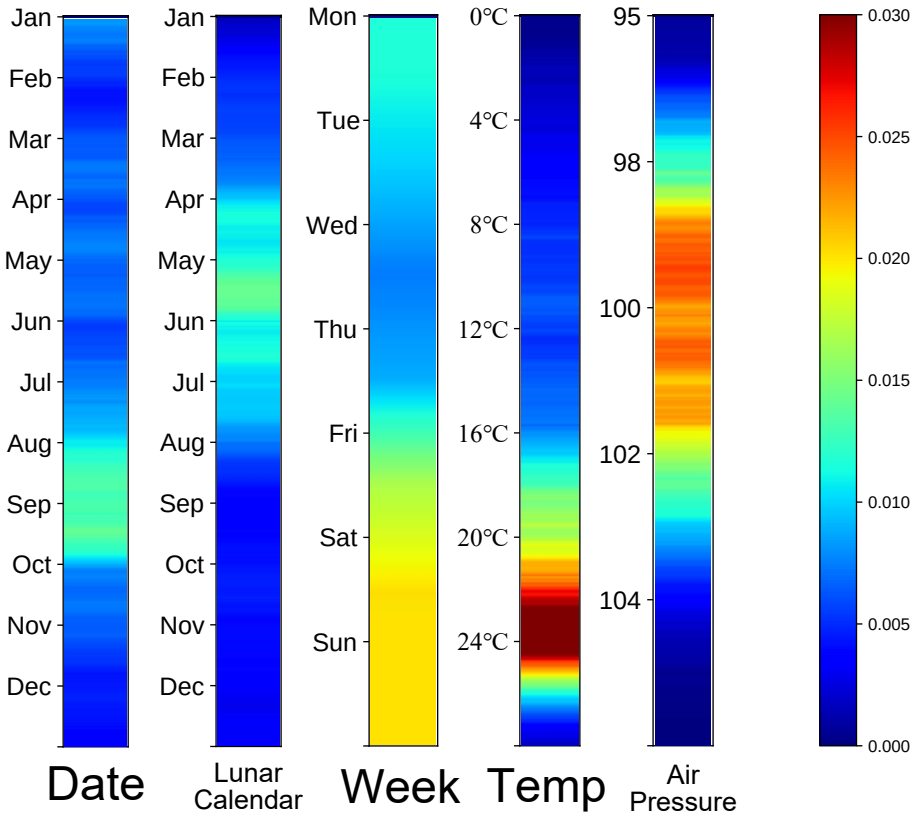


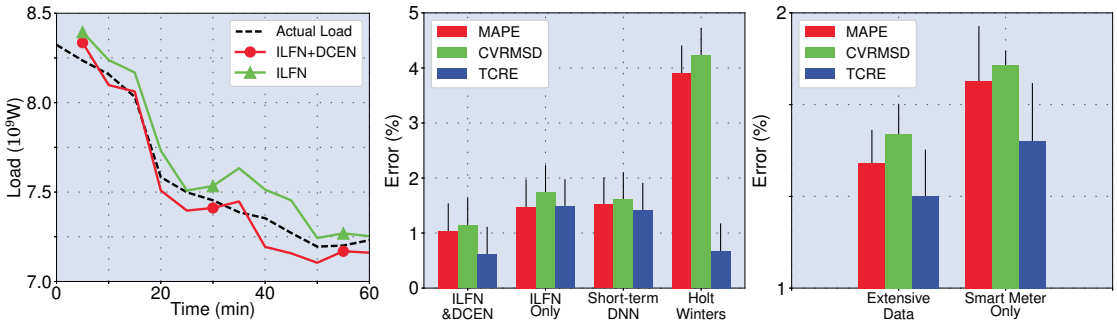
Fig. 5.4 The generated heatmap for influence analysis.

pressure, according to several existing research, there is a strong inverse correlation between the air pressure value and electrical load. It is because that the lower air pressure frequently results in the oppressive weather, which fuels the increase in cooling needs. Besides the channels presented in the figure, the author also analyzes the influence of historical load data, i.e., the daily consumption of past 7 days. Their normalized influence values are 0.07, 0.011, 0.009, 0.007, 0.005, 0.008 and 0.007, respectively for the past days from yesterday to 7 days ago. It can be seen, the closest point in time has the most significant effect on the forecasting result.

As expected, the visualization results demonstrate that, the proposed system can draw rational conclusions with intelligible inferential process. The analysis method enables researchers to select attribute combinations and judge overfitting networks.

5.3 Performance Evaluation

To show the actual forecasting performance of the proposed method and demonstrate the effectiveness of the specially designed two-step forecasting scheme, several simulations are



(a) (b) (c)

Fig. 5.5 Evaluation results of the forecasting methods. (a) Forecasting results (12 predictions in 60 minutes). (b) Comparison of forecasting precision. (c) Comparison in an IoT-enable building.

conducted in this section. The input is the actual record of an urban area in China. The author creates an instance [34] of the proposed models, and train the system with the input data. As shown in Fig. 5.5(a), the author performs two forecasting tests in a period of one hour. The red line indicates the forecasting results which are generated with both of the proposed DCEN and ILFN models, the green line represents the results generated with only ILFN model, and the dotted line is the actual load value. It can be seen that, although both forecasting lines are close to the truth value, the green line shows some offset as a whole, when the two-step forecasting is not adopted. More precisely, nearly all prediction values in the green line are bigger than the actual value, leading to an inaccurate daily total consumption, which is much bigger than the truth value. In contrary, the red line is well distributed in both sides of the dotted line, which may result in a more accurate total consumption. A quantitative analysis experiment is performed to give a precise performance comparison between these two tests, also with another two existing approaches, i.e. the state-of-the-art deep learning based SDNN model [72] working on the same 32 attributes including electrical data, time relevant data and weather data, and the classical HW method working with merely electrical record.

Three mathematical indexes are adopted to quantitatively measure their performance. Fig. 5.5(b) gives the comparison result. The mean absolute percentage error (MAPE) is a famous measure of forecasting precision in statistics. MAPE is scale-independent, and is favored when compare predict accuracy between different datasets. The root mean square deviation (RMSD) is another accuracy index, but can only be used to compare prediction errors of different models for a same dataset, as it is scale-dependent. RMSD is normalized with the mean value of the measurements, namely, coefficient of variation of the RMSD (cvRMSD). In addition, the author designs a new measure named total consumption relative

error (TCRE) to show the effect of the proposed two-step forecasting scheme. TCRE is calculated using the actual daily consumption value and the sum of all estimations in one day. All of these three measures express as percentages.

In Fig. 5.5(b), the first group represents the result of the proposed two-step approach, the second group indicates the prediction without DCEN model, the third group is the SDNN model, and the last group represents the HW method. As can be seen, even without DCEN, the proposed method can achieve a state-of-the-art performance similar to SDNN. However, it is significantly outperformed by the proposed two-step approach in the measure of TCRE. These numerical results once again demonstrate the necessity and effectiveness of the proposed two-step forecasting method. Compared to other approaches, the proposed method performs much better in the prediction precision of both intra-day load variation and daily total consumption.

To demonstrate the effect of the extensive data from IoT-enabled devices, the author performs an additional comparison experiment in a residential building in the same city. As shown in Fig. 5.5(c), the first group is the results of the proposed method using lower-granularity data, which is monitored per house; while the second group represents the results using higher-granularity data, which is captured as the whole building. We can see the first group outperforms the second one in all of the three indexes. This improvement can be attributed to the differences of temperature, humidity, sunshine duration, indoor air quality among the rooms in the building. Through IoT-enabled devices, the system obtains the ability to accurately forecast the energy consumption for every electrical unit, and as a result, improve its prediction precision of the total consumption.

Chapter 6

Artificial Intelligence for Internet-of-Sensors

With the proliferation of Internet-of-Sensors devices, crowdsensing has become an appealing technique to collect and process big data. Meanwhile, the rise of 5th generation wireless systems (5G), especially the new cellular base stations with computing ability, brings about the revolutionary edge computing. Although many approaches regarding the mobile crowdsensing have emerged in the last few years, very few of them are focused on the combination of edge computing and crowdsensing.

In this chapter, the author designs an edge computing based deep learning system for universal crowdsensing tasks, as shown in Fig. 6.1. The right part is the network architecture, in which three edge nodes are connected to the central cloud, i.e., the cellular base station, the wired router and the gateway in the buildings. Through these network nodes, various edge devices can be connected to the cloud, e.g., mobile phones, sensors, smart meters, electrical appliances, etc., which is very common in the current IoT age. The difference between this method and existing crowdsensing approaches is that the author successfully adapts the state-of-the-art deep learning model to the edge computing framework. As a result, with the proposed deep model, the captured data is directly processed in the edge devices and edge nodes, as is shown in the left part of Fig. 6.1. And only the irreversibly extracted features are uploaded to the centralized servers. Although the centralized servers in the cloud end have strong computing abilities, the author attempts to push the computing task to the user end. On the one hand, this scheme can keep the sensitive data in the user side and prevent the unauthorized access to the user privacy; on the other hand, it can also reduce the huge computing load of the centralized servers and fully utilize the computing resource in the cloud edge.

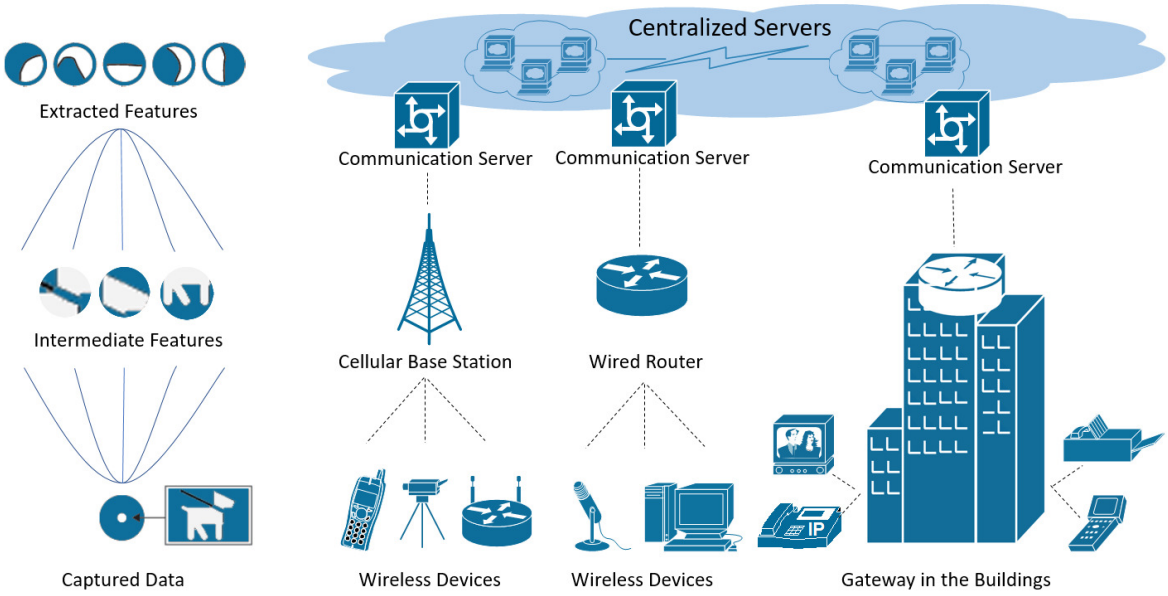


Fig. 6.1 The proposed edge computing based crowdsensing system.

6.1 Crowdsensing System: Concept and Design

In this work, the author attempts to adapt the deep learning methods into the edge computing environment, with the following reasons. As mentioned above, edge computing is for the load balance of the cloud network, and there are two reasons the author chooses deep learning in the system. First, the crowdsensed data should ultimately be processed using some analysis methods, and deep learning is one of the most successful approaches to perform this task. Second, due to the hierarchical structure of deep learning models, it can well meet the requirement of edge computing. Deep learning is able to give both cost-efficiency and privacy protection to the crowdsensing system.

The main principle is to utilize the hierarchical deep model, and allocate its computation tasks to available resources in the cloud, including the centralized servers, the edge nodes, and devices. As shown in Fig. 6.2, an edge computing based deep (ECD) model is proposed for the crowdsensing task. ECD model is mainly characterized in its distributed nature and dynamic framework, which are both essential for the contemporary edge environment.

Fisrt, to adopt the deep learning in the edge computing area, the deep model must have the potentials for distributed operation. Same with all other deep models, the ECD model also has a large number of layers, as shown in Fig. 6.2. One obvious solution is to allocate the layers to all possilbe computing nodes. More precisely, the lower layers can be assigned to the edge of the cloud, and the higher layers can be assigned to the centralized servers. Because the activations, which are the output of each layer, require unidirectional transmission, each

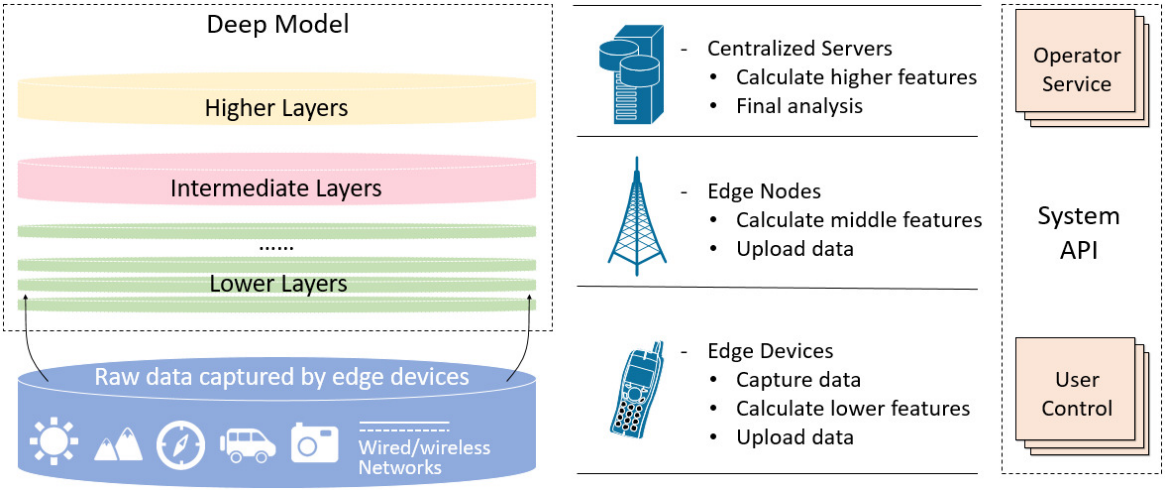


Fig. 6.2 The framework of the edge computing based deep learning system.

computing node must send the calculation results to the next logically adjacent node. The edge devices, including the mobile phones, deployed sensors, vehicles, and other devices which capture the raw data, are responsible for the first part of the crowdsensing task. As the nearest devices to the users, they should be exclusively under the control of their holders. According to the decision of the users, the edge devices can either calculate for a few lowest layers by themselves and send out the extracted features, or totally leave the calculation away and directly push the raw data to the network. It is a key feature of the proposed human-driven design to empower the users with the power to make decisions, which will be detailed in the next section. The edge nodes, including cellular base stations, gateways, and routers, will handle the calculating of several intermediate layers, which is also a big difference with the existing deep learning based crowdsensing approaches. In the next generation of the network infrastructure, the edge nodes will be greatly enhanced in their computing ability, therefore, they will become an extremely important resource for cost-efficient services. The output of the edge nodes is uploaded to the central part of the cloud, i.e., the centralized server, for the calculation of higher layers and final analysis. Although the servers have powerful hardware to perform parallel computing, they are likely to be overloaded due to the extremely large sensing data captured by numerous device. However, with the feature extraction in the lower layers, the burden of centralized servers can significantly decrease.

Through the aforementioned one-way communication scheme, the ECD model obtains the basic distributing ability, however, it needs more to be fully adaptable to the edge computing architecture. The dynamic structure is another important piece of puzzle for this. Because one same deep model cannot handle all crowdsensing tasks, the ECD model in the proposed system should have the ability to be fitted into different objectives. On the other

hand, the upgrade frequency of the edge hardware is usually very low, so it is very difficult to frequently change the lower-layer structures. Therefore, the proposed system is implemented with a flexible framework, which is adaptable for various crowdsensing tasks, while keeping the lower layers relatively stable. The lower layers are integrated into the firmware and only updated with a low frequency, and the higher layers are implemented as the software installed in the centralized servers and keep adaptable for different tasks.

The service provider can deploy several sub-models with different lower layers for some typical kinds of crowdsensing tasks, such as the image classification, 3D scene understanding, road condition monitoring, audio recognition, network control [37], etc. A well-trained sub-model can be used for most of the subtasks in the same category, as their low-level features are very similar.

6.2 Human-driven Crowdsensing

Following the trend of human-driven design, the author attempts to empower the users with the right to decide the balance between energy consumption and privacy protection. On the one hand, users may prefer their devices can output more abstracted features, which is more secure but costs more energy; on the other hand, users may prefer to save the energy of their devices and leave the computing task to the outside devices, which can lead to some privacy concerns. The major reason is that the reconstruction from the output features to the original input deteriorates with the increase of the forward propagation progress. Therefore, there is a trade-off between efficiency and security. The author believes, this dilemma should be decided, or at least partly directed, by the users. In other words, it must be guaranteed that the users can decide that to what level the deep model should be calculated on their own devices, which common users may not be very experienced in but have enough motivation to have the choice. With the human-driven design, service providers can give some recommendations or a changeable security level for the users, just like all the similar solutions integrated in mainstream operating systems or mobile applications for other security problems. As mentioned above, it must be guaranteed that the users can decide that to what level the deep model should be calculated on their own devices. Due to the characteristic of the deep learning model, the smallest calculation unit, which can be assigned to different computing nodes, is the layer. Therefore, the best way to control the calculation level is to select the desired layer numbers for calculation.

The author models the aforementioned dilemma as an optimization problem, and defines the solution set as $\mathscr{B} = \{b_1, b_2, \cdots, b_n\}$, where n represents the maximum number of the layers for calculation. As there are two main factors regarding the energy consumption of

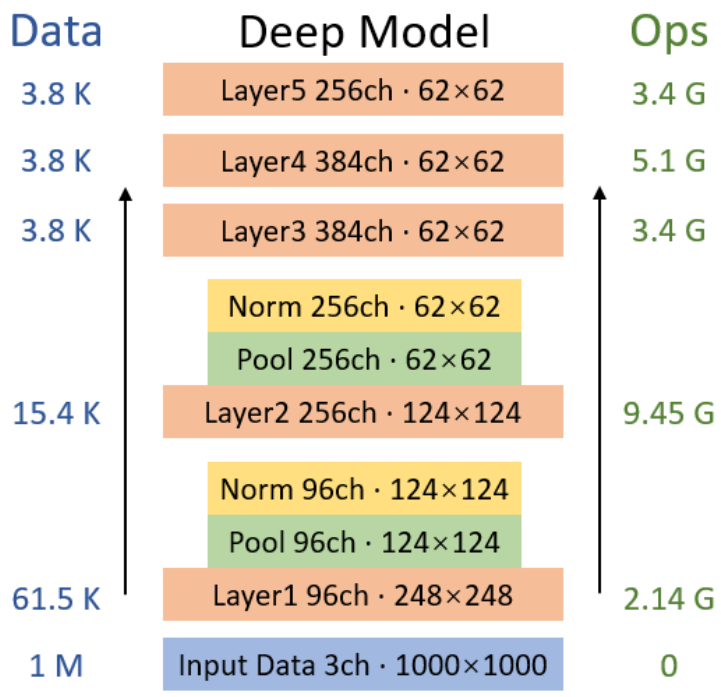


Fig. 6.3 The communication data size and the numbers of unit operations in a deep model.

edge devices, i.e., the network communication $S_{b_i}^{comm}$ and deep model computation $S_{b_i}^{comp}$, the author also defines their energy consumption values as $P(S_{b_i}^{comm})$ and $P(S_{b_i}^{comp})$ respectively. In fact, given a specific model, both $P(S_{b_i}^{comm})$ and $P(S_{b_i}^{comp})$ have fixed values. A common deep model, shown in Fig. 6.3, is used as an example. The first few layers in this model are shown in the figure. The input data is image files with 1000×1000 pixels in three channels. The data column in the left part represents the output size of each layer, which is also the packet size for network communication. The ops column in the right part is for the unit operation number, which is a rough approximation of the calculation cost. To quantify the $P()$ function, the author conducts several tests on various devices, and find the relationship between $P(S_{b_i}^{comm})$, $P(S_{b_i}^{comp})$ and exact power consumption. As shown in Fig. 6.4(a), the author defines a regularized energy cost, in terms of Joule. The dotted black line represents the energy consumption resulted from the calculation, and the red line represents the communication cost. It can be seen the computing cost gradually increase with the calculation level, while the communication cost significantly drops down due to the decrease of output data size.

In addition to the energy consumption, the privacy protection $S_{b_i}^{priv}$ is another important factor to consider in the proposed human-driven crowdsensing system. It is very difficult to accurately measure the risks of the possible privacy leak. A more obvious way is to calculate the consequences if the uploading data are captured by unauthorized devices. There is some

approaches to reconstruct the input data with the output results, such as the intermediate features or even the final results. These approaches can start with a random generated noise data, calculate its output, and compare it with the objective feature, then these methods can optimize the generated input gradually, and, ultimately, get the data which maybe similar to the original input. Due to the characteristic of the deep learning models, it is not possible find the exact initial input from the output, and the more abstract the features are, the less similar the inferred input is. Therefore, the author defines a data similarity index $P(S_{b_i}^{priv})$ to measure the sensitivity of the output data. This index measures the degree of similarity between the reconstruction of the output feature and the raw captured data. Because the extracted features are not exactly reversible to the specific input, therefore, the output data with lower reconstruction similarity has a better security, and it can keep the users' privacy even if it is leaked to someone else. This point is especially notable in image processing tasks, where the activations of the higher layers are much more abstracted and illegible than the ones of the lower layers. In addition, there are lots of ways to measure the similarity, for example, the Euclidean distance, etc. Fig. 6.4(b) gives an example regarding the relationship between data similarity and calculation level. In conformance with the estimates, the similarity decreases with increasing calculation level.

Given $P(S_{b_i}^{comm})$, $P(S_{b_i}^{comp})$ and $P(S_{b_i}^{priv})$, an utility function is defined as

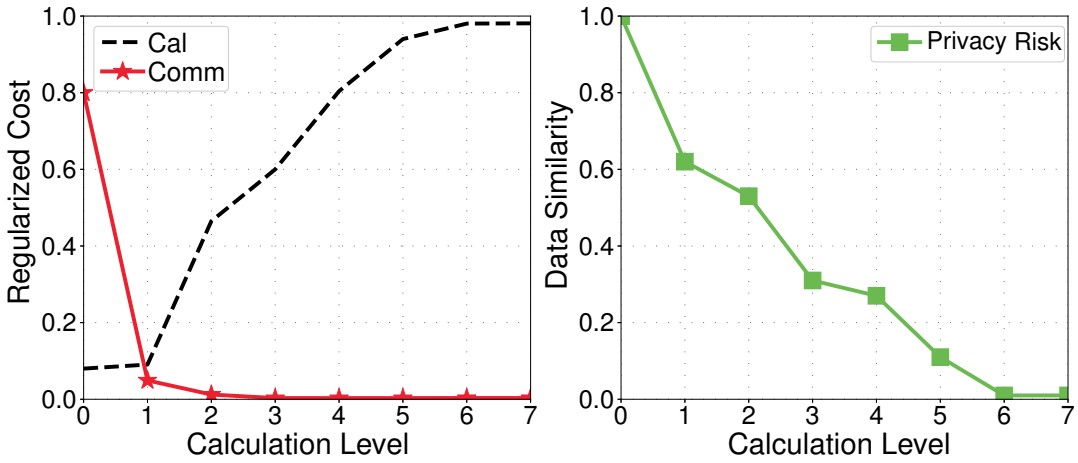
$$U = \lambda_1(P(S_{b_i}^{comm}) + P(S_{b_i}^{comp})) + \lambda_2 P(S_{b_i}^{priv}), \tag{6.1}$$

in order to make a comprehensive evaluation on different solutions. The weights λ_1 and λ_2 are set by the users to reflect the individual orientation. The solution with smallest U value is presented to the users as an instructive recommendation.

6.3 Performance Evaluation

6.3.1 Demonstration for System Validity

The author first uses a small testbed to demonstrate the validity of the proposed system, i.e., it is a feasible solution to divide the deep model into several parts and allocate them to different computational resources, and it works well with actual hardware. The testbed consists a central server, an edge server, and a mobile device. The edge server has a wireless network interface card for wireless access and an Ethernet network interface connected to the central server. The author uses the Raspberry Pi as the edge server, which is fully capable of forward propagation.



(a) (b)

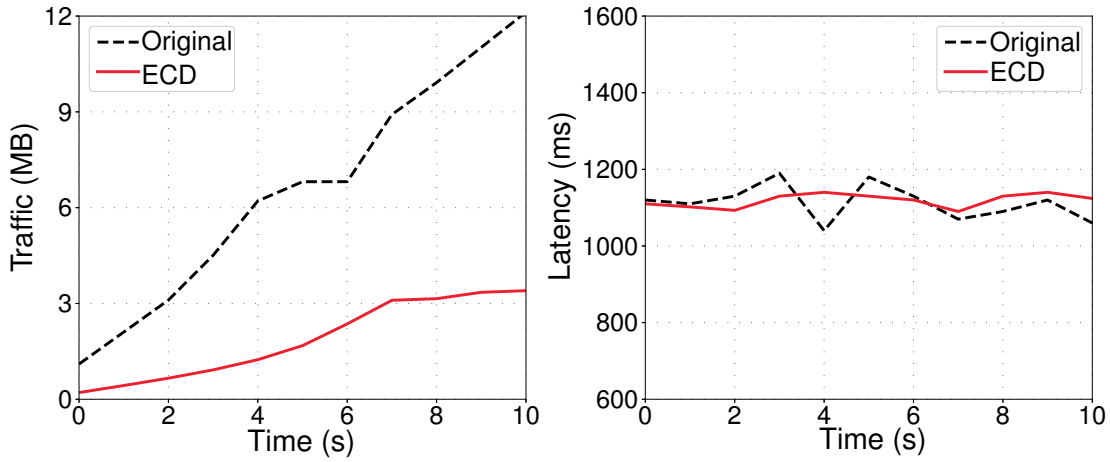
Fig. 6.4 The energy cost and privacy risk of different calculation levels. (a) The calculation and communication cost of each calculation level. (b) The privacy risk of each calculation level.

The author implements one image classification application. The user captures an image, and attempts to know the information of the image. In the testbed, the author installs the classification model in the mobile phone, edge server, and the central server. The mobile device consistently captures images and calculates for the results using the available resources. The author compares the performance between the common solution, i.e., directly uploading the raw data, and the proposed method. The results are shown in Fig. 6.5. Fig. 6.5(a) gives the traffic change during actual deep learning based tasks. Compared with the original method, the proposed ECD approach consumes less network traffic. After 10 seconds, the traffic size of the original method have achieved three times larger than the proposed method. In the meantime, according to Fig. 6.5 (b), the ECD can keep a similar performance with the original method in the response latency. It can be seen that the proposed method outperforms the original solution in the network traffic while keeping a similar latency.

6.3.2 Numerical Simulation

In this simulation, two servers, ten edge nodes and 100 mobile devices are deployed to serve as the edge computing system. The mobile devices keep performing sensing tasks and upload the data to the cloud. The adopted deep model is used for image recognition, and all the collected data is in the form of image. The data size is measured in megabyte.

As shown in Fig. 6.6, the author tests the energy saving performance with several different strategies, and present their consumption changes with increasing captured data size. The two



(a) (b)

Fig. 6.5 Network traffic and latency of the demonstration application.

black curves are the simulation results of the proposed ECD method, with $\lambda_1 = 0.8, \lambda_2 = 0.2$ and $\lambda_1 = 0.2, \lambda_2 = 0.8$, respectively. The green line represents the average strategy, i.e., always calculate for the average layers in the edge devices. The blue line is the maximum strategy, in which the edge devices are responsible for all lower layers; while the orange line represents the strategy in which the edge devices directly upload the raw captured data. And the last one, the red line, is a random select strategy, i.e., the calculation level is randomly selected in each run. It can be seen that the proposed ECD model has a significant advantage in energy efficiency. When λ_2 is set with a large weight value, the ECD model shows an obvious orientation on the cost saving. As a result, it achieves better performance than the one with a larger λ_1. On the contrary, when λ_1 is set to 0.8, the ECD model cares more about the user privacy. Fig. 6.7 (a) gives the comparison results of data similarity. And the deep model with a larger λ_1 value, "ECD2" bar in the figure, prefers safety over the energy efficiency. Of course, the "max" bar shows a huge lead in this test, because it calculates for all the lower layers and is able to output the most abstract features. In addition, the author also presents the privacy-protection efficiency in Fig. 6.7 (b), in order to measure the "value" of unit energy consumption. It can be seen, the proposed method has the highest efficiency in improving the effect of privacy protection with one unit of energy consumption.

Another simulation is conducted to demonstrate the performance of the ECD model in balancing the computing load, as shown in Fig. 6.8. The difference between left and right sub-figures is that the edge devices and edge nodes simply send the raw data to the centralized servers for further computing in the left, while in the right one both edge devices and nodes play a part in the computing task. The z axis represents the hardware usage. We can see the

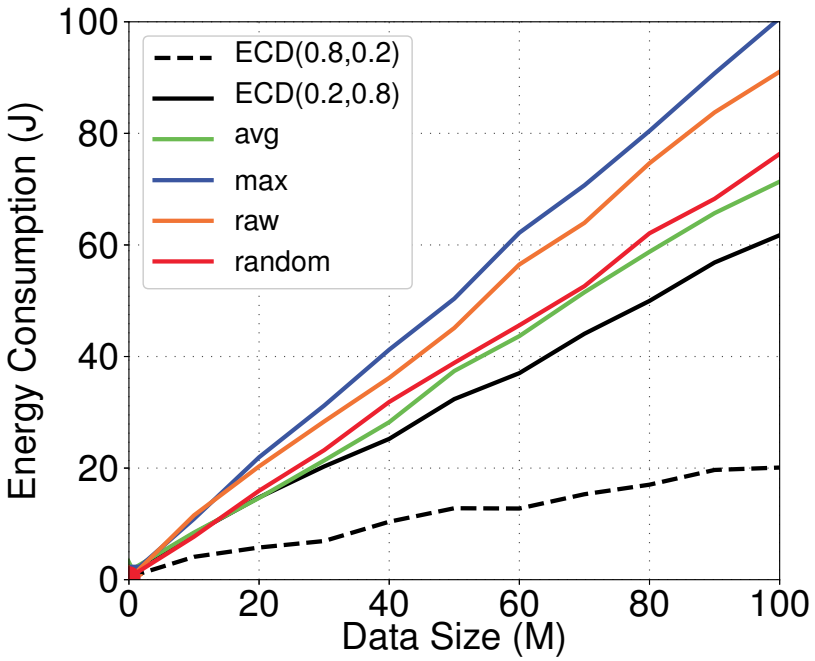


Fig. 6.6 The comparison results of energy consumption.

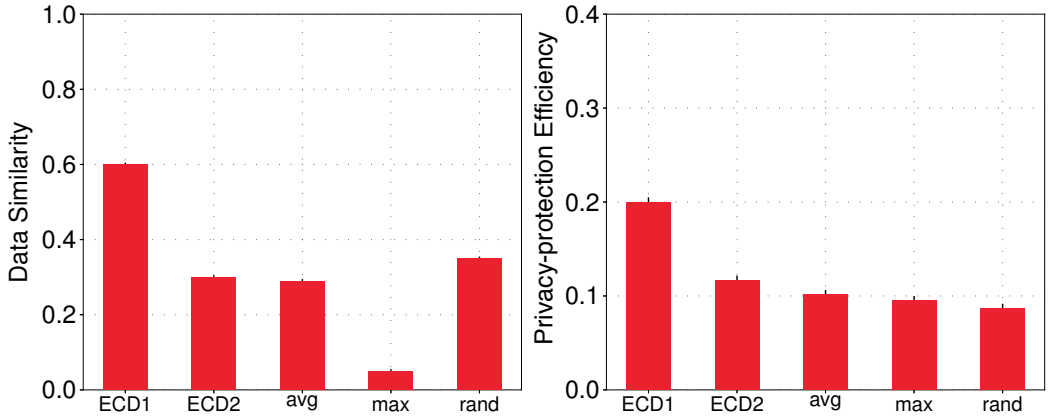
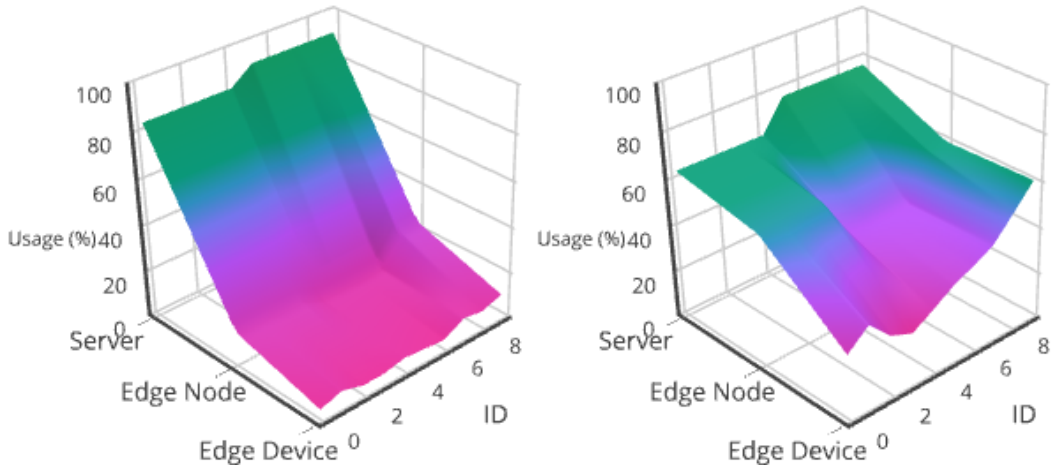


Fig. 6.7 The comparison experiments on privacy-preserving. (a) The results of output data similarity. (b) The results of privacy-protection efficiency.

servers in the left carry most of the calculation burden, which results in the steep surface in the left. On the contrary, with the ECD model, all the computing resource in the right can participate in the process of the captured data, and the load surface is flat and balanced.



(a) (b)

Fig. 6.8 Simulation results of load balance. (a) Edge devices directly upload the raw data for further process. (b) Edge devices and edge nodes participate in the pre-process of the captured data, using the proposed ECD model.

These numerical results once again demonstrate the necessity and effectiveness of the proposed hierarchical deep model. Compared to other approaches, the proposed ECD model performs much better in energy consumption, privacy security and load balancing.

Chapter 7

Conclusion and Future Works

A series of approaches are proposed in the dissertation for several emerging Internet-of-Things (IoT) applications, including Internet-of-Robots, Internet-of-Vehicles, Internet-of-Energy, and Internet-of-Sensors. In this chapter, the author will give their conclusions, as well as their limitations and future works.

7.1 Intelligent Robots

In Chapter 3, a CNN-based robotic 3D scene understanding method is proposed. The most significant advantage of the method is its view-invariant ability and no requirement on RGB data, which are essential in disordered and extreme disaster scenarios. The author proposes several loss functions to perform multi-task learning process on the proposed models. And the author also works out an optimization algorithm to improve the learning speed. The simulations demonstrate that the proposed method is effective, and outperforms many state-of-art approaches in several comparison experiments.

A limitation of the proposed system is that a big size of labeled data are still necessary in the training process. Therefore, in the future work, the author will work on the automatic labeling problem to further decrease the system cost.

7.2 Intelligent Vehicles

In Chapter 4, a human-like autonomous driving system is proposed. This system mainly includes two parts, i.e. a road scene perception method and an empirical decision-making network. One obvious difference with the existing approaches is that it can imitate human drivers' social intelligence, which can better adapt the self-driving vehicles to the real-life

road conditions. In addition, the author implements an efficient training scheme to improve the quality and speed of data collection, and alleviates the tedious and time-consuming manual labeling process. The author also finds a feasible approach to analyze the possible influence factors in the decision-making process, which can help in the testing and validating of autonomous driving systems. The experimental results prove that the proposed method is efficient, and shows meaningful analysis results in the visualizations of unit influence.

Future work includes developing a more efficient optimization method to decrease the time cost of the training process. Also, some information may be extracted from RGB data, even when it is incomplete and unstable, which can be a useful supplement to the proposed method. In addition, the author will extend this work by conducting more driving testings in order to further expand the dataset for network training.

7.3 Intelligent Energy

In Chapter 5, an IoT-based electrical load forecasting method is proposed. A huge advantage of the method is its two-step forecasting scheme, which significantly increases the prediction precision for daily total consumption. Another major difference is that, the author adopts deep learning methods to learn complicated patterns form all the possible influences, and achieves a state-of-the-art forecasting performance in the evaluations. In addition, the author also proposes an analysis method to find the relationship between the influences and the electrical load, and designs a heatmap generation method to show the specific impacts of each attribute on forecasting results. This analysis method is also of much guiding significance for the smart grids in other countries, especially for the ones with vast territory and varied climates. The results prove its effectiveness.

One limitation is that, in the proposed system, a huge number of data needs to be transferred on the communication network, which can bring a big challenge to the existing infrastructures. One feasible solution is to adopt edge servers near the client side for better computing balance and less communication cost, which is also included in future works.

7.4 Intelligent Sensing

In Chapter 6, the author proposes an edge computing based crowdsensing method, which can adopt the available computing resources in the whole network, both in the cloud and in the edge side, to ensure the load balance and reduce communication cost. The author also adopts a specially-designed deep model to transform the crowdsensing problem into a hierarchical task, which is not only an effective data processing and analysis approach, but gives users

the right to control the crowdsensing process. The experimental results well prove that it can provide better performance while considering and keeping users' privacy.

In the future, it is necessary to test the compatibility of the proposed framework in different cloud environments, especially the calculation efficiency on various 5G base stations.

References

[1] Alahakoon, D. and Yu, X. (2016). Smart electricity meter data intelligence for future energy systems: A survey. *IEEE Transactions on Industrial Informatics*, 12(1):425–436.

[2] Alsheikh, M. A., Niyato, D., Lin, S., p. Tan, H., and Han, Z. (2016). Mobile big data analytics using deep learning and apache spark. *IEEE Network*, 30(3):22–29.

[3] Arbeláez, P., Pont-Tuset, J., Barron, J. T., Marques, F., and Malik, J. (2014). Multiscale combinatorial grouping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 328–335.

[4] Atia, M. M., Liu, S., Nematallah, H., Karamat, T. B., and Noureldin, A. (2015). Integrated indoor navigation system for ground vehicles with automatic 3-d alignment and position initialization. *IEEE Transactions on Vehicular Technology*, 64(4):1279–1292.

[5] Barbarossa, S., Ceci, E., Merluzzi, M., and Calvanese-Strinati, E. (2017). Enabling effective mobile edge computing using millimeterwave links. In *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 367–372.

[6] Bilal, K. and Erbad, A. (2017). Edge computing for interactive media and video streaming. In *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 68–73.

[7] Cetin, K. S. and O'Neill, Z. (2017). Smart meters and smart devices in buildings: a review of recent progress and influence on electricity use and peak demand. *Current Sustainable/Renewable Energy Reports*, 4(1):1–7.

[8] Chen, C., Luan, T. H., Guan, X., Lu, N., and Liu, Y. (2017). Connected vehicular transportation: Data analytics and traffic-dependent networking. *IEEE Vehicular Technology Magazine*, 12(3):42–54.

[9] Chen, C., Seff, A., Kornhauser, A., and Xiao, J. (2015a). Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2722–2730.

[10] Chen, C., Zhu, S., Guan, X., and Shen, X. S. (2014). *Wireless sensor networks: Distributed consensus estimation*. Springer.

[11] Chen, X., Kundu, K., Zhu, Y., Berneshawi, A. G., Ma, H., Fidler, S., and Urtasun, R. (2015b). 3d object proposals for accurate object class detection. In *Advances in Neural Information Processing Systems*, pages 424–432.

[12] Cheng, G., Zhou, P., and Han, J. (2016). Learning rotation-invariant convolutional neural networks for object detection in vhr optical remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, 54(12):7405.

[13] Choi, S., Zhou, Q.-Y., Miller, S., and Koltun, V. (2016). A large dataset of object scans. *arXiv:1602.02481*.

[14] Coelho, I. M., Coelho, V. N., da S. Luz, E. J., Ochi, L. S., Guimarães, F. G., and Rios, E. (2017). A gpu deep learning metaheuristic based model for time series forecasting. *Applied Energy*.

[15] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE.

[16] Eitel, A., Springenberg, J. T., Spinello, L., Riedmiller, M., and Burgard, W. (2015). Multimodal deep learning for robust rgb-d object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 681–687. IEEE.

[17] Erhan, D., Manzagol, P.-A., Bengio, Y., Bengio, S., and Vincent, P. (2009). The difficulty of training deep architectures and the effect of unsupervised pre-training. In *AISTATS*, volume 5, pages 153–160.

[18] Fadlullah, Z. M., Tang, F., Mao, B., Kato, N., Akashi, O., Inoue, T., and Mizutani, K. (2017). State-of-the-art deep learning: Evolving machine intelligence toward tomorrow?s intelligent network traffic control systems. *IEEE Communications Surveys Tutorials*, 19(4):2432–2455.

[19] Galben, G. (2011). New three-dimensional velocity motion model and composite odometry-inertial motion model for local autonomous navigation. *IEEE Transactions on Vehicular Technology*, 60(3):771–781.

[20] Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448.

[21] Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Aistats*, volume 15, page 275.

[22] Gupta, S., Girshick, R., Arbeláez, P., and Malik, J. (2014). Learning rich features from rgb-d images for object detection and segmentation. In *European Conference on Computer Vision*, pages 345–360. Springer.

[23] Hadsell, R., Sermanet, P., Ben, J., Erkan, A., Scoffier, M., Kavukcuoglu, K., Muller, U., and LeCun, Y. (2009). Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics*, 26(2):120–144.

[24] Harzallah, H., Jurie, F., and Schmid, C. (2009). Combining efficient object localization and image classification. In *2009 IEEE 12th International Conference on Computer Vision*, pages 237–244. IEEE.

[25] He, S., Dong, M., Ota, K., Wu, J., Li, J., and Li, G. (2017). Software-defined efficient service reconstruction in fog using content awareness and weighted graph. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pages 1–6.

[26] Hernandez, L., Baladron, C., Aguiar, J. M., Carro, B., Sanchez-Esguevillas, A. J., Lloret, J., and Massana, J. (2014). A survey on electric power demand forecasting: Future trends in smart grids, microgrids and smart buildings. *IEEE Communications Surveys Tutorials*, 16(3):1460–1495.

[27] Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

[28] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

[29] Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206.

[30] Hua, B.-S., Pham, Q.-H., Nguyen, D. T., Tran, M.-K., Yu, L.-F., and Yeung, S.-K. (2016). Scenenn: A scene meshes dataset with annotations. In *International Conference on 3D Vision (3DV)*.

[31] Imani, Z. and Soltanizadeh, H. (2016). Person reidentification using local pattern descriptors and anthropometric measures from videos of kinect sensor. *IEEE Sensors Journal*, 16(16):6227–6238.

[32] Jayaraman, P. P., Gomes, J. B., Nguyen, H. L., Abdallah, Z. S., Krishnaswamy, S., and Zaslavsky, A. (2015). Scalable energy-efficient distributed data analytics for crowdsensing applications in mobile environments. *IEEE Transactions on Computational Social Systems*, 2(3):109–123.

[33] Ji, J., Khajepour, A., Melek, W., and Huang, Y. (2016). Path planning and tracking for vehicle collision avoidance based on model predictive control with multi-constraints. *IEEE Transactions on Vehicular Technology*, PP(99):1–1.

[34] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22Nd ACM International Conference on Multimedia*, MM '14, pages 675–678, New York, NY, USA. ACM.

[35] Jiang, J. and Qian, Y. (2016). Distributed communication architecture for smart grid applications. *IEEE Communications Magazine*, 54(12):60–67.

[36] Kato, N., Fadlullah, Z. M., Mao, B., Tang, F., Akashi, O., Inoue, T., and Mizutani, K. (2017a). The deep learning vision for heterogeneous network traffic control: Proposal, challenges, and future perspective. *IEEE Wireless Communications*, 24(3):146–153.

[37] Kato, N., Fadlullah, Z. M., Mao, B., Tang, F., Akashi, O., Inoue, T., and Mizutani, K. (2017b). The deep learning vision for heterogeneous network traffic control: Proposal, challenges, and future perspective. *IEEE Wireless Communications*, 24(3):146–153.

[38] Kiani, A. and Ansari, N. (2017). Toward hierarchical mobile edge computing: An auction-based profit maximization approach. *IEEE Internet of Things Journal*, 4(6):2082–2091.

[39] Kong, Y., Ding, Z., Li, J., and Fu, Y. (2017). Deeply learned view-invariant features for cross-view action recognition. *IEEE Transactions on Image Processing*, 26(6):3028–3037.

[40] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[41] Lai, K., Bo, L., Ren, X., and Fox, D. (2011). A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817–1824. IEEE.

[42] Lane, N. D. and Georgiev, P. (2015). Can deep learning revolutionize mobile sensing? In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, pages 117–122. ACM.

[43] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

[44] Lennie, P. (2003). The cost of cortical computation. *Current biology*, 13(6):493–497.

[45] Li, H., Dong, M., and Ota, K. (2016a). Control plane optimization in software-defined vehicular ad hoc networks. *IEEE Transactions on Vehicular Technology*, 65(10):7895–7904.

[46] Li, H., Ota, K., and Dong, M. (2016b). Network virtualization optimization in software defined vehicular ad-hoc networks. In *2016 IEEE 84th Vehicular Technology Conference (VTC-Fall)*, pages 1–5.

[47] Li, H., Ota, K., Dong, M., and Guo, M. (2017). Mobile crowdsensing in software defined opportunistic networks. *IEEE Communications Magazine*, 55(6):140–145.

[48] Li, H., Ota, K., Dong, M., Vasilakos, A., and Nagano, K. (2018a). Multimedia processing pricing strategy in gpu-accelerated cloud computing. *IEEE Transactions on Cloud Computing*, pages 1–1.

[49] Li, L., Ota, K., and Dong, M. (2018b). Deep learning for smart industry: Efficient manufacture inspection system with fog computing. *IEEE Transactions on Industrial Informatics*, pages 1–1.

[50] Li, L. and Xiao, N. (2015). Volumetric view planning for 3d reconstruction with multiple manipulators. *Industrial Robot: An International Journal*, 42(6):533–543.

[51] Liebelt, J. and Schmid, C. (2010). Multi-view object class detection with a 3d geometric model. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1688–1695.

[52] Liu, J., Wan, J., Zeng, B., Wang, Q., Song, H., and Qiu, M. (2017). A scalable and quick-response software defined vehicular network assisted by mobile edge computing. *IEEE Communications Magazine*, 55(7):94–100.

[53] Liu, X., Lu, L., Shen, Z., and Lu, K. (2016). A novel face recognition algorithm via weighted kernel sparse representation. *Future Generation Computer Systems*.

[54] Lloret, J., Tomas, J., Canovas, A., and Parra, L. (2016). An integrated iot architecture for smart metering. *IEEE Communications Magazine*, 54(12):50–57.

[55] Lueth, K. L. (2018). State of the IoT 2018: Number of IoT devices now at 7B – market accelerating. https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/.

[56] Luo, T., Kanhere, S. S., Huang, J., Das, S. K., and Wu, F. (2017). Sustainable incentives for mobile crowdsensing: Auctions, lotteries, and trust and reputation systems. *IEEE Communications Magazine*, 55(3):68–74.

[57] Malisiewicz, T., Gupta, A., and Efros, A. A. (2011). Ensemble of exemplar-svms for object detection and beyond. In *2011 International Conference on Computer Vision*, pages 89–96. IEEE.

[58] Mao, B., Fadlullah, Z. M., Tang, F., Kato, N., Akashi, O., Inoue, T., and Mizutani, K. (2017). Routing or computing? the paradigm shift towards intelligent computer network packet transmission based on deep learning. *IEEE Transactions on Computers*, 66(11):1946–1960.

[59] Menze, M. and Geiger, A. (2015). Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[60] Muller, U., Ben, J., Cosatto, E., Flepp, B., and Cun, Y. L. (2005). Off-road obstacle avoidance through end-to-end learning. In *Advances in neural information processing systems*, pages 739–746.

[61] Nathan Silberman, Derek Hoiem, P. K. and Fergus, R. (2012). Indoor segmentation and support inference from rgbd images. In *ECCV*.

[62] Nielsen, J. J., Madueño, G. C., Pratas, N. K., Sørensen, R. B., Stefanovic, C., and Popovski, P. (2015). What can wireless cellular technologies do about the upcoming smart metering traffic? *IEEE Communications Magazine*, 53(9):41–47.

[63] Ota, K., Dong, M., Gui, J., and Liu, A. (2018). Quoin: Incentive mechanisms for crowd sensing networks. *IEEE Network*, PP(99):1–6. doi: 10.1109/MNET.2017.1500151.

[64] Ota, K., Dong, M., Zhu, H., Chang, S., and Shen, X. (2011). Traffic information prediction in urban vehicular networks: A correlation based approach. In *2011 IEEE Wireless Communications and Networking Conference*, pages 1021–1025.

[65] Pournajaf, L., Garcia-Ulloa, D. A., Xiong, L., and Sunderam, V. (2016). Participant privacy in mobile crowd sensing task management: a survey of methods and challenges. *ACM SIGMOD Record*, 44(4):23–34.

[66] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan.

[67] Rana, M. and Koprinska, I. (2016). Forecasting electricity load with advanced wavelet neural networks. *Neurocomputing*, 182:118–132.

[68] Rimal, B. P., Van, D. P., and Maier, M. (2017). Mobile edge computing empowered fiber-wireless access networks in the 5g era. *IEEE Communications Magazine*, 55(2):192–200.

[69] Rodrigues, T. G., Suto, K., Nishiyama, H., and Kato, N. (2017). Hybrid method for minimizing service delay in edge cloud computing through vm migration and transmission power control. *IEEE Transactions on Computers*, 66(5):810–819.

[70] Rodrigues, T. G., Suto, K., Nishiyama, H., Kato, N., and Temma, K. (2018). Cloudlets activation scheme for scalable mobile edge computing with transmission power control and virtual machine migration. *IEEE Transactions on Computers*, 67(9):1287–1300.

[71] Rusu, R. B. and Cousins, S. (2011). 3d is here: Point cloud library (pcl). In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1–4. IEEE.

[72] Ryu, S., Noh, J., and Kim, H. (2016). Deep neural network based demand side short term load forecasting. *Energies*, 10(1):3.

[73] Salsano, S., Chiaraviglio, L., Blefari-Melazzi, N., Parada, C., Fontes, F., Mekuria, R., and Griffioen, D. (2017). Toward superfluid deployment of virtual functions: Exploiting mobile edge computing for video streaming. In *2017 29th International Teletraffic Congress (ITC 29)*, volume 2, pages 48–53.

[74] Sato, K. and Fujii, T. (2017). Radio environment aware computation offloading with multiple mobile edge computing servers. In *2017 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 1–5.

[75] Schwarz, M., Schulz, H., and Behnke, S. (2015). Rgb-d object recognition and pose estimation based on pre-trained convolutional neural network features. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1329–1335. IEEE.

[76] Serre, T., Wolf, L., Bileschi, S., Riesenhuber, M., and Poggio, T. (2007). Robust object recognition with cortex-like mechanisms. *IEEE transactions on pattern analysis and machine intelligence*, 29(3):411–426.

[77] Sheridan, T. B. (2016). Human–robot interaction status and challenges. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 58(4):525–532.

[78] Shi, B., Bai, S., Zhou, Z., and Bai, X. (2015). Deeppano: Deep panoramic representation for 3-d shape recognition. *IEEE Signal Processing Letters*, 22(12):2339–2343.

[79] Singh, A., Sha, J., Narayan, K. S., Achim, T., and Abbeel, P. (2014). Bigbird: A large-scale 3d database of object instances. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 509–516. IEEE.

[80] Socher, R., Huval, B., Bath, B., Manning, C. D., and Ng, A. Y. (2012). Convolutional-recursive deep learning for 3d object classification. In *Advances in Neural Information Processing Systems*, pages 665–673.

[81] Son, Y. S., Kim, W., Lee, S. H., and Chung, C. C. (2015). Robust multirate control scheme with predictive virtual lanes for lane-keeping system of autonomous highway driving. *IEEE Transactions on Vehicular Technology*, 64(8):3378–3391.

[82] Song, S. and Xiao, J. (2014). Sliding shapes for 3d object detection in depth images. In *European Conference on Computer Vision*, pages 634–651. Springer.

[83] Song, S. and Xiao, J. (2015). Deep sliding shapes for amodal 3d object detection in rgb-d images. *arXiv preprint arXiv:1511.02300*.

[84] Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E. (2015). Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 945–953.

[85] Su, Z., Hui, Y., Luan, T. H., and Guo, S. (2017a). Engineering a game theoretic access for urban vehicular networks. *IEEE Transactions on Vehicular Technology*, 66(6):4602–4615.

[86] Su, Z., Xu, Q., Hui, Y., Wen, M., and Guo, S. (2017b). A game theoretic approach to parked vehicle assisted content delivery in vehicular ad hoc networks. *IEEE Transactions on Vehicular Technology*, 66(7):6461–6474.

[87] Sun, B. and Feng, H. (2017). Efficient compressed sensing for wireless neural recording: A deep learning approach. *IEEE Signal Processing Letters*, PP(99):1–1.

[88] Sun, W. and Liu, J. (2017). Congestion-aware communication paradigm for sustainable dense mobile crowdsensing. *IEEE Communications Magazine*, 55(3):62–67.

[89] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9.

[90] Tang, F., Fadlullah, Z. M., Mao, B., and Kato, N. (2018). An intelligent traffic load prediction based adaptive channel assignment algorithm in sdn-iot: A deep learning approach. *IEEE Internet of Things Journal*, pages 1–1.

[91] Tao, M., Ota, K., and Dong, M. (2017a). Foud: Integrating fog and cloud for 5g-enabled v2g networks. *IEEE Network*, 31(2):8–13.

[92] Tao, X., Ota, K., Dong, M., Qi, H., and Li, K. (2017b). Performance guaranteed computation offloading for mobile-edge cloud computing. *IEEE Wireless Communications Letters*, 6(6):774–777.

[93] Tran, T. X., Hajisami, A., Pandey, P., and Pompili, D. (2017a). Collaborative mobile edge computing in 5g networks: New paradigms, scenarios, and challenges. *IEEE Communications Magazine*, 55(4):54–61.

[94] Tran, T. X., Pandey, P., Hajisami, A., and Pompili, D. (2017b). Collaborative multi-bitrate video caching and processing in mobile-edge computing networks. In *2017 13th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, pages 165–172.

[95] Uijlings, J. R., van de Sande, K. E., Gevers, T., and Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision*, 104(2):154–171.

[96] Valerio, L., Passarella, A., and Conti, M. (2017). Optimal trade-off between accuracy and network cost of distributed learning in mobile edge computing: An analytical approach. In *2017 IEEE 18th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–9.

[97] Wang, C., Yu, F. R., Liang, C., Chen, Q., and Tang, L. (2017). Joint computation offloading and interference management in wireless cellular networks with mobile edge computing. *IEEE Transactions on Vehicular Technology*, 66(8):7432–7445.

[98] Wang, L., Zhang, D., Wang, Y., Chen, C., Han, X., and M'hamed, A. (2016). Sparse mobile crowdsensing: challenges and opportunities. *IEEE Communications Magazine*, 54(7):161–167.

[99] Xiao, L., Shao, W., Wang, C., Zhang, K., and Lu, H. (2016). Research and application of a hybrid model based on multi-objective optimization for electrical load forecasting. *Applied Energy*, 180:213–233.

[100] Xiong, X., Wang, J., Zhang, F., and Li, K. (2016). Combining deep reinforcement learning and safety based control for autonomous driving. *arXiv preprint arXiv:1612.00147*.

[101] Yigit, M., Gungor, V. C., Tuna, G., Rangoussi, M., and Fadel, E. (2014). Power line communication technologies for smart grid applications: A review of advances and challenges. *Computer Networks*, 70:366–383.

[102] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328.

[103] Yuan, Q., Gao, Y., and Li, Y. (2016). Suppose future traffic accidents based on development of self-driving vehicles. In *Man-Machine-Environment System Engineering*, pages 253–261. Springer.

[104] Zhang, X., Yang, Z., Sun, W., Liu, Y., Tang, S., Xing, K., and Mao, X. (2016). Incentives for mobile crowd sensing: A survey. *IEEE Communications Surveys Tutorials*, 18(1):54–67.

Publications

Journals

1. Liangzhi Li, Kaoru Ota, Mianxiong Dong, "Sustainable CNN for Robotic: An Offloading Game in the 3D Vision Computation," IEEE Transactions on Sustainable Computing (TSUSC), vol. 4, no. 1, pp. 67-76, 1 Jan.-March 2019.

2. Liangzhi Li, Kaoru Ota, Mianxiong Dong, "DeepNFV: A Light-weight Framework for Intelligent Edge Network Functions Virtualization," IEEE Network, vol. 33, no. 1, pp. 136-141, January/February 2019.

3. Liangzhi Li, Kaoru Ota, Mianxiong Dong, "Human in the Loop: Distributed Deep Model for Mobile Crowdsensing," IEEE Internet of Things Journal (IOTJ), vol. 5, no. 6, pp. 4957-4964, December 2018.

4. Liangzhi Li, Kaoru Ota, Mianxiong Dong, "Deep Learning for Smart Industry: Efficient Manufacture Inspection System with Fog Computing," IEEE Transactions on Industrial Informatics (TII), vol. 14, no. 10, pp. 4665-4673, October 2018.

5. Liangzhi Li, Kaoru Ota, Mianxiong Dong, "Humanlike Driving: Empirical Decision-Making System for Autonomous Vehicles," IEEE Transactions on Vehicular Technology (TVT), vol. 67, no. 8, pp. 6814-6823, August 2018.

6. Liangzhi Li, Kaoru Ota, Mianxiong Dong, Wuyunzhaola Borjigin, "Eyes in the Dark: Distributed Scene Understanding for Disaster Management," IEEE Transactions on Parallel and Distributed Systems (TPDS), vol. 28, no. 12, pp. 3458-3471, December 2017.

7. Liangzhi Li, Kaoru Ota, Mianxiong Dong, "When Weather Matters: IoT-based Electrical Load Forecasting for Smart Grid," IEEE Communications Magazine, vol. 55, no. 10, pp. 46-51, October 2017.

Proceeding of International Conference

1. Liangzhi Li, Kaoru Ota, Mianxiong Dong, Christos Verikoukis, "Enabling 60 GHz Seamless Coverage for Mobile Devices: A Motion Learning Approach," IEEE Global Communications Conference (IEEE GLOBECOM 2018), Abu Dhabi, UAE, December 9-13, 2018.

2. Liangzhi Li, Kaoru Ota, Mianxiong Dong, "Everything is Image: CNN-based Short-term Electrical Load Forecasting for Smart Grid," The 11th International Conference on Frontier of Computer Science and Technology (FCST-2017), Exeter, UK, June 21-23, 2017.