HYBRID MOBILE COMPUTING FOR CONNECTED AUTONOMOUS VEHICLES

JIAN WEI

Doctor of Philosophy

April 2018

ASTON UNIVERSITY

ASTON UNIVERSITY

Hybrid Mobile Computing for Connected Autonomous Vehicles

Doctor of Philosophy

Jian Wei
April 2018

Summary

With increasing urbanization and the number of cars on road, there are many global issues on modern transport systems, Autonomous driving and connected vehicles are the most promising technologies to tackle these issues. The so-called integrated technology connected autonomous vehicles (CAV) can provide a wide range of safety applications for safer, greener and more efficient intelligent transport systems (ITS). As computing is an extreme component for CAV systems, various mobile computing models including mobile local computing, mobile edge computing and mobile cloud computing are proposed. However it is believed that none of these models fits all CAV applications, which have highly diverse quality of service (QoS) requirements such as communication delay, data rate, accuracy, reliability and/or computing latency.

In this thesis, we are motivated to propose a hybrid mobile computing model with objective of overcoming limitations of individual models and maximizing the performances for CAV applications. In proposed hybrid mobile computing model three basic computing models and/or their combinations are chosen and applied to different CAV applications, which include mobile local computing, mobile edge computing and mobile cloud computing. Different computing models and their combinations are selected according to the QoS requirements of the CAV applications. Following the idea, we first investigate the job offloading and allocation of computing and communication resources at the local hosts and external computing centers with QoS aware and resource awareness. Distributed admission control and resource allocation algorithms are proposed including two baseline non-cooperative algorithms and a matching theory based cooperative algorithm. Experiment results demonstrate the feasibility of the hybrid mobile computing model and show large improvement on the service quality and capacity over existing individual computing models. The matching algorithm also largely outperforms the baseline non-cooperative algorithms.

In addition, two specific use cases of the hybrid mobile computing for CAV applications are investigated: object detection with mobile local computing where only local computing resources are used, and movie recommendation with mobile cloud computing where remote cloud resources are used. For object detection, we focus on the challenges of detecting vehicles, pedestrians and cyclists in driving environment and propose three methods to an existing CNN based object detector. Large detection performance improvement is obtained over the KITTI benchmark test dataset. For movie recommendation we propose two recommendation models based on a general framework of integrating machine learning and collaborative filtering approach. The experiment results on Netflix movie dataset show that our models are very effective for cold start items recommendation.

Keywords: *CAV, QoS, Hybrid Mobile Computing, Object Detection, Recommendation System*

Acknowledgements

First of all, the deepest gratitude goes to my supervisors, Dr. Jianhua He and Dr. Zuoyin Tang, for their constant research guidance through my entire PhD journey. During the writing of this thesis, they have carefully read the manuscript and provided professional instructions and suggestions, without which the completion of this thesis would be impossible.

Next I would like to thank all the colleagues and teachers at Aston University and Shanghai Jiao Tong University, who once offered me help in the past three years. I am greatly indebted to Dr. Kai Chen, Dr. Yi Zhou and Dr. Qu Zhou, who have provided crucial support in my academic studies.

I also feel grateful to my friends and classmates who have spent time listening to me and giving me advice when I met problems. Special thanks to Xiaotian Lu, Wen Cui, Yao Deng, Haoyuan Xin and Zhi Qiao for their efforts on proofreading my thesis.

Finally my sincere thanks are given to my family for their patience and encouragement.

Contents

Acronyms

A-Fogs Ad-hoc fogs.

AaaS Analytics as a service.

ACF Aggregated channel features.

ACRA Admission control and resource allocation.

ADAS Advanced driving assistance systems.

AI Artificial intelligence.

ALS Alternating least squares.

AP Average precision.

AWS Amazon web services.

BFS Breadth first search.

BN Batch normalization.

CAV Connected autonomous vehicles.

CB Content-based.

CCS Complete cold start.

CDL Collaborative deep learning.

CF Collaborative filtering.

CNN Convolutional neural networks.

CPU Central processing unit.

CS Cold start.

CTR Collaborative topic regression.

CV Connected vehicles.

D-Fogs Dedicated fogs.

D2D Device to device.

DAE Denoising autoencoder.

DBB Deconvolution building blocks.

DPM Deformable parts model.

DSRC Dedicated short radio communications.

ETSI European Telecommunications Standards Institute.

FC Fully connected.

FCW Forward collision warning.

GPS Global positioning systems.

GPU Graphics processing unit.

HMCM Hybrid mobile computing model.

HOG Histogram of oriented gradients.

IaaS Infrastructure as a service.

ICF Integral channel features.

ICS Incomplete cold start.

IoU Intersection of union.

ITS Intelligent transportation systems.

LDA Latent Dirichlet allocation.

LR Logistic regression.

MANET Mobile ad-hoc networks.

MCC Mobile cloud computing.

MF Matrix factorization.

MIMO Multiple input multiple output.

MS-CNN Multi-scale CNN.

NCS Non-cold start.

NMS Non-maximal suppression.

OPBB Object proposal building blocks.

PaaS Platform as a service.

PCWS Pedestrian collision warning systems.

PMF Probabilistic matrix factorization.

QoS Quality of Service.

RDD Resilient distributed dataset.

RoI Region of interest.

RPN Region proposal network.

RRC Recurrent rolling convolution.

SA Simple average.

SaaS Software as a service.

SAE Society of Automotive Engineers.

SDAE Stacked denoising autoencoder.

SVD Singular value decomposition.

SVM support vector machine.

TC Triangle counting.

TF-IDF Term frequency-inverse document frequency.

ToA Top of All.

TOPS Trillion operations per second.

ToU Top-of-User.

V2I Vehicle to infrastructure.

V2N Vehicle to networks.

VANET Vehicle ad-hoc networks.

VCC Vehicular cloud computing.

VM Virtual machines.

WCC Weakly connected component.

WLAN Wireless local area network.

List of Figures

List of Tables

1 Introduction

1.1 Autonomous Driving and Connected Vehicles

A rapid growth of worldwide vehicle stock was witnessed in the last two decades. According to [1], the total number of vehicles was reported to be 800 million in 2002, and was expected to rise to two billion by 2030. However, the fast increasing number of vehicles also exacerbates major global issues facing by the transport systems, such as big number of fatal road accidents, heavy productivity losses and fuel waste due to congestion, road pollution and large contribution to green house emissions. In 2016, more than £30 billion were wasted in traffic jams in UK. The annual global deaths caused by road crashes is nearly 1.3 million, and 20-50 million people were injured or disabled.

Autonomous driving and connected vehicles are two of the most promising technologies to tackle the above challenges faced by modern transport systems. The so-called integrated technology connected autonomous vehicles (CAV) can provide a wide range of safety applications, transport efficiency applications and entertainment applications. Next the technologies of autonomous driving and connected vehicles are introduced respectively. The research challenges on mobile computing for CAV are analyzed. Then the research objectives and contributions of this thesis are presented.

1.1.1 Advanced Driving Assistance Systems

According to the study by National Highway Transportation and Safety Administration, 93% of the road accidents are due to human error, with driver inattention being the primary cause. Thus autonomous vehicles is an ambitious vision to address this problem by "driving" themselves without humans control and intervention.

Autonomous driving was developed on top of the advanced driving assistance systems (ADAS). ADAS is a type of technologies designed to support driving and reduce accidents. It is moving forward fast globally, with an estimated worth of $70 billion by 2030. Equipped with different sensing systems and advanced data processing algorithms, ADAS can support driving and warn drivers of impending danger so that the driver can take corrective action, or even intervene on the driver's behalf. It can provide many enhanced safety features such as blind spot detection and forward collision warning (FCW). While present ADAS use different sensing systems such as ultrasonic, radar, video, infrared and laser radar, the most common sensing solutions are based on video and radar. There are many successful ADAS products and applications, including FCW systems and pedestrian collision warning systems (PCWS) [2]. However, even

with powerful computing and sensing devices, the current sensing systems can't provide satisfactory performance expected by road safety standards. According to the latest KITTI vision benchmark results [98], which is the largest public dataset dedicated to ADAS and autonomous driving benchmarking, even with graphics processing unit (GPU) computer and deep learning neural networks, the pedestrian and cyclist detection accuracy is only 72.4% and 67.5%, respectively, which is still far away from the expected level. It is noted that fully autonomous driving has higher level of automation than ADAS, but it also shares the limitations of ADAS. The strong limitation of autonomous driving is underlined by the Tesla fatal crashes.

1.1.2 Connected Vehicles

Connected vehicles (CV) is another accident avoidance technology, which is designed to transmit basic safety information between vehicles to facilitate warning drivers of impending crashes [4]. It mainly relies on communications and global positioning system (GPS) sensors. CV technology was extended somehow from mobile ad hoc networks (MANET) and vehicle ad hoc networks (VANET), but with more focus on collaborative driving safety. In the early days of CV research and development, IEEE 802.11p which was developed by IEEE was the main standard used for CV. There are two major CV systems based on IEEE 802.11p, including dedicated short radio communications (DSRC) promoted by the United States [5] and ITS-G5 developed by European Telecommunications Standards Institute (ETSI) [6]. Using on-board radio communication, mainly DSRC technology, messages can be exchanged with information about host vehicle's speed, heading, brake status and other information to other vehicles. It can achieve nearly twice the detection range of current ADAS and overcome the non-line of sight object detection problem in the ADAS products, which helps vehicles perceive threats sooner. US Department of Transportation issued a proposed rule that all new cars starting in 2021 are equipped with vehicle communication devices. Although CV holds great potential in avoiding crashes and saving lives, there are still many challenges faced by the applications of CV after more than a decade development. Technically, DSRC safety channel is prone to safety message congestion. Unreliable and delayed message delivery can make the safety messages useless and even generate adverse safety consequence. There are increasing research and standardization efforts on 5G cellular technologies for vehicle communications, which can offer lower latency and higher data rate. 4G/5G cellular technology has wide area coverage and can provide good quality of service (QoS). Device to device (D2D) [7, 8] is widely accepted as a candidate of 5G technology for vehicle. However, existing 5G D2D developments are focused on unicast communication, which does not meet the broadcasting needs of road safety applications. Another strong limitation of CV is that it is only effective when a critical mass of vehicles on the road can send and receive safety messages.

1.1.3 Connected Autonomous Vehicles

It is noted that the local sensors based ADAS technology and radio communication based technology have their own advantages and disadvantages. ADAS is a proactive road safety technology, using mainly the host vehicle's own computing and sensing resources. No collaboration is needed to achieve ADAS functionalities. On the other hand, CV is a passive road safety technology, which is completely relying on message exchange with neighbor vehicles to achieve driving context awareness. As both technologies are advancing with great paces and they can work together without major technical obstacles, there are increasing research development interests on the integration of connected vehicles and autonomous driving, leading to the term connected autonomous vehicles (CAV).

One of the major drives for CAV is that autonomous driving and connected vehicles technologies can complement each other and are really needed to work together to achieve safer, greener and more efficient ITS, in order to tackle the ITS challenges mentioned beforehand.

Another major drive comes from the increasing popularity and market potentials of autonomous driving. In the past several years, huge attention has been paid to autonomous driving from both industries and research communities. Many big automakers and IT companies have announced the plans to build fully autonomous cars in the next two or three years, including Tesla, Volvo, Google, etc. The huge investments and the technology advances such as those on deep learning are pushing the autonomous driving closer to reality.

There are five automation levels defined by Society of Automotive Engineers (SAE) for autonomous driving:

- level 1 automation: vehicle is under human control except for some driving assist tasks like small steering or acceleration.
- level 2 automation: vehicle can automatically perform driving actions but the driver must stay alert and monitor the environment at all times.
- level 3 automation: driver doesn't need to monitor the environment but must be ready to control the vehicle at all times with notice.
- level 4 automation: vehicle can automatically drive itself almost all the time, but driver needs to take over under certain traffic and environment conditions such as unmapped areas or severe weather .
- level 5 automation: vehicle is capable of performing all driving functions under all conditions

According to the automation levels above, the ADAS technologies may be sufficient to achieve level 1 and level 2 automation. But to achieve level 3 and above automation, connected vehicles technology is a necessity.

1.2 Applications and Research Challenges for CAV

While CAV holds great promises for future safer and greener ITS, there are also many research challenges faced by CAV. In this section, a wide range of applications of CAV are presented. Then the major technology challenges are discussed. More specifically, mobile computing challenges for CAV technology are analyzed.

1.2.1 Applications of Connected Autonomous Vehicles

CAV can support a wide range of applications, which can be separated into three categories: driving safety applications; transport efficiency applications; entertainment and comfort applications.

Driving safety applications are the main driving forces for CAV with main goals of reducing road accidents and improving driving safety. As introduced earlier, they can be supported by either autonomous driving or connected vehicles separately, or by both technologies. Road safety applications are based on environment sensing via local sensors or collaborative context awareness messages exchange. With the development of computing engine and data analytic algorithms, vehicles could efficiently detect surrounding objects such as lane, other vehicles, pedestrians and traffic signs and make decisions accordingly. Example safety applications include forward collision detection, forward collision warning, emergency braking, rear collision warning, lane departure warning and lane keeping.

Transport efficiency applications are mainly used to improve mobility management and road efficiency, reduce transport congestion and cost. Information such current traffic, road works and driving environment can be exchanged among the vehicles and transport infrastructure, to achieve transport efficiency. Example transport efficiency applications include emergency vehicles notification, coordinated traffic lights, live traffic conditions, high definition map creation and update;

Entertainment and comfort applications involve the entertainment and comfort of the driver and passengers. The applications are mainly supported through connected vehicles technology, such as vehicle to infrastructure (V2I) and vehicle to networks (V2N) technologies. Example entertainment applications include music streaming, video streaming, online gaming, virtual reality, etc.

1.2.2 Research Challenges

While CAV technology holds great potential, there are still many technical and non-technical challenges faced by CAV. With rapid development of autonomous driving and connected vehicles technologies, various autonomous driving systems have been developed. There are two representative technology frameworks from Mobileye and Baidu, which also partially highlight the major technology challenges.

In the Mobileye autonomous driving technology framework, they identified three technology pillars, which include sensing, mapping and driving.

1.2.2.1 Sensing

Sensing technical pillar is mainly responsible for developing environmental model and 360 degree of driving environment awareness. It relies on local sensors and radio communications and advanced computing algorithms (such as deep learning algorithms for object detection). while there are increasing efforts in improving the sensing accuracy and reducing the cost of sensors, how to fuse the information from various type of sensors is still a big challenge. In addition, providing expected latency, data rate and reliability by radio communication technologies such as IEEE 802.11 are still very challenging for CAV applications.

1.2.2.2 Mapping

Mapping technical pillar is responsible for developing and maintaining high-definition maps for autonomous driving. The existing GPS technology has an accuracy around 5 to 10 meters, which is far away from the expected positioning accuracy of centimeters for autonomous driving. How to achieve accurate, scalable, real time and low cost high definition map is a big challenge for CAV. Mobileye proposed a crowd sourcing approach, called RoadBook, in which users collect real time information and send to a server for processing. The server processes the received road maps and send the updated high definition maps to the drivers.

1.2.2.3 Driving

The Driving technological pillar is responsible for assessing threats, planning maneuvers and negotiating the multi-agent games of traffic. The game players may be human drivers or computer drivers. While reinforcement learning technologies are applied for autonomous driving decision making and collaboration, such research is still on the early stage and there is a long way to go before it becomes mature.

The technology framework of Baidu is presented in their open source system Apollo [3]. In addition to the three technical pillars proposed in Mobileye's technical framework, Apollo covers more technical general computing and operation system. In the Apollo, the whole CAV system is divided into vehicle platform, hardware platform, software platform and cloud service platform. The Apollo autonomous driving framework covers broad areas, but the core technologies are similar to those discussed in the Mobileye's technology framework.

Apart from the above major technological challenges, there are also many other technical and non-technical challenges, such as security issue, privacy issue, reliability issue. Connected vehicles is a necessary component of autonomous vehicles with high level driving automation. However, a big concern with CAV is the security as hackers may get access and control the autonomous vehicles. In addition, privacy is another big issue for CAV. For example, sense data gathered from other cars such as location, mobility pattern could be collected by other vehicles or hacked and be used for personal or commercial reasons. Reliability is another major issue for CAV. From the component wise, the cars, sensors and radio communication devices all could malfunction. From the system wise, the hardware system, the mapping system and

computing system may also fail. Any failure of the components or sub-systems could lead to fatal consequences, which is much more severe compared to that in human-driven vehicles.

1.3 Research Motivations and Objectives

1.3.1 Research Motivations

In the previous section, the high level technical challenges including sensing, mapping and driving are discussed for autonomous driving applications. In addition to the autonomous driving (mainly relevant to the driving safety applications of CAV), there are other technical challenges faced by CAV, for example, big data analytics and computing for transport efficiency and entertainment applications supported by CAV. In this thesis mobile computing, which is one of the low level technical challenges for CAV, is investigated.

Mobile computing is the cornerstone of CAV, as all the CAV applications are heavily relying on mobile computing. The core driving safety applications such as forward collision detection and warning systems require very high accuracy object detection, which should be ideally achieved by deep learning techniques. But there are strong limitations on the computation capacity in the vehicles. Next we discuss the challenges on the mobile computing faced by the CAV applications.

Firstly, as we discussed in the previous section, there are a wide range of applications supported by CAV, including driving safety, transport efficiency and entertainment applications. These CAV applications have highly diverse QoS requirements on computing, communications and sensing. Driving safety applications require very high computation speed, low latency and they generate very high volume of sensing data, which is extremely hard for the current radio technologies to support. For example, collision avoidance applications require extremely high object detection accuracy and low computation latency, high definition map applications require high speed communication and fast data analytics. On the other hand, the non-safety CAV applications have relatively less strict requirements on computing and communication latency, and communication data rates. For example, video streaming and virtual reality applications can accept relatively higher latency but still require high speed communication.

Secondly, although the computing power of mobile devices has increased significantly in the last decades, it is noted that the computing power of devices in vehicles is still highly limited compared to the desktop GPU computers. Due to the mobility, computing and communication capabilities of vehicles, it is still very challenging to support all these CAV applications, which have diverse QoS requirements (such as communication delay, data rate, accuracy, reliability and/or computing latency).

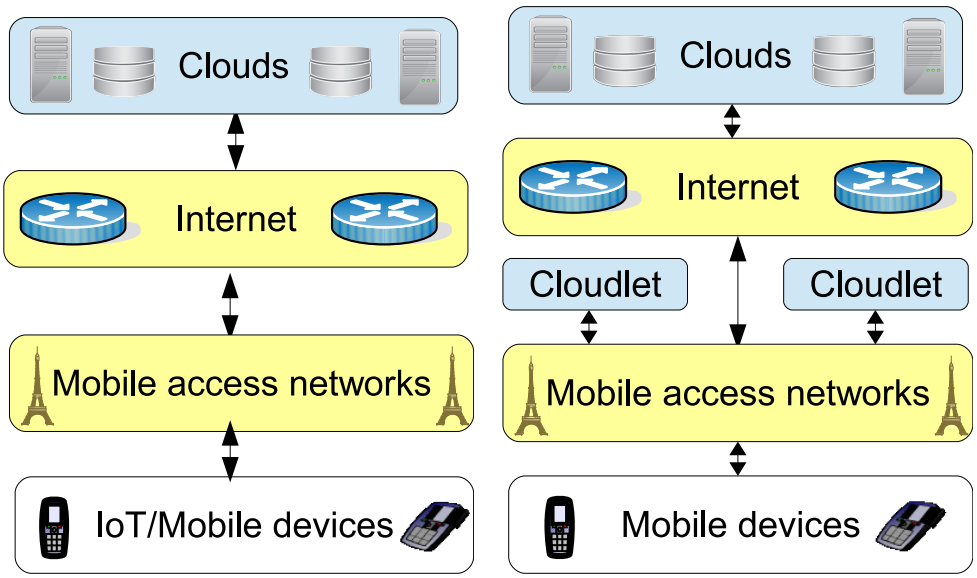
How to support the CAV applications with the limited computing and communication power of the vehicles is still an open research issue, which requires significant research and development efforts. While there have been wide research efforts on mobile computing, it is believed that none of mobile computing models fits all the CAV applications.

Next we presented a survey of relevant mobile computing models for CAV applications. Then the research objectives are presented.

1.3.2 Existing Mobile Computing Models

1.3.2.1 Mobile Cloud Computing

Within the last several years we witnessed an explosive growth of mobile smart personal devices (e.g. smart phones and tablets), which raised huge demands on mobile applications and services. However the limitations of mobile devices on computation resources (e.g. CPU, storage) and energy prevent advanced and complex mobile applications and services to be run at these devices directly. Mobile cloud computing (MCC) is an efficient solution, which integrates the general cloud computing to mobile computing environment to bring the rich cloud computation resources to the mobile users [15]. It aims to overcome the computation resource problem of mobile devices to run rich mobile applications with improved quality of experiences. Unrestricted mobile applications can be enabled by ubiquitous access to the computation and storage resources at the remote clouds. Mobile cloud computing is expected to be a dominant way for mobile application operations. Fig. 1.1(a) presents a system architecture for MCC. The main research problems on MCC include computing task offloading and minimizing network bandwidth and energy consumption [15]. As the computing devices installed in vehicles are also characterized by the features of limited computation and energy resources, the MCC solution is applicable to some CAV applications and services as well.



(a) MCC. (b) Cloudlet.

Figure 1.1: Mobile cloud computing and cloudlet architecture.

1.3.2.2 Mobile Edge Computing

Several edge computing models have been proposed in the literature, which are mainly designed to tackle the data analytics problems in the cloud computing based solution. The principle is to move computing resources and analytics services closer to the things where data is generated. In this following we introduced relevant edge computing models and technologies for data analytics services.

Cloudlet and its kind are mobility enhanced small scale clouds, which are introduced to address the problems faced by MCC, by moving clouds to the edge of the Internet to support real-time mobile applications. Cloudlet is an extension of and supplementary to the cloud [16]. A cloudlets based computing system architecture is presented in Fig. 1.1(b), where cloudlets locate in the middle of the path from mobile devices to clouds. Usually cloudlets are deployed one wireless hop away from the mobile devices, e.g. connected to the cellular base stations or WiFi access points. In this way cloudlets can provide powerful computing resources to support interactive mobile applications.

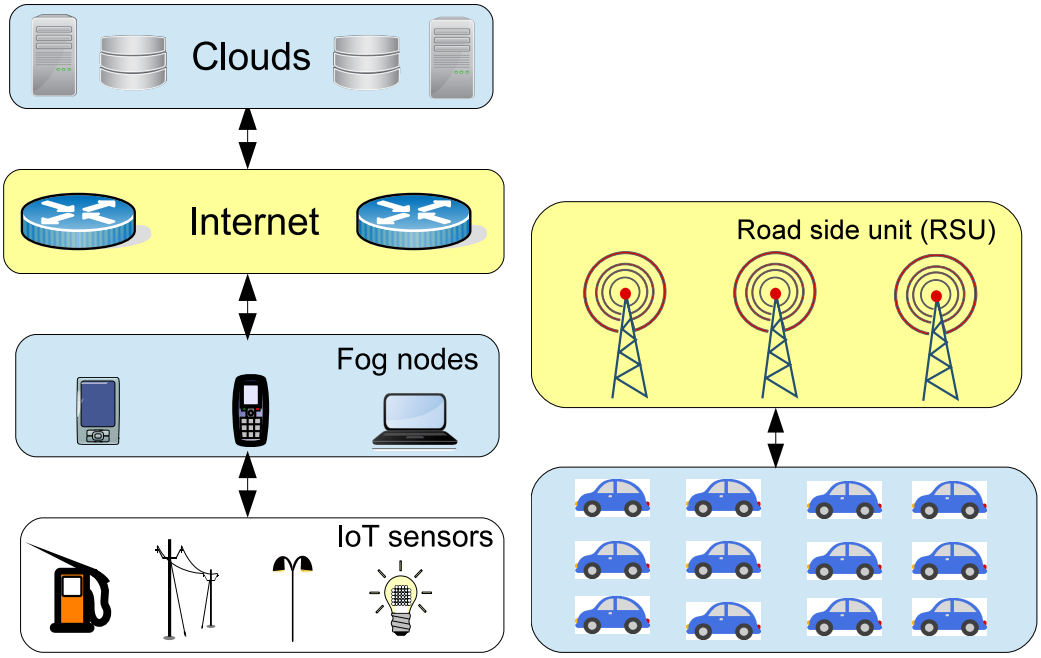
Apart from the technical challenges faced by MCC such as task offloading and admission control, user mobility poses extra challenges to cloudlets such as handoff of virtual machines. Clearly cloudlets will benefit large scale data analytics services. However, cloudlets are usually deployed by third-party and deployed in a large number of locations, which may not be close enough to the end devices where a large volume of data is generated and analytics services are needed.

Fog computing is originally proposed by Cisco specifically for Internet of Things (IoT) applications [12, 17]. The idea is similar to cloudlet by moving computing closer to the places where data is generated. However, in the fog computing model any device with computing, storage and network connectivity (the so-called fog node) can join the data analysis tasks, such as smart phones, laptops, video surveillance cameras, routers and WiFi access pointers. This is different from the cloudlets where dedicated small scale data centers are deployed by third-parties. A system architecture of fog computing is presented in Fig. 1.2(a).

1.3.2.3 Vehicular Cloud Computing

Vehicular cloud computing (VCC) is a computing platform proposed to harness the computing and sensing resources of interconnected vehicles [18–20]. Its main interests focus on the contents from vehicles and sensors within a local area. VCC is a variant of mobile cloud computing within which services are produced, maintained and consumed [18]. A system architecture of VCC is presented in Fig. 1.2(b). VCC has a deep root on VANET. M. Whaiduzzaman et al present a survey on VCC with discussion on system operations (such as cloud resource discovery, cloud formation and release) and design principles [19].

It is noted that although computing resources sharing is claimed as an important function of VCC, we are not aware of reported experiments that use these resources and study the feasibility for data analytics services. The main concerns of VCC are on the vehicle traffic management and



(a) Fog computing. (b) VCC.

Figure 1.2: Fog computing and vehicular cloud computing.

driving safety applications. The application and capabilities of VCC for IoT data analytics and the impact of wireless connection bandwidth and vehicle mobility have not been investigated.

1.3.2.4 Mobile Local Computing

Apart from the above research efforts of utilizing external computing resources (e.g., in the clouds, network edge or neighbor vehicles), there is a trend of increasing local computing power and using it for more advanced artificial intelligence (AI) computing tasks. Examples include the AI-integrated circuit in the latest Huawei smart phones Mate 9 for applications such as face detection, and the NVidia embedded GPUs for autonomous driving such as Jetson TX1. Compared to central processing unit(CPU), the NVidia GPUs are very good for matrix-matrix multiplies and also have very high bandwidth to memory, which makes them efficient in executing computer vision and deep learning tasks. This technical trend may be explained by the increasing computing power of mobile devices following Moore rule, and the necessity of keeping some advanced AI computing tasks local due to the real time computing requirement of autonomous driving applications.

1.3.3 Research Objectives

According to the introduction on the CAV applications and the mobile computing models, it can be found that there is no mobile computing model which can fit all the CAV applications. For the CAV safety applications, they have very strict computing latency requirement. For example, for the camera vision based forward collision avoidance applications, the computing system should be able to detect objects in the images in the speed of 15 or more frames per

second. Such computing tasks are unlikely to be performed in the edge or remote clouds, which has too high communication delay, not to mention the needed computation time. On the other hand, for the collaborative safety and transport efficiency applications, it is neither practical nor necessary to transport the vehicle data to every other vehicles. Performing data analytics and global network control computing tasks at the edge or remote clouds is a much better solution, which have virtually unlimited computing and storage resource.

In order to address the mobile computing problems faced in CAV systems, in this thesis we are motivated to propose a hybrid mobile computing model to deliver the CAV applications. In the hybrid mobile computing model the computing resources at the local host and external entities are efficiently utilized according to the CAV application requirements. The main objectives of this thesis are summarized as below:

- Develop a hybrid mobile computing model for CAV applications, which should help achieve the expected computing performances such as low computing latency, high scalability and low communication bandwidth consumption. To achieve the objectives, the three mobile computing models (namely mobile local computing, mobile edge computing and mobile cloud computing) and/or their combinations should be chosen and applied to different CAV applications according to the computing performance requirements of these CAV applications.

- Design efficient algorithms to improve the performance of the hybrid mobile computing model. In the hybrid mobile computing model, there are many vehicles running various CAV applications and many local and external computing resources. There are strong requirements on the algorithms to determine which computing jobs should be run locally or externally, and if to be run externally, it is further needed to determine which external computing center to run the job and allocate how much computing resources for this job.

- Design and evaluate computing algorithms for example CAV applications with various mobile computing models, to demonstrate the effectiveness of the hybrid mobile computing model.

- Develop system level experiment and/or simulations to evaluate the effectiveness of the hybrid mobile computing model and the individual mobile computing models, and facilitate the computing system planning and optimization.

1.4 Thesis Outline and Contributions

Following the research objectives, effective research works have been conducted and interesting results were obtained. Next the thesis outline and the research contributions of this thesis are summarized.

In Chapter 2, a hybrid mobile computing model is proposed, which is applicable to CAV applications as well as to general data analytics. In the proposed hybrid mobile computing

model, not only the local computing resources and external cloud computing resources, but also opportunistic computing resources from neighbor vehicles or other mobile devices, are considered to perform CAV computing tasks. To address the QoS requirements of the CAV computing tasks and the related cost, innovative computing job admission control, offloading algorithms and resource allocation schemes are designed and evaluated. The main research work and contributions of this chapter are summarized below.

- We proposed a hybrid mobile computing model which includes multiple interconnected ad hoc Fogs (A-Fogs), dedicated Fogs (D-Fogs) and clouds for large scale mobile computing services. With opportunistic computing resources and distributed computing engines, the proposed framework can help mitigate the problems faced by cloud only or local computing only solutions. In addition the interconnection and cooperation among the fogs can improve resource utilization and edge analytics capacity.

- We developed a framework for QoS aware service and resource management with opportunistic computing and communication resources, which is vital for comprehensive evaluation and improvement of the hybrid mobile computing model. An optimization problem of analytics job admission control and resource allocation (ACRA) was formulated with objective of maximizing the service utility for edge analytics. The computing and communication resources were jointly considered in the optimization problem. Workload models with distributed computing engine Spark [63] were also created for the ACRA process from benchmark experiments over A-Fogs and D-Fogs.

- As the ACRA optimization problem is NP-complete with very high computation complexity, two baseline non-cooperative ACRA algorithms were proposed, and a QoS aware matching theory based cooperative algorithm was designed to solve the problem. A system level simulator was developed for performance evaluation. It is observed that in the hybrid mobile computing model A-Fogs is very effective in increasing service utility and service cost reduction, while D-Fogs plays a key role in maintaining a very low job blocking probability. The results demonstrated the feasibility and efficiency of the framework for large scale edge analytics and superior performance of matching based ACRA algorithm.

The works in this chapter form the basis of the following publication:

- J. He, J. Wei, K. Chen, Z. Tang, Y. Zhou and Y. Zhang, "hybrid Fog Computing with Large-scale IoT Data Analytics for Smart Cities." IEEE Internet of Things Journal, No. 99, July 2017.

- J. Wei, J. He, K. Chen, Y. Zhou, "Benchmarking of Distributed Computing Engines Spark and GraphLab for Big Data Analytics." In Proc. IEEE BigDataService, 2016.

- J. Wei, J. He, Y. Zhou, X. Fu, Y. Yuan, Y. Zhang and Z. Tang, "QoS Aware Internet of Things Edge Analytics Services with Hybrid Fog Computing Network." submitted to IEEE

Transactions on Services Computing (under review), 2017.

In chapter 3, deep learning based object detection with mobile local computing was investigated. For CAV driving safety applications, accurate and real time detection of surrounding objects such as lane, vehicles, pedestrians and traffic signs is crucial for self driving and road safety. This chapter is focused on the design and optimization of detecting vehicles, pedestrians and cyclists with mobile local computing. Despite fast growth of convolutional neural network (CNN) in object detection over datasets with a large number of object classes, the popular CNN detectors including Faster-RCNN [103] and SSD [104] do not perform very well for object detection in driving environments. We proposed several following methods to an existing multi-scale CNN (MS-CNN) model to improve object detection performance for CAV driving safety applications. The research contributions of this research work are summarized below:

- Deconvolution of CNN features was applied at smaller feature output scales, which was further fused with features at larger feature output scales, to provide richer context for object detection at individual feature output scale. Such a method can effectively address the large object scale variation challenge in CAV driving safety applications.
- In most of existing CNN detectors, non-maximal suppression (NMS) method is used for suppression of overlapping object proposals. With such process there is very little chance to properly detect occluded objects. But in driving environments occluded objects are normal and are potential driving hazards. To address the object occlusion challenge soft-NMS was applied at object proposals from different feature output scales to strike a balance on the number and quality of object proposals.
- In the existing CNN detectors, default anchor boxes with certain sizes are used to generate object proposals. In the driving environment the interested objects have strong features in shape, for example, the width of a car should not exceed lane width. We measured the aspect ratio statistics of objects from KITTI training samples and found proper anchor box settings by exploiting the statistics for better object localization and prediction.

The proposed CNN methods were individually and jointly evaluated with various image input sizes by extensive experiments over KITTI benchmark dataset. Good detection performance improvement was reported with both individual and combined CNN methods. The research achievements in this chapter were presented in the following research paper:

- J. Wei, J. He, Y. Zhou, K. Chen, Z. Tang and Z. Xiong, "Enhanced Object Detection with Deep Convolutional Neural Networks for Advanced Driving Assistance." submitted to IEEE Transactions on Intelligent Transportation Systems (under review), 2017.

In chapter 4, a representative CAV entertainment application of movie recommendation for video streaming with mobile cloud computing was investigated. Online recommendation is a

typical service for mobile cloud computing. The past user ratings on items and auxiliary information are stored in the cloud. When a user's request is received, the recommendation system recommends relevant items to the interested user. The recommendation process is performed in the cloud and only final results are sent to the user via wireless access network.

One of the major challenges faced by online recommendation for CAV entertainment applications is the so-called cold start (CS) problem, where the recommendation system may not have the enough users' ratings on newly added items. To tackle the problem, two integrated recommendation models were proposed, in which item features are learned from a deep learning architecture SDAE [89] using the descriptions of items retrieved online, and then these features are exploited and integrated into the timeSVD++ collaborative filtering (CF) model [75]. The research contributions are summarized as follows.

- A general framework of integrating the CF approach and machine learning algorithms was developed to improve recommendation performance for CS items. In the proposed models, content features extracted from content descriptions (such as movie plots) by deep learning neural networks were used as the key item factor vectors and approximated by the item factor vectors in the recommendation model. The content features are not only used loosely to determine item similarity as done in the existing hybrid approaches for CS items, but also become key component of the recommendation models, which affects both the training of the models and prediction of the unknown ratings for CS items.

- The framework of integrating the CF approach and machine learning algorithms for CS item recommendation is general. Various CF approaches and machine learning algorithms can be used for general recommendation systems. The key integration point is on the extraction of item features by machine learning algorithms and embedding the item features into the CF recommendation models.

- Based on the general framework specific system design and models were presented, Application of the models to Netflix movie recommendation with nearly 100 million ratings was investigated. The experiments results showed that tight coupling of the CF approach and content based approach for recommendation was feasible and effective.

The research achievements in this chapter were presented in the following research papers:

- J. Wei, J. He, K. Chen, Y. Zhou and Z. Tang, "Collaborative filtering and deep learning based recommendation system for cold start items." Expert Systems with Applications, 69, 29-39, 2017.

- J. Wei, J. He, K. Chen, Y. Zhou and Z. Tang, " Collaborative filtering and deep learning based hybrid recommendation for cold start problem." In Proc. IEEE DataCom, pp. 874-877, 2016.

Finally this thesis is concluded in chapter 5. The future works are also discussed, which include the extension of the work carried out in this thesis and other new directions of CAV area.

2 QoS Aware Hybrid Mobile Computing Model

2.1 Introduction

With virtually unlimited computing and storage resource, clouds are thought to be the natural places for big data analytics and can provide easy management of CAV applications. However there are several problems faced by the cloud based solutions. Firstly, the clouds are usually set up at remote areas to reduce development and operation costs. The round trip delay could be a big issue for the CAV applications and services with fast response requirement. Secondly, with expansion of transportation systems and emerging big data from CAV applications transmitting all the data to clouds could cause congestion at network bottlenecks and incur huge networking cost.

Edge computing (including Cloudlet and fog computing) and hybrid cloud computing are proposed to mitigate the problems faced by the cloud based solution [12, 14, 16, 17]. By deploying extra computing resources and intelligence to the edge, fast response can be provided and transmission of unnecessary data to remote clouds is avoided. In the existing cloudlet and mobile edge computing based solutions [16], computing facilities are usually deployed by third parties at fixed locations, which may not be flexible and closer enough to the CAV applications. And the wireless access bottleneck problem still exists for the CAV data traffic. On the other hand, fog computing is gaining increasing research and development momentums. But in the original fog computing model fog nodes have very limited computing power and only perform simple analytics such as data aggregation and computing basic statistics.

In this chapter we propose a hybrid mobile computing model (HMCM), based on which data analytics service is developed and provisioned for CAV applications. HMCM is consisted of interconnected ad-hoc fogs (A-Fogs) with consumer computing devices, dedicated fogs (D-Fogs) with service operator devices at edge and remote clouds. Within the last several years we witnessed an explosive growth of road side devices (e.g. smart phones, video surveillance cameras and WiFi access pointers). In HMCM, the interconnected vehicles and nearby road side devices can be utilized to form small A-fogs. On the other hand, the number of small cell base stations and WiFi based home hotspots are also expected to grow fast. Dedicated computing resources can be deployed alongside these small base stations and home hotspots in addition to the macro cellular base stations and gateways to form dedicated fogs. With the potential cooperation among these fogs, more reliable and powerful data analytics services and more efficient

resource utilization can be achieved. And QoS aware algorithms with and without cooperation are proposed to address the challenging service and resource management problems in HMCM.

2.2 Related Works

Fog computing was originally proposed by Cisco [11, 12, 17]. The idea is similar to cloudlet by moving computing closer to the places where data is generated, but more specifically for IoT applications. In the Cisco fog computing model any device (the so-called fog node) with computing, storage and network connectivity can join the data analysis tasks for IoT devices. Fog nodes, fog aggregation nodes and clouds undertake different kinds of data analysis and storage tasks. Fog nodes take the most time-sensitive analysis tasks with response time of milliseconds to seconds. A revised definition for fog computing was proposed in [23] as a scenario where ubiquitous and decentralised devices communicate and potentially cooperate among them and with the network to perform storage and processing tasks without the intervention of third-parties. The fog nodes are similar to the simple computing devices in the A-Fogs discussed in this thesis, but they can only process very simple data analytics tasks.

In the last two years there was a sharp increase of published research papers on fog computing. The works cover the areas of fog computing model and network architecture, applications and services, and algorithm design. The original fog computing model was extended to embrace fogs with dedicated computing resources, which is actually very close to the edge computing model [24, 25]. In fact the line between fog computing with dedicated computing resources and edge computing is blurred. Apart from the general applications and services to IoT [17, 26, 27], fog computing was applied to a wide range of more specific applications. As an extension of fog computing model, concept and architecture of vehicle cloud and fog computing were studied in [28, 29]. Fog computing was also applied to cellular radio access networks to support functionalities such as access mode selection [30–33], where the fogs have dedicated computing resources but they are used exclusively for radio access networks. Specific applications of fog computing were reported for smart cities[35], traffic light control[36] and urban surveillance video stream processing[34]. Security of fog computing service was studied in [29, 36, 37]. It is noted that none of the above works investigated large scale data analytics services. While [38] studied fog computing based big data analytics service, only the concept with D-Fog computing is discussed without real experiments.

Service and resource management is a core problem of fog computing. In the literature it has been studied for fog computing and cloudlet systems in several papers [39–44]. An offloading algorithm was proposed for a mobile cloud system where a cloudlet is used to support mobile devices [14]. Energy efficiency is the main algorithm design objective. Deng *et al* proposed a solution to allocate workload between fog and clouds to achieve a balance between delay and power consumption [39, 40]. There is no interaction among the fogs and only centralized solution is considered. A framework for edge node resource management was proposed in [41] for provisioning and auto-scaling resources of one edge node. An online game use-case

was demonstrated with improved QoS. Zhang *et al* proposed a hierarchical game framework for fog computing resource management [43], where data services subscribers request general service from and negotiate service price with service operators, and service operators purchase computing resources from D-Fogs. Service delay is the key performance metric and the cost of resources (such as computing and networking) are not considered.

There are a few papers studying the impact of wireless communication in fog computing workload offloading. Chen *et al* investigated mobile task offloading to nearby mobile devices via D2D communication[42]. A graph matching solution was proposed to jointly allocate communication and computing resources, with objective of reducing the total task cost of energy consumption and execution time. Xiao *et al* investigated the power efficiency and quality of experience tradeoff for fog computing [44]. Fog node cooperation is considered for workload offloading. A distributed alternating direction method based algorithm was proposed to solve the workload allocation problem. In [44] each fog is a low cost device with highly limited computing power. A concept of opportunistic fog computing was presented in [45], which is consisted of virtual clusters of mobile devices. The opportunistic fog is similar to A-Fog, but the study is focused on conceptual work without any algorithms or experiments for fog computing.

2.3 Proposed Hybrid Mobile Computing Model

2.3.1 Overall Architecture

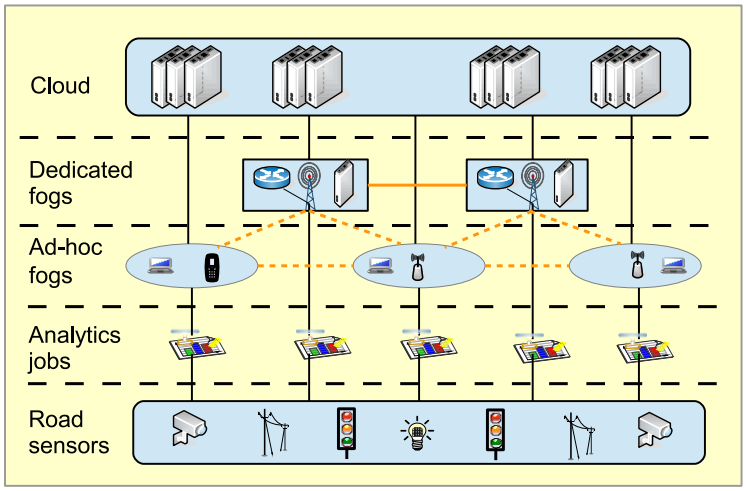


Figure 2.1: A hybrid mobile computing model architecture.

In Fig. 2.1 we present an architecture of HMCM, which includes multiple D-Fogs, A-Fogs and clouds. D-Fogs are similar to the concepts of cloudlet computing and mobile edge computing, which include the fogs supported by the dedicated routers and cellular network base stations. But A-Fogs are different which may be formed on demand as needed with distributed and opportunistic computing resources, such as smart phones, laptops and vehicles [20].

Sensors generate data and analytics jobs, which are managed by job owners. Job owners may be independent users outside HMCM or the fog nodes in the A-Fogs or D-Fogs. For simplicity,

A-Fogs, D-Fogs and clouds are all called computing center in this chapter, which offers data analytics services with different level of capacity. They can process the analytics jobs independently or cooperatively. Once the analytics jobs are processed, the outcome is sent back to the job owners or remote interested analytics service consumers.

There are several distinct features of HMCM, which include A-Fogs with opportunistic computing resources, interconnection of A-Fogs and D-Fogs and large scale data analytics services. These features jointly help achieve the objective of improving data analytics capacity.

With increasing number and power of mobile devices such as tablets, laptops and vehicles, there is a great opportunity to exploit the computing power of these devices in A-Fogs. The A-Fogs can be much closer to the data sources and communicate with data sources by low cost or free wireless communications technologies. The A-Fogs are replacing D-Fogs or clouds, but used in HMCM to complement the D-Fogs and clouds to provide better analytics services with reduced service costs and traffic congestion. Each A-Fog is formed by a cluster of consumer computing devices. There are two types of fog nodes in an A-Fog, i.e., fog master and fog worker. The functionalities of these nodes are illustrated by the functional models shown in Fig. 2.2. Devices with available computing resources can broadcast their resources availability to neighbors. Devices willing to act as fog master invite computing devices to form A-Fog and take the responsibility of A-Fog management, resource and computing jobs management, which are introduced next. Fog workers are responsible for sharing their computing resources, undertaking computing jobs, monitoring and reporting available computing and communication resources to fog masters etc. Each A-Fog has one primary fog master and optionally a number of secondary masters for reliability concern. Physically the fog masters may be located in the same devices where the fog workers are operated or independent devices such as D-Fog nodes. Next we introduce the main functional modules of the fog masters and the fog workers.

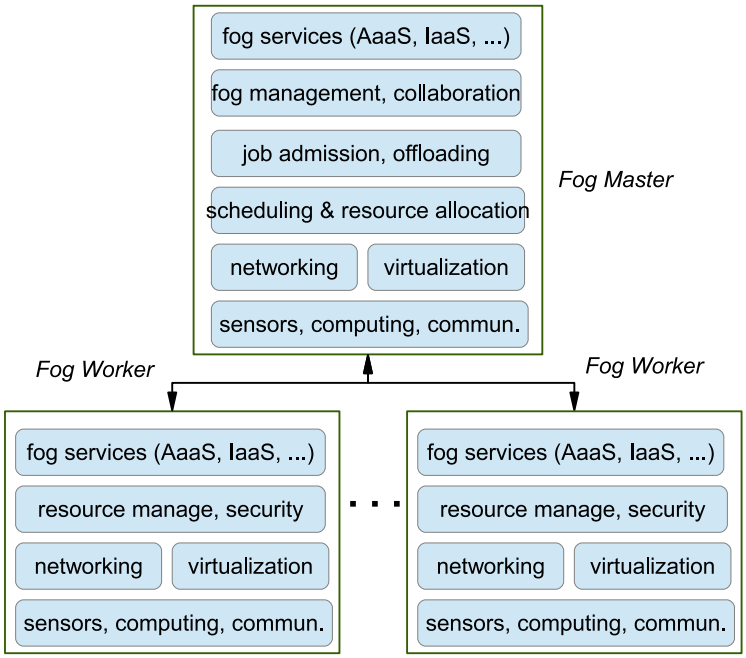


Figure 2.2: Function model for fog nodes.

2.3.1.1 Resource Module

The resource module is at the bottom of the function models for both fog master and fog workers. It represents physical resources of fog nodes, which may include sensing resources, computing resources and communication resources for connection to other fog nodes. It is noted that apart from WiFi technology, other communication technologies such as cellular radio and visible light communication technologies can also be used for fog node communication.

2.3.1.2 Networking and Virtualization Module

Networking is a critical part of fog, especially for the scenarios where fog nodes are mobile. In the fogs resource virtualization is optional but very important for the fog nodes which may have their own heavy computing tasks. With virtualization a part of computing resources can be reserved for the local computing tasks. And fog computing tasks can be run only in the isolated resources, by which local computing and security performance are ensured. The existing virtualization technologies can be applied with modifications for both A-Fogs and D-Fogs.

2.3.1.3 Fog and Resource Management Module

Fogs can be formed on demand and managed by fog master nodes. Each fog has a life cycle of formation, maintenance and release. Fog workers are responsible of monitoring and reporting computing resources and communication conditions to fog masters. Fog masters maintain the status of the available computing resources and communication conditions of the members in the fogs. Special incentive and reward schemes can be applied by fog masters to encourage interconnected devices to join fogs and share their unused computing resources.

2.3.1.4 Job Admission and Scheduling Module

When a computing job request is received (from CAV applications or other IoT applications), a fog master needs to assess the computing resources required to complete the job, and admit or reject the job request according to the available compute resources. If a job is accepted, it is scheduled to run over one or more fog workers depending on their available compute resources and network conditions. The fog master may communicate with other fog masters to jointly work on analytics tasks, or make decisions on offloading jobs to other fogs or remote clouds.

2.3.1.5 Services Module

There are three standard service models provided by traditional clouds, namely infrastructure as a service (IaaS), platform as a service (PaaS), software as a service (SaaS). If the fog nodes are static and have powerful computing resources, a large computation resource pool can be created for fogs and the standard cloud service models can be offered by the fogs. However, due to the limitations on the computing power, bandwidth of wireless connections and mobility of fog nodes, A-Fogs may not be ideal to provide these standard cloud computing services.

In this chapter we propose a new service model, analytics as a service (AaaS) for fogs. With AaaS model the users of CAV applications can request analytics services from fog masters. The masters analyze the analytics service request, choose the required analytics algorithm and computing engine, and assess the service requirements on computing and communication resources. If the service request is admissible, fog member nodes and computing resources are scheduled to provide the service. Multiple fog member nodes may work collectively with distributed computing engines to provide advanced analytics if needed.

Interconnection of the fogs is an important approach to improve the service reliability and the number of analytics jobs that can be processed by fogs at the edge, A-Fogs are usually connected by wireless communications. The emerging high speed and low cost wireless communications pave the way for cost effective and fast cooperation among the A-Fogs and D-Fogs. In addition there can be high speed wired connection among D-Fogs. The connections among the fogs are highlighted by dashed and solid orange lines in Fig. 2.1. The communication and cooperation among the fogs provide more options of job offloading, e.g., horizontal offloading among A-Fogs, among D-Fogs, on top of the traditional vertical offloading from fogs to clouds. It can effectively improve the data analytics service quality and resource utilization.

While using a cluster of computing devices can increase the aggregate computing resources of A-Fogs, distributed computing engines can be used to efficiently utilize the computing resources to process large data analytics jobs. Existing distributed computing engines such as Spark [63] have been very successful for big data analytics. They can be used for large scale data analytics jobs with or without critical real-time processing requirements. While a major motivation of HMCM design is providing large scale data analytics services, simple data analytics services such as data aggregation and filtering can be easily supported as well.

2.3.2 Key Enabling Technologies for HMCM

While cloud technology is mature and has already been widely used in numerous businesses, mobile edge computing and D-Fog are relatively new concepts. Nonetheless, there are strong technologies basis for fogs and some practical applications such as mobile caching at cellular base stations have already been implemented. Although there are many technical challenges faced by HMCM, we believe that it is an emerging technology holding huge potentials for CAV and general data analytics services. It is driving by many enabling technologies (especially for A-Fogs), some of which are discussed below.

2.3.2.1 Powerful Computing Devices

According to Moore's law the processor speed or overall computing power for computers doubles every two years. Although the growth rate started to slow around 2013 to double every two and a half years, the computing power of mobile devices will become much stronger in the near future. In addition, the huge industry interests in autonomous driving, industry robots, drones and smart surveillance cameras are creating intensive demands on powerful mobile computing

devices. There are already many powerful embedded computing devices on the market. For example the Drive PX Pegasus system announced by Nvidia in October 2017 has a total of 320 trillion operations per second (TOPS) of computational power.

2.3.2.2 Advanced Communication and Networking Technologies

In addition to the computing power, there are significant advances on wireless communication technologies recently, such as WiFi, cellular networks and visible light communications. The wireless communication data rates experienced fast growth in the past decade. For example, the IEEE 802.11ac based wireless local area network (WLAN) can provide up to 3.5 Gbit/s data rate with multiple user multiple input multiple output (MIMO) technologies. The IEEE 802.11ay standard which is under development and is estimated to complete in 2019 is expected to provide 100 Gbits/s with use of 60 GHz spectrum resources. The IEEE 802.11 based WLAN technologies can provide high speed free wireless communications among fog nodes and between A-Fogs and nearby IoT data sources. In addition, the 3GPP standard organization for cellular network technologies has already defined device to device (D2D) communication mode for direct communication between mobile devices. It is developing enhanced D2D technologies for 5G, which can support multiple Gbits/s among the devices. Due to the localized communication and high frequency resource reuse the cost of D2D communications can be much lower compared to infrastructure based communication via cellular base stations. Combined with the well studied multiple hop ad-hoc wireless networking technologies, new WLAN and cellular D2D technologies can remove the obstacles of communication cost and bandwidth for large volume data transfer of data analytics jobs.

2.3.2.3 Resource Virtualization and Security

Virtualization is a process of creating virtual environment on a computer to run desired programs, without interfering with the other services provided by the computer to other users. It has the benefits of efficient utilization of resources, better accessibility and minimization of risk among others. A-Fogs can be viewed as a light weight clouds. Virtualization is the first and essential step for the deployment of fog nodes, as each A-Fog node may have its own primary computing tasks, which should not be affected by the A-Fog analytics jobs from outside, and the security of the fog nodes has to be protected.

With the advent of cloud computing many virtualization tools and platforms such as KVM and Xen have been developed. There are also many virtualization platforms such as VMware and VirtualBox for general computers. Some virtualization tools like Xen and VirtualBox provide the support for virtual machine migration between computers without loss of availability. That support is particularly important for A-Fogs as some fog nodes are mobile.

2.3.2.4 Distributed Data Analytics

Compared to the cloud and D-Fogs, the computing power of the individual A-Fog nodes are still very low, which limits the scale of data analytics jobs that an A-Fog node can process. The large scale data processing engines developed for big data analysis could be an important enabling technology for large scale data analytics, which include Hadoop MapReduce, Spark, GraphLab and GraphChi [21][63]. Among these data processing engines, Spark is of particular interest as it is an open source, fast and general distributed computing engine. Spark can be executed locally or over a computer cluster. A collection of toolkits are provided to support a wide range of data analytics applications related to SQL, streaming, machine learning and analytics.

It is noted that fog computing is still a new computing model with research focusing on concepts and architectures. There is very little research reported on the design and evaluation of practical fog computing protocols and algorithms, and data analytics services over fog computing. There are still many research challenges to be tackled for HMCM, e.g. how to integrate these key technologies together to make the framework work efficiently and reliably, and how to control the analytics service quality and utilize the resources efficiently. QoS aware algorithm design is one of the research challenges addressed in this chapter, which is presented in the next two sections.

2.3.3 Example Use Cases

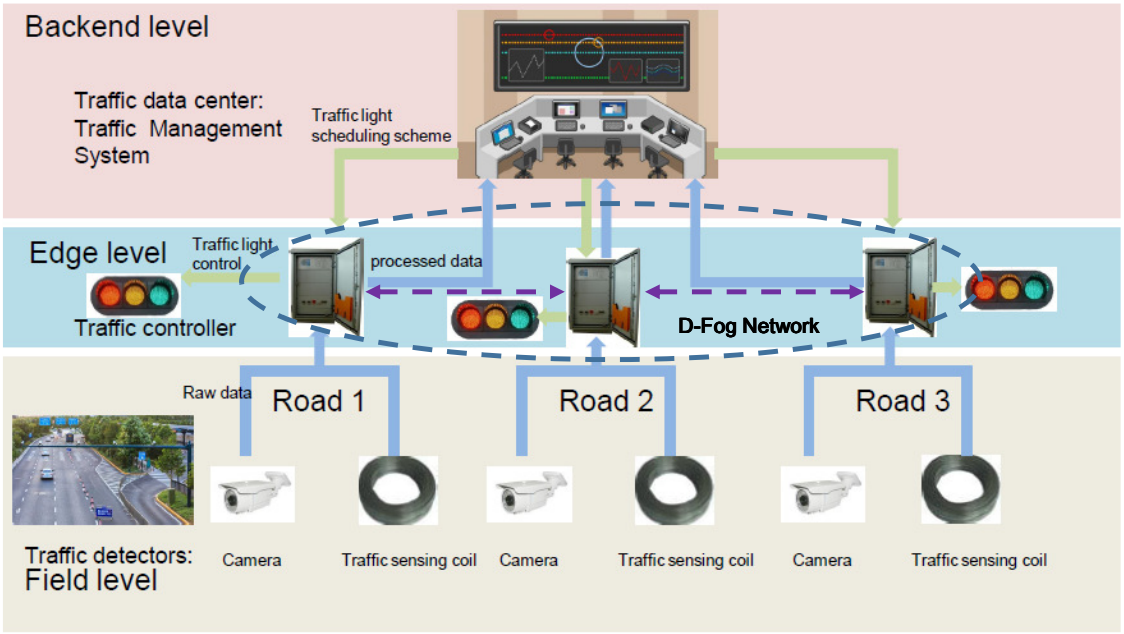


Figure 2.3: Traffic light scheduling application with interconnected fogs.

HMCM can be used for a wide range of data analytics services besides CAV applications. Next we discuss two example use cases.

2.3.3.1 Traffic Light Scheduling

Intelligent transportation systems (ITS) schedule traffic lights to keep vehicles moving and provide drivers with information to help them make smarter decisions on the road. For such applications with relatively fast response requirement, dedicated computing and communication resources are critical and should be used. One typical use case of HMCM for traffic light scheduling is shown as Fig. 2.3. There are monitoring and control systems for traffic detectors, traffic signal controllers and traffic management located in the data center, at the Backend level as shown in Fig. 2.3. The traffic detectors at the Field level shown in Fig. 2.3, such as the cameras and traffic sensing coils buried in the ground, monitor the traffic status of the roads, such as the queuing length of the vehicles on the road. The data collected from different detectors is sent to the traffic signal controller. In the existing practices, the data is sent to the traffic management systems in the data center, which runs advanced analytics algorithm to schedule traffic lights. For such an application, it is beneficial to put more intelligence at the field side. For example, the traffic signal controllers can be deployed with more computing resources and cooperate with close neighbors to carry out the data analytics tasks, such as data feature extraction before sending data to the data center. The purpose is to improve the data transmission reliability and reduce communication cost by reducing the amount of data transmitted from the traffic controllers to the data center, which often relies on the 3G/4G wireless links. The traffic controllers with dedicated computing and communication resources at the edge level can be viewed as D-Fogs in the HMCM. The traffic controllers with dedicated computing resources can also work with autonomous vehicles with opportunistic computing resources to support autonomous driving, e.g., on the tasks of hazard detection, accident avoidance, high definition maps, real time traffic reports and route planning.

2.3.3.2 Data Analytics for Multiple Smart Cities Applications

Besides the CAV applications, there are many general data analytics applications for smart cities. For such applications, historical and current data from different sources in the city are integrated and analyzed to provide trend analysis and predictions. It often needs to integrate multimodal data from diverse sources and jointly analyze the data based on some analytics algorithms for different analytics tasks to provide advanced services. A typical example is shown in Fig. 2.4. In the traditional approach, data from traffic sensors, car sharing data, bike sharing data and trip requests data from other IT systems are collected and sent to the cloud platforms. The clouds run advanced analytics algorithms to predict passenger movement pattern, which can be used by city planner to plan the city infrastructure. The response time is often not a key parameter here. However, the data volume can be so high that it is cost prohibitive to send and store all raw data in the cloud for analytics processing. In addition, it is hard to design centralized methods and infrastructure to acquire data from diverse sources using different communication protocols and data format. A more efficient approach is to add intelligence to the edge and perform data analytics as close as possible to the data sources, and send only the key features to the

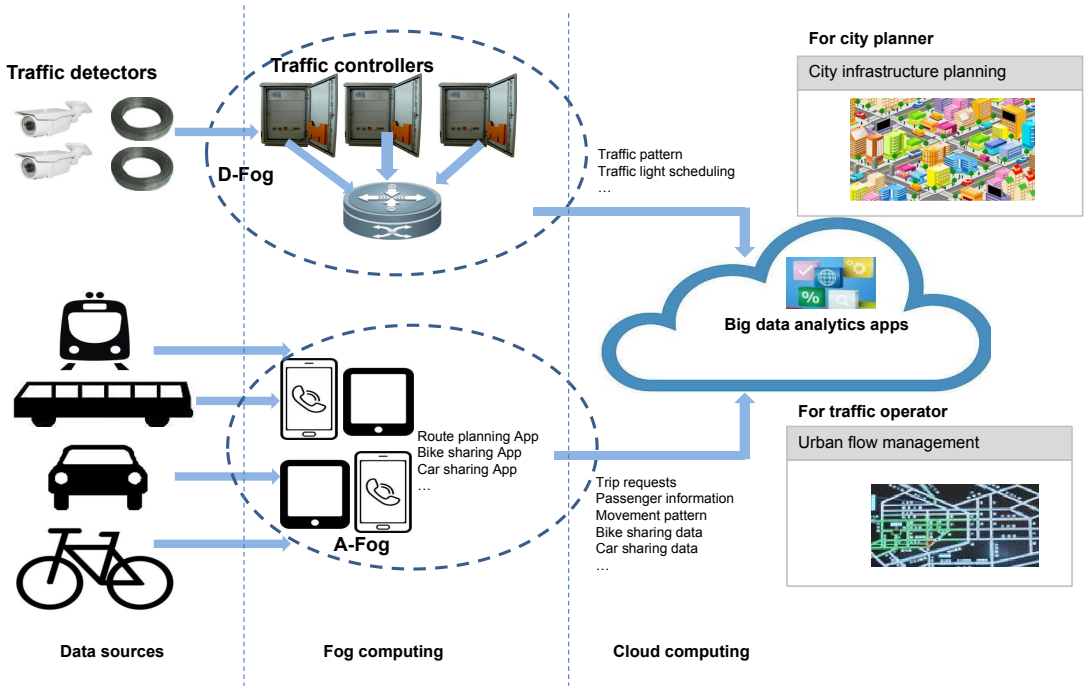


Figure 2.4: Data analytics applications for smart city.

clouds for more advanced analysis. For such applications, the Fog computing layer has both D-Fogs with dedicated computation and communication resources (such as the traffic controllers) and A-Fogs, e.g., formed by vehicles, mobile phones and tablets running different smart cities applications. The connection and cooperation among the fogs can help run multiple analytics tasks and achieve better service quality and resource utilization.

2.4 Benchmarking Experiments for Analytics Applications over Fogs

As the HMCM will undertake various analytics services from CAV applications with diverse QoS requirements, it is important to design and implement QoS aware service and resource management schemes for the analytics services. However, in order to do so, a key is to measure and model the workloads of different analytics services over the fogs with various computing and communication resources. In this section we present benchmarking experiments over A-Fogs and D-Fogs for such a purpose, to provide a basis for the QoS scheme design which is introduced in the next section.

2.4.1 Overall Experiment Methodology

For the benchmarking of analytics systems there could be three major dimensions of diversities that need to consider [21]: analytics computing platform, analytics algorithm and analytics job dataset. Analytics computing platform diversity refers to the wide availability of computing

processing platforms that can be used to support large scale analytics tasks, including generic data processing platforms and graph-specific platforms. Algorithm diversity refers to the wide availability of analytics algorithms that can be used for various analytics tasks, including machine Learning and graph based analytic algorithms. Dataset diversity refers to the varieties of data from applications that advanced analytics need to support.

In [21] authors have performed an intensive benchmarking of analytics systems over both private and public clouds, with various computing platforms and analytics algorithms:

- computing platforms: Spark [53], GraphLab [55], XStream [47] and GraphChi [57].
- analytics algorithms: logistic regression (LR), SVM for classification; K-means++ for clustering; alternating least squares (ALS) for collaborative filtering; and PageRank, weakly connected component (WCC), triangle counting (TC), breadth first search (BFS) for graph applications.

It was found from [21] that Spark and GraphLab perform the best over the compared computing platforms. As GraphLab has not been updated for a long time, we decide to use Spark as the only computing platform for benchmarking. The overall benchmarking experiment framework is shown in Fig. 2.5. In this thesis we present only experiments with LR and SVM algorithms for demonstration purpose. It is trivial to include more analytics algorithms and computing platforms to the benchmarking experiments.

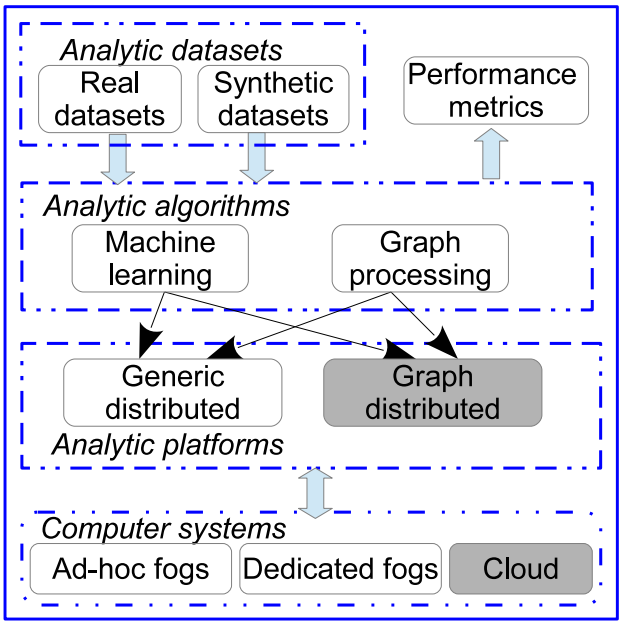
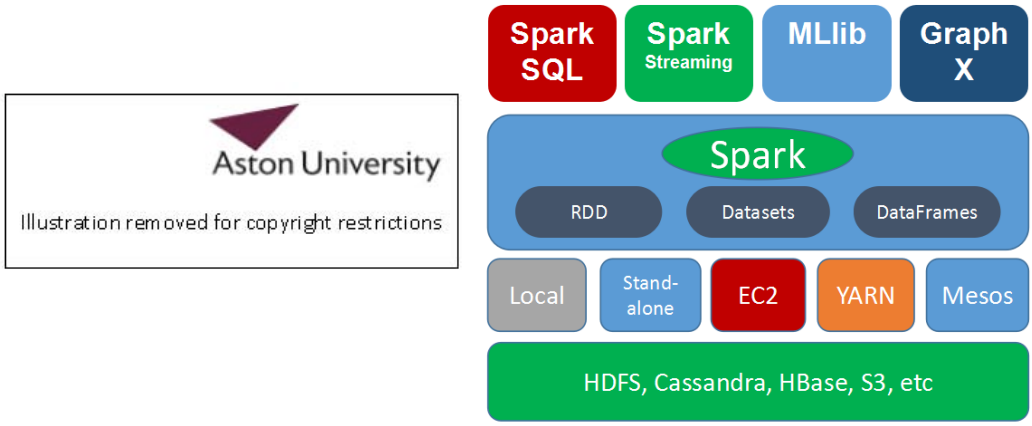


Figure 2.5: Overall benchmarking framework.

2.4.2 Computing System Setup for A-Fog Benchmarking

In the benchmarking with A-Fogs, we consider a pool of computing resources with one desktop PC and 8 Raspberry Pi 3 credit card sized micro computers, which is believed to form a reasonable ad-hoc fog environment. The Raspberry Pis are connected to a WiFi ad-hoc network

through their built-in wireless 802.11 module. One of the computers acting as fog master, while the rest act as fog workers. Virtual machines are installed on the computers, each allocated 700 MB RAM. Spark with the latest version 2.0 is installed in the virtual machines. Analytics job requests are sent from one of the fog nodes to the master node, which dispatches the jobs to the fog member nodes. Job completion time and resource consumption of the analytics jobs over Spark are recorded and used in the QoS aware resource management.



(a) Raspberry Pi. (b) Spark architecture.

Figure 2.6: Raspberry Pi and Spark architecture.

Raspberry Pi (shown in Fig. 2.6(a)) is a single-board computer with a 1.2 GHz 64-bit quad-core ARMv8 CPU and 1 GB RAM [58]. A 32 GB micro SD card with operating system Raspbian installed is slotted on each machine. They are connected to a WiFi ad-hoc network through the built-in wireless 802.11 module. Raspberry Pi has the features of low cost, low power consumption, small size but still good computing power. It has been used for many cost-effective entertainment, surveillance, mobile and IoT applications. The price and power consumption comparison between Raspberry Pi and general x86 server is presented in Table 2.1. In addition computing power and storage of Raspberry Pi have the similar features of A-Fog nodes such as smart phones, tablets, but has a better user-friendly programming environment.

Table 2.1: Comparison between x86 server and Raspberry Pi.

	Unit price	Power consumption
x86 server	$2000	180W/h
Raspberry Pi	$35	3.5W/h

Spark is an open source fast and general distributed computing engine for large scale data analytics. It is very popular for big data analytics with significant performance enhancement over Hadoop [53]. Spark has been evaluated and mainly used in large centrally controlled computer clusters. To the best of our knowledge, it has not been tested and evaluated in highly resource (computing and communications) constrained distributed computing environments. Fig. 2.6(b) shows the computing framework of Spark. The bottom layer is the data storage systems. Spark can access distributed datasets from different storage systems like HDFS [59], Cassandra [60],

Hbase [61] and Amazon S3 [62]. Above the storage system is the cluster management and execution layer. Spark can be executed locally or using several existing cluster managers over a computer cluster. The execution modes for Spark include standalone mode, Amazon EC2, Apache Mesos and Hadoop YARN [63]. Above the cluster management layer is the Spark compute engine. Spark adopts the resilient distributed dataset (RDD) as its architectural foundation, which has high fault tolerance. The fault resilient distributed computing engine is particularly needed for A-Fogs considering the mobility of A-Fog nodes, dynamic network topology and resource availability in A-Fogs. Datasets and DataFrames are extended object abstractions added in the latest version of Spark, which provide additional optimization for SQL operations. The top layer is a collection of toolkits including Spark SQL, Spark Streaming, MLlib and GraphX, which provide libraries and programming support to a wide range of analytics applications related to SQL, streaming, machine learning and graph processing.

Synthetic jobs are created with analytics algorithms SVM and LR over different dataset sizes. The configurations for the jobs are presented in Table 2.2 for A-Fogs. Experiment datasets for analytics jobs are generated by the Spark datasets generator. The synthetic datasets are used for performance evaluation in our study mainly for easy control of the dataset size, which can be adjusted by changing the number of vertices. Datasets generated from real CAV applications can be used as well. In this set of jobs the largest data size is 1770 MB, which is believed to be large enough for A-Fogs mainly consisted of devices with limited computing resource and energy. Jobs with larger datasets should be offloaded to and processed by dedicated fogs or remote clouds.

Table 2.2: Job types for A-Fog experiments.

A-Fog job	J1	J2	J3	J4	J5
# of Vertices (10^6)	1	2.5	5	10	30
LR size (MB)	58	145.8	291.6	583.3	1770
SVM size (MB)	61.3	153.3	306.6	590	1770

2.4.3 Benchmark Results over A-Fogs

Table 2.3: Computer settings used in the A-Fog experiments.

Label	A-1	A-2	A-3	A-4	A-5	A-6	A-7
Desktop	0	0	0	0	0	0	0
Raspberry	1	2	3	4	5	6	7

Label	A-8	A-9	A-10	A-11	A-12	A-13	A-14
Desktop	1	1	1	1	1	1	1
Raspberry	1	2	3	4	5	6	7

In this subsection we present benchmarking results with Spark over 14 A-Fog computer settings as shown in Table. 2.3. It is noted that the letter 'A' in the computer setting labels designates to A-Fogs.

As each A-Fog node may have only very limited computing resources, they should work together cooperatively and efficiently to process large datasets. It is unclear if Spark can run over distributed compute environment with small size and resource constrained micro computers, and how scalable it is to support large scale data analytics services over A-Fogs. Here scalability is referred to the ability of a computing platform or fogs to improve its computing performance with increasing computing resources [64].

Ideally a scalable platform is supposed to improve its performance linearly with additional computing resources. As single fog node with computers like Raspberry Pi has limited computing power, a scalable computing platform is important to extend the AaaS capabilities over fogs. We can study the AaaS feasibility and system scalability from two different aspects: the size of datasets that can be processed and the service completion (or running) time with increasing number of fog nodes.

Fig. 2.7 presents the job completion time for LR and SVM jobs with different datasets over the 14 A-Fog computer settings. It can be observed that the job completion time of both LR and SVM applications reduces quickly with increasing computing resources. Without surprise the large datasets requires much more service time, and the computer setting A-1 fails to complete the jobs J4. The results show clearly the feasibility of running AaaS service over A-Fogs but also the necessity of distributed computing to process large analytics jobs. In addition the desktop computer shows a large impact on the analytics service capacity, which substantially reduces job completion time and extends the size of datasets that can be processed.

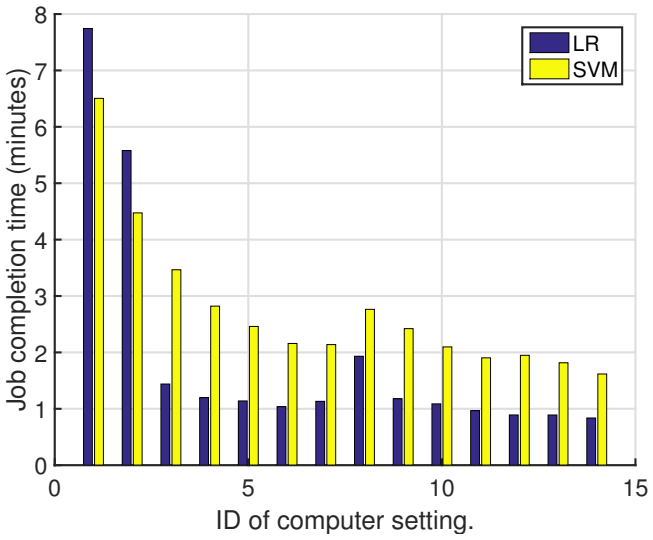
2.4.4 Benchmark Performance over Dedicated Fogs

For the benchmarking experiments over dedicated fogs, it is assumed that more powerful computing resources (including higher standard processor, memory and storage) are available. DELL R620 servers are used to set up computer clusters to represent dedicated fogs. Each server consists of an Intel Xeon E5-2680v2 2.8GHz CPU (dual core), a total memory of 256 GB and a standard 240G SSD drive. An extra server with 16 TB to store and access the input and output files consistently for all configurations.

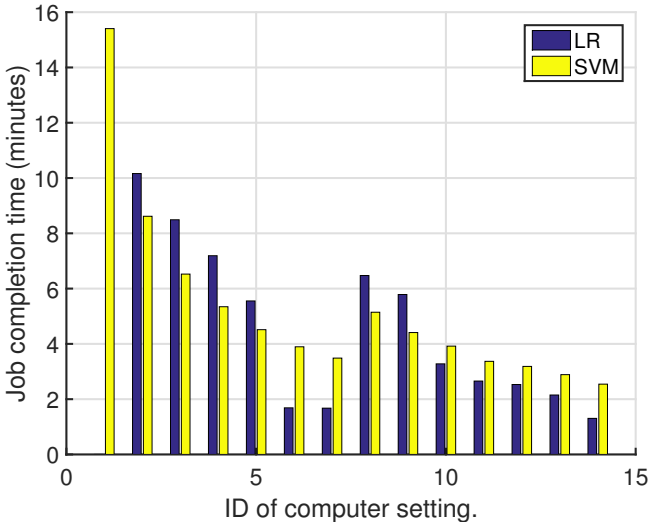
In order to test the computing performance of the D-Fog, we create a pool of 16 virtual machines (VM). The operation system installed on each VM is CentOS release 6.5 with the kernel version 2.6.32. To evaluate and compare the analytics computing performance under different experiment configurations, 12 computer settings in terms of number of CPU cores and memory size are used, which are shown in Table. 2.4. It is noted that the letter 'D' in the computer setting labels designates to D-Fogs.

Similar to the experiments over A-Fogs, the configurations for the jobs are presented in Table 2.5 for D-Fogs. In total 8 types of jobs with different data sizes ranging from 1.1 GB to 35.4 GB are used for the experiments with D-Fogs.

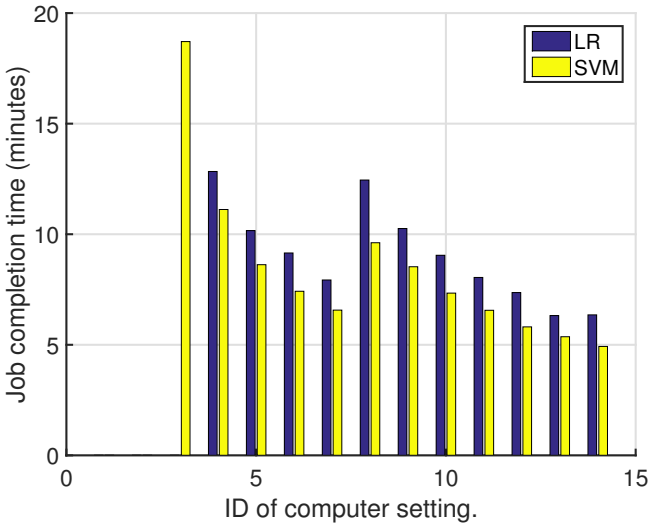
Benchmarking results for LR and SVM algorithms are presented in Fig. 2.8. Similar performance trends with D-Fogs benchmarking results are obtained as these for A-Fogs. Firstly the



(a) A-Fog job J2.

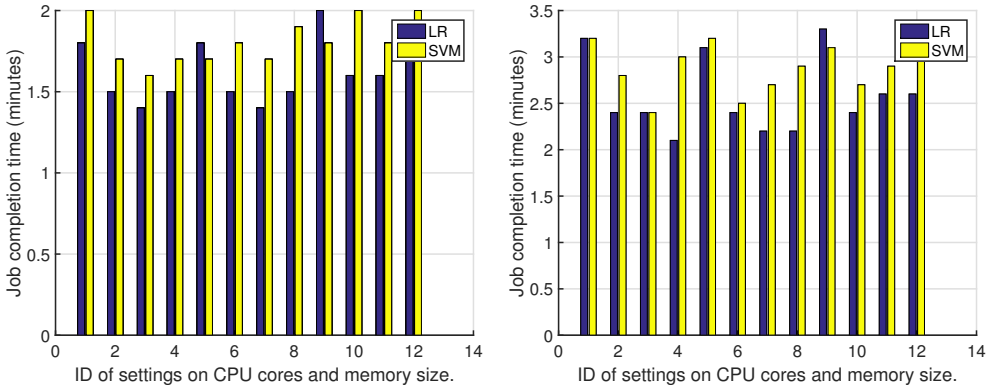


(b) A-Fog job J3.



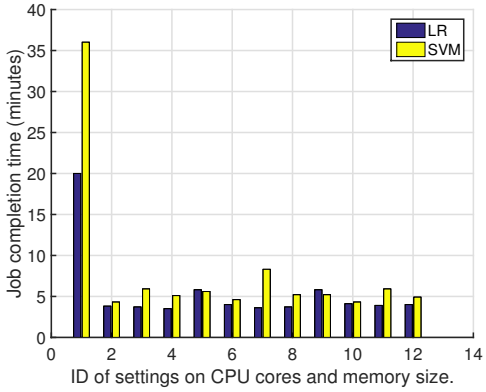
(c) A-Fog job J4.

Figure 2.7: Job completion time of LR and SVM algorithms versus A-Fog computer settings over different jobs

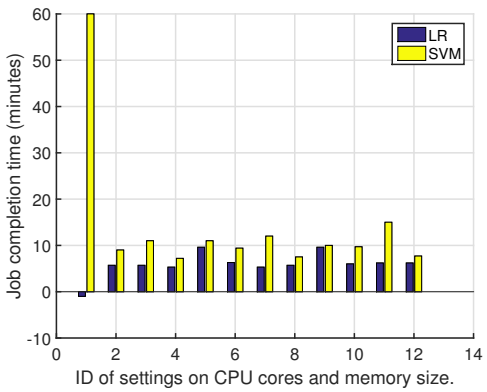


(a) D-Fog job J1.

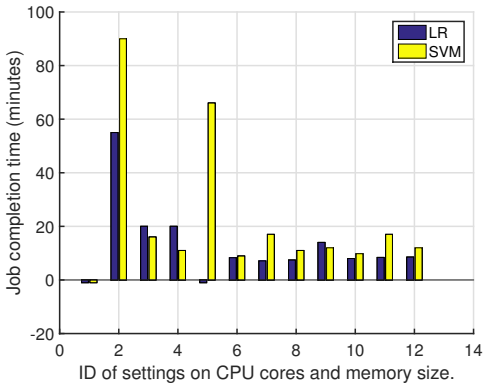
(b) D-Fog job J2.



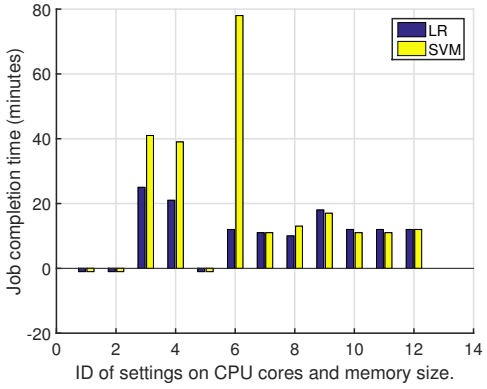
(c) D-Fog job J3.



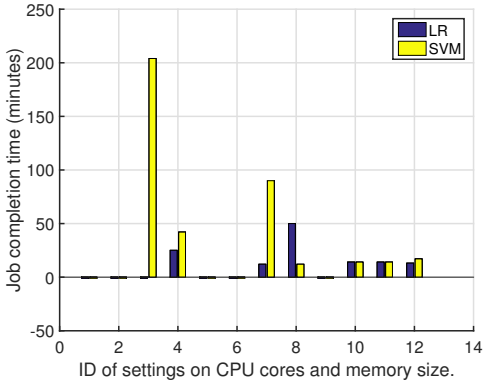
(d) D-Fog job J4.



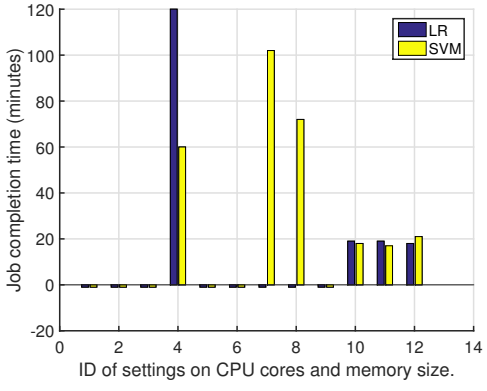
(e) D-Fog job J5.



(f) D-Fog job J6.



(g) D-Fog job J7.



(h) D-Fog job J8.

Figure 2.8: Job completion time of LR and SVM algorithms versus D-Fog computer settings over different jobs.

Table 2.4: Computer settings used in the D-Fog experiments.

Label	D-1	D-2	D-3	D-4	D-5	D-6
CPU cores	4	4	4	8	8	8
Memoery (GB)	4	8	16	4	8	16

Label	D-7	D-8	D-9	D-10	D-11	D-12
CPU cores	12	12	12	16	16	16
Memoery (GB)	4	8	16	4	8	16

Table 2.5: Job types for D-Fog experiments with different number of vertices (10^6) and dataset size (GB).

D-Fog job	J1	J2	J3	J4	J5	J6	J7	J8
Vertices	20	50	100	175	250	350	450	600
LR size	1.1	2.8	5.7	10	14.2	19.9	25.6	34.2
SVM size	1.1	3	5.9	10.3	14.7	20.6	26.5	35.4

job completion time decreases with computing resources (CPU and memory) scaling up. The size of dataset that can be processed by the D-Fogs also increases largely due to more computing power and memory. However for the J7 and J8 with the largest datasets, only the highest computer settings can complete the analytics jobs successfully. The completion time for those settings failing to complete the jobs is set to -1 in the figures.

2.5 QoS Aware Service and Resource Management

2.5.1 Overall Design

In this section we are focused on the core problem of analytics service quality and resource management for HMCM. Design of QoS aware data analytics service and resource management is presented. It is noted there are still many other challenges faced by the proposed framework based data analytics, such as fog formation, pricing and security for resource sharing, service discovery and mobility management, which are left as our future works.

In HMCM we assume there are a set $\mathcal{A}$ of N_{a} A-fogs and a set $\mathcal{D}$ of N_{d} D-fogs located in an investigated area. For simplicity we let $\mathcal{S}$ denote the set of all computing centers, including the N_{a} A-fogs, N_{d} D-Fogs and a cloud. The size of the set $\mathcal{S}$ is $N_{\mathrm{a}} + N_{\mathrm{d}} + 1$. The CAV sensors may have connection to multiple fogs and cloud. Data analytics jobs can be executed in any of these computing centers with sufficient computing resources and communication bandwidth.

The overall functionalities of service and resource management module is shown in Fig. 2.9. The CAV sensors generate data and analytics jobs with associated QoS targets (e.g., target job completion time). Job owners send requests to the computing centers, which may be accepted or rejected. Admission control and offloading scheme is applied to decide if a job request can be accepted or rejected, and which computing center the job should be executed if it is accepted. The decisions are made based on many factors, including cost models for computing and communication resources, workload models, available computing resources and network bandwidth. If a job request is accepted, computing resources at the A-Fogs, D-Fogs or cloud are

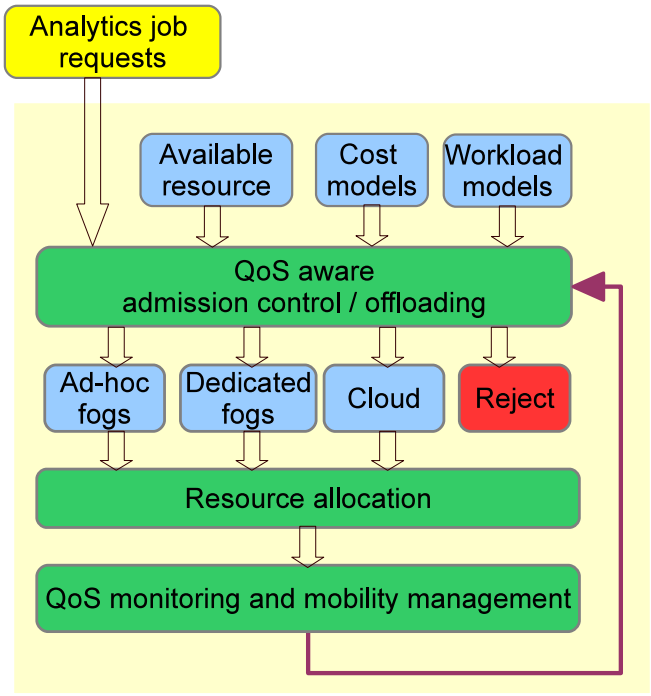


Figure 2.9: Service and resource management framework.

allocated. The computing and QoS performance is then recorded. Mobility of the job owners and fog workers is monitored, and necessary actions such as renewed offloading and resource allocation are taken due to loss of fog workers.

Next we present the assumptions and/or models for the main components of HMCM in the remaining of this subsection, including computing resources, analytics jobs, workload model, communications and power consumption. These models form a general framework which is vital for a practical and comprehensive evaluation. Then centralized and distributed algorithms for admission control and resource allocation (ACRA) are presented in Section 2.5.2 and 2.5.3, respectively.

2.5.1.1 Computing Resource Assumption

For simplicity, we assume that each A-Fog has a fog master and a cluster of computers as workers, which form a computing resource pool with computers and other smart personal devices like Raspberry Pi. The fogs are uniformly distributed in the studied area. The computers have identical computing specifications in terms of CPU and memory size. We let $N_i^{\mathrm{a,c}}$ and $N_i^{\mathrm{a,m}}$ denote the number of computers and mobile devices in the A-Fog i. The D-Fog i has a fog master and a computing resource pool of $N_i^{\mathrm{d,c}}$ CPU cores and $N_i^{\mathrm{d,m}}$ GB memory. The remote cloud is assumed to have unlimited computing resources. Let $P_i^{\mathrm{a,c}}$ and $P_i^{\mathrm{a,m}}$ denote the computing price per seconds for using a computer and a mobile device in the A-Fog i, respectively. Let $P^{\mathrm{d,c}}$ and $P^{\mathrm{d,m}}$ denote the computing price per seconds for using a CPU core and 1 GB memory at D-Fogs or clouds, respectively.

2.5.1.2 Analytics Service Model

Analytics jobs are generated by groups of city sensors and mobile devices. An analytics job is jointly characterized by the analytics algorithms (such as SVM and LR) to be applied to the job and its dataset size. Let (g, s) denote a job type with g representing the analytics algorithm, $g \in [1, N_{\mathrm{G}}]$, and s representing the ID of dataset interval, $s \in [1, N_{\mathrm{S}}]$. N_{G} represents the total number of investigated analytics algorithms, and N_{S} denotes the number of discrete dataset sizes. Let $J_{g,s,j}$ denote the jth job of type (g, s). Each analytics job belongs to a job owner. Job owners make requests of analytics services to fogs or cloud, set service payment and monitor the received QoS for their analytics jobs.

Without loss of generality, we assume that jobs are generated following Poisson distributions. The job generation rate per second for job type (g, s) is denoted by $\lambda_{g,s}$. The actual dataset size for a job $J_{g,s,j}$ is assumed to be uniformly distributed in the considered range of dataset size. The geographic positions of the mobile devices that generate jobs are assumed to be uniformly distributed in the considered network area.

Target job completion time is the main QoS metric of interest for analytics jobs. Each job is associated with a target job completion time and a service completion charge. Job completion time is measured from the time a job request is accepted to the completion of the job, including the network latency, data communication time, queue delay and job computation time. Data communication time is determined by network bandwidth and dataset size, while job computation time is determined by the analytics algorithm, dataset size and allocated computing resource for the job.

Let $T^{\mathrm{e}}_{g,s,j}$ denote target completion time in seconds, and $R_{g,s,j}$ denote the service completion charge in dollars, for job $J_{g,s,j}$, respectively. The service completion charge of a job is set proportional to the dataset size of the job. And a simple business model is considered here: if a job request is accepted and completed within the target completion time, the job owner pay to the fog operator running the job; otherwise no payment is made.

2.5.1.3 Workload model

Workload model is an important component for HMCM. which maps a given analytics job and a computing resource allocation to a job completion time. With the expected job completion time, the computing centers and the job owners can have a rough estimation of the required computing resource, the best resource allocation and the cost for completing the job. Therefore ACRA decisions can be made for job requests.

Obviously workload model is highly dependent on the job type and computing resource specification. To create the workload model, extensive benchmarking experiments over A-Fogs, D-Fogs and cloud computing environments have been designed and run, which provide the basis for the design and evaluation of the QoS aware algorithms. Detailed experiment design and results are presented in Section 2.4. By running the sample analytics jobs over the different computing options available at the fogs and cloud, we can obtain job completion time with

given computing options. A workload model is then created for computing centers as a table mapping jobs and computing resources to job completion time. In the table each entry stores a job completion time for one dataset size and one computing resource setting.

It is noted that the real analytics jobs may have different dataset sizes from those evaluated in the benchmarking experiments. However it is impossible and not necessary to run benchmarking experiments with every possible dataset size. Through the sample jobs with properly configured dataset sizes we can estimate the completion time of real jobs. Estimation accuracy can be improved with more sample jobs and additional real job computing results.

2.5.1.4 Communication and Networking Assumption

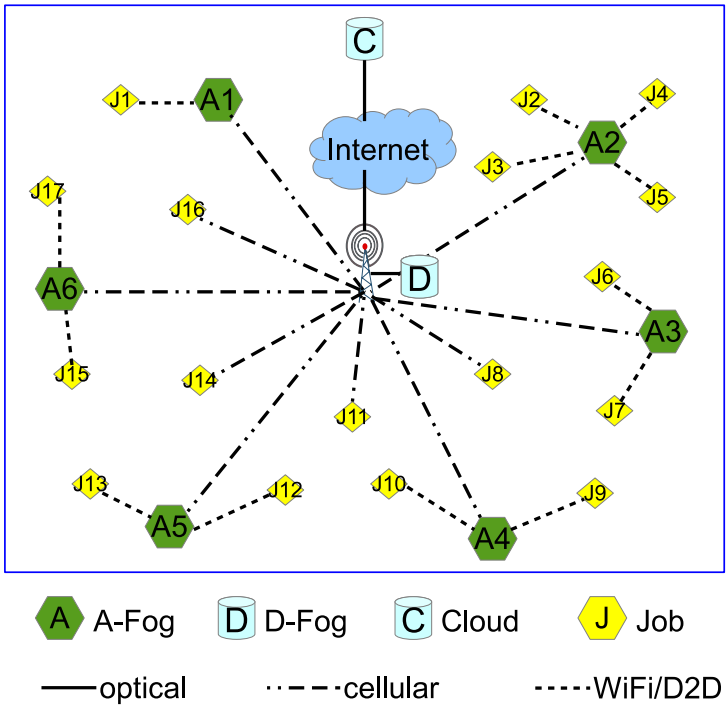


Figure 2.10: Illustration of communications among Fogs and jobs.

Heterogeneous communication and networking technologies are assumed in HMCM. For the communications among computing centers, we have the following assumptions: clouds are connected to the Internet via high speed optical links; D-Fogs are connected to the Internet via wired links; A-Fogs are connected to the Internet via cellular networks.

For the communications between the analytics jobs and the computing centers, we assume that the mobile devices are always within service coverage of cellular networks and connected to the Internet. They are connected to the D-Fogs and clouds via only cellular base stations. In addition, some of the devices may have alternative connections to the A-Fogs via cheap wireless communication technologies WiFi or cellular D2D. Fig. 2.10 shows an example of HMCM with 6 A-Fogs, 1 D-Fog and 17 jobs. All the jobs are connected to the Internet via cellular base stations, which are not shown in the figure. Some jobs are close to the A-Fogs, therefore it is possible to transfer data to the A-Fogs using WiFi or D2D technologies.

We let β denote the probability of job $J_{g,s,j}$ connected to an A-Fog via WiFi or D2D technologies. For simplicity we assume the value of β is dependent on the Euclidean distance of the job source $J_{g,s,j}$ to the A-Fog i, which is denoted by $d_{g,s,j,i}$ (meters), for $i \in \mathcal{A}$. As WiFi and D2D have around 300 meters communication range we use a heuristic method to compute β:

$$\beta = \exp\left(-\frac{d_{g,s,j,i}}{300}\right). \tag{2.1}$$

The main consideration with the method is that a decrease in the distance between a job and an A-Fog will increase the availability of WiFi or D2D wireless connection.

Let P_{wifi} and P_{cell} denote the communication service prices ($/GB), for using WiFi and cellular base stations, respectively. Let $P^{\text{n}}_{g,s,j,i}$ denote communication price for job $J_{g,s,j}$ to be processed at the computer center i, which is computed according to the used connection mode and the price of communications with the chosen connection mode. Let $C^{\text{n}}_{g,s,j,i}$ denote the communication cost from job $J_{g,s,j}$ to the computer center i. The cost is a linear function of the job dataset size and the communication price.

Let B^{d} and B^{c} denote the aggregate bandwidth of the bottleneck link on the path from job sensors to the D-Fogs and clouds, respectively. Let $B^{\text{a,c}}_i$ and $B^{\text{a,w}}_i$ denote the aggregate bandwidth of the bottleneck link for A-Fog i ($i \in \mathcal{A}$) via cellular base stations and via WiFi, respectively. If a job is assigned to a computing center, for example an A-Fog, we assume the job will share the bottleneck link bandwidth equally with other jobs assigned to this A-Fog. Let $B_{g,s,j,i}$ denote the data rate requirement for a job $J_{g,s,j}$ when it is assigned to computing center i. For a real-time streaming analytics job, the data rate $B_{g,s,j,i}$ equals to the source data rate of the streaming job. Otherwise the data rate $B_{g,s,j,i}$ is computed as an equal share of the bottleneck bandwidth of computer center i with other jobs.

2.5.1.5 Power Consumption Model

For the power consumption, the same model is adopted for the mobile devices and computers used in A-Fogs, and servers used in D-Fogs and clouds. Let f_u denote machine CPU frequency. The power of a mobile device or computer, denoted by P_w is approximated by the following formula [40]:

$$P_w = p_1 f_u^{p_e} + p_2, \tag{2.2}$$

where p_1 and p_2 are configurable positive constants, set to 3.206 and 68, respectively; and p_e is set to 2 for A-Fogs, and 3 for D-Fogs and clouds [40]. The electricity price p_e is set to 0.15 $/KWh. Let $C^{\text{p}}_{g,s,j,i}(k)$ denote the energy cost for a job $J_{g,s,j}$ completed at the i computing environment, $i \in \mathcal{S}$ with computing resource specification option k. According to the computing time of a job at a specific computing center, the power consumption can be computed and the energy cost for a job can be computed accordingly.

2.5.2 Centralized Admission Control and Resource Allocation

For any new analytics job request, the HMCM needs to make decisions on job admission control and resource allocation (ACRA), including acceptance of a job, deciding which computing center (A-Fog, D-Fog or cloud) to run the job, and the computing resource configuration from selected computing center. With the above system assumptions, a straight forward solution is a centralized ACRA algorithm, with objectives of maximizing computing system revenue while ensuring service quality of the analytics jobs, under given price and constraints of computing and communication resources and target job completion time. With the centralized algorithm, the above three decisions can be jointly made with globally system state and resource information.

Let $C^{\mathrm{c}}_{g,s,j,i}(k)$ denote the computing resource cost for the job $J_{g,s,j}$ under the kth computing resource configuration in computing center i. The computing cost depends on computing resource price and the time used by the analytics job. Let $T_{g,s,j,i}(k)$ denote the computation time (in seconds) under the case that the job $J_{g,s,j}$ is processed with computing option k in the computer center i. Then the computing cost for a given job can be computed according to the computing resource allocated for this job, the computation time with the allocated resources, and the computing resource price.

Let us define the total cost for job $J_{g,s,j}$ being processed with computing option k in the computer center i, which is denoted by $C^{\mathrm{t}}_{g,s,j,i}(k)$, as the sum of the communication, computing and power costs for the job. $C^{\mathrm{t}}_{g,s,j,i}(k)$ can be computed by:

$$C^{\mathrm{t}}_{g,s,j,i}(k) = C^{\mathrm{n}}_{g,s,j,i} + C^{\mathrm{c}}_{g,s,j,i}(k) + C^{\mathrm{p}}_{g,s,j,i}(k). \tag{2.3}$$

Let K denote the total number of computing options. The kth computing option at A-Fog i is configured with $N^{\mathrm{a,c}}_{i,k}$ computers and $N^{\mathrm{a,p}}_{i,k}$ portable devices; and the kth computing option at D-Fog i is configured with $N^{\mathrm{d,c}}_{i,k}$ CPU cores and $N^{\mathrm{d,m}}_{i,k}$ GB memory; The computing resources configurations used for A-Fogs and D-Fogs are shown in Table 2.3 and Table 2.4, respectively. Let $U_{g,s,j,i}(k)$ denote the service utility of completing a job $J_{g,s,j}$ with the kth computing option in the computer center i, for $k \in [1, K]$, and $i \in \mathcal{S}$. $U_{g,s,j,i}(k)$ is computed by:

$$U_{g,s,j,i}(k) = \begin{cases} R_{g,s,j} - C^{\mathrm{t}}_{g,s,j,i}(k), & T_{g,s,j,i}(k) \le T^{\mathrm{e}}_{g,s,j} \\ -C^{\mathrm{t}}_{g,s,j,i}(k), & T_{g,s,j,i}(k) > T^{\mathrm{e}}_{g,s,j} \end{cases}, \tag{2.4}$$

Let $\sigma_{g,s,j,i,k}$ be a binary variable with value 1 indicating a job $J_{g,s,j}$ is assigned to the computing center i with computing option k and with value 0 otherwise. Let $\chi_{g,s,j,i}$ be a binary variable with value 1 indicating a job $J_{g,s,j,i}$ has WiFi connection to computing center i and with value 0 otherwise. With the above definitions, the optimization problem for the Centralized ACRA

algorithm can be formulated below:

$$
\begin{aligned}
\max_{\sigma_{g,s,j,i,k}} \quad & \sum_{g,s,j}\sum_{i,k}\sigma_{g,s,j,i,k}U_{g,s,j,i}(k) \\
\text{s.t.} \quad & \sigma_{g,s,j,i,k} \in \{0,1\} \\
& \sum_{g,s,j}\sum_{i,k}\sigma_{g,s,j,i,k} \leq 1 \\
& \sigma_{g,s,j,i,k}U_{g,s,j,i}(k) > 0 \\
& \sum_{g,s,j}\sum_{k}\sigma_{g,s,j,i,k}N^{\text{a,c}}_{i,k} \leq N^{\text{a,c}}_{i}, for\ i \in \mathcal{A} \\
& \sum_{g,s,j}\sum_{k}\sigma_{g,s,j,i,k}N^{\text{a,c}}_{i,k} \leq N^{\text{a,m}}_{i}, for\ i \in \mathcal{A} \\
& \sum_{g,s,j}\sum_{k}\sigma_{g,s,j,i,k}N^{\text{d,c}}_{i,k} \leq N^{\text{d,c}}_{i}, for\ i \in \mathcal{D} \\
& \sum_{g,s,j}\sum_{k}\sigma_{g,s,j,i,k}N^{\text{d,m}}_{i,k} \leq N^{\text{d,m}}_{i}, for\ i \in \mathcal{D} \\
& \sum_{g,s,j}\sum_{k}\sigma_{g,s,j,i,k}B_{g,s,j,i,k}(1-\chi_{g,s,j,i}) \leq B^{\text{a,c}}_{i}, for\ i \in \mathcal{A} \\
& \sum_{g,s,j}\sum_{k}\sigma_{g,s,j,i,k}B_{g,s,j,i,k}\chi_{g,s,j,i} \leq B^{\text{a,w}}_{i}, for\ i \in \mathcal{A} \\
& \sum_{g,s,j}\sum_{k}\sigma_{g,s,j,i,k}B_{g,s,j,i,k} \leq B^{\text{d}}, for\ i \in \mathcal{D} \\
& \sum_{g,s,j}\sum_{k}\sigma_{g,s,j,i,k}B_{g,s,j,i,k} \leq B^{\text{c}}, for\ i \in \mathcal{C}
\end{aligned}
\tag{2.5}
$$

The optimization objective is to maximize system utility by finding the best decisions for job admission, resource assignment and computing options for the analytics jobs. The first and second constraints ensure that a job can not be assigned to more than one computing center. The third constraint prevents jobs with negative service utility from being accepted. The forth to seventh constraints ensure the computing resources are not overused. The eighth to eleventh constraints control the aggregate data rate of jobs assigned to a computing center does not exceed its bottleneck bandwidth.

As all the unknown variables in the centralized ACRA algorithm optimization problem are binary and only restrictions must be satisfied, the problem is clearly NP-complete, which is well known can't be solved in polynomial time. The computation complexity is $\mathcal{O}(2^{N_{\text{a}}+N_{\text{d}}+N_{\text{J}}})$, where N_{J} is the total number of computing jobs. Next we proposed heuristic approximate algorithms to solve the ACRA problem.

2.5.3 Distributed ACRA algorithms

As the centralized ACRA algorithm has very high computational complexity, we propose distributed ACRA algorithms in this subsection. For the distributed ACRA algorithms, we assume that all the job owners have the information of the remote clouds, such as the communication and computing resources and their prices. In addition, the job owners may have the information of close A-Fogs and D-Fogs. Two types of distributed ACRA algorithms, cooperative and

non-cooperative algorithms, are presented as follows.

2.5.3.1 Non-cooperative Distributed ACRA Algorithms

In the non-cooperative distributed ACRA algorithms, there is no cooperation among the computing centers, which make ACRA decisions according to the local information and do not interact with other computing centers. Two baseline non-cooperative ACRA algorithms are considered, random ACRA algorithm and preference based ACRA algorithm.

In the random ACRA algorithm, each job owner sends a request to a randomly selected computing center. The request carries the information of job size, job type, target job completion time and job completion reward. The computing centers receiving a request make an admission decision on the job according to the available computing and communication resources. If the job is accepted, it is put to job queue and is completed before the target job completion time. Analytics results are sent back to the job owner and the computer center is paid as agreed. If the job is rejected, the job owner sends a request to another randomly selected computing center. The process repeats until the job is accepted or rejected by all the computing centers, in which case the job is blocked.

In the user preference based algorithm, each job owner creates a preference list for the computing centers known to it, according to the cost of computing and communication resources, and computing capacity. While there are many ways to compute a preference for a given computing center, a simple approach is used here. Suppose that a job owner has the information of a list of computing centers, which is denoted by $\mathcal{L}$. For a given job, the job owner computes the distance to each computing center in the list, and the minimal computing cost among the available computing options in each computing centers. Then the computing centers are ranked in decreasing order of distance, computing cost, and computing resources usage, separately, which are denoted by O_i^{d}, O_i^{c}, O_i^{u}, respectively, for $i \in \mathcal{L}$. Let O_i^{p} denoted the preference value for the computer center i in the list $\mathcal{L}$, which can be computed according to the above ranks:

$$O_i^{\mathrm{p}} = w_{\mathrm{n}} O_i^{\mathrm{d}} + w_{\mathrm{c}} O_i^{\mathrm{c}} + w_{\mathrm{u}} O_i^{\mathrm{u}}, \tag{2.6}$$

where w_{n}, w_{c} and w_{u} are the weights for the rankings, which are set to 0.8, 0.1 and 0.1, respectively. It is noted that the distance is given a dominate weight, as it directly determines the availability of low cost wireless connection, and also introduces randomness to avoid too many jobs are submitted to one computing center. The preference list can be created by ranking the computing center preference values in decreasing order. The job owner sends request to the computing centers according to the preference list. If the request is accepted by a computing center, the ACRA process is completed. Otherwise the job owner finds the next computing center in the preference list. The process repeats until the request is accepted or rejected by all computing centers.

It is noted in the non-cooperative ACRA algorithms, the decisions on job requests are made immediately by the computing centers without waiting for other job requests or feedback from

other computing centers.

2.5.3.2 Cooperative Matching Theory based ACRA Algorithm

While the non-cooperative ACRA algorithms are simple, the computing and communication resources may not be effectively utilized. To improve the resource utilization and analytics service quality, a cooperative matching theory based ACRA algorithm is proposed in this subsection. Matching theory is a mathematical framework used to form mutually beneficial relationships over time, which has been widely used for lab economics and distributed wireless management.

In the matching theory based ACRA algorithm, the system is operated with matching phases. Each matching phase lasts no more than T_{m} seconds. Each computing center regularly publishes its computing resource price and available computing resources, which are available to job owners. At any time if a computing job is generated, the job owner creates a preference list of the computing centers known to it for this job with formula (2.6) according to the method used in the user preference based ACRA algorithm.

In the start of each matching phase, each computing center creates an empty waiting list of analytics jobs. And then the job owners send service request to computing centers, and the computing centers select jobs following the matching process presented in Algorithm 1. The matching process stops if all jobs not in any waiting list have empty preference list or the matching phase expires. Once the matching process stops, the jobs in the waiting lists of the computing centers are accepted and notifications are sent to the job owners. The accepted jobs are then executed immediately. Then a new matching phase follows.

Algorithm 1 Matching theory based cooperative ACRA algorithm.

1: Job owners with new jobs create preference list of known computing centers.
2: **for** each matching period **do**
3: Each computing center creates an empty waiting list of jobs.
4: **while** (jobs not in any waiting list has non-empty preference list and matching phase not expire) **do**
5: Requests for jobs not in any waiting list are sent to first computing center in preference list.
6: Computing centers compute the utility of the received and queued job requests with all computing options according to (2.4) and rank the jobs in decreasing job utility.
7: The top ranked job requests with positive utilities are added to waiting lists sequentially and available computing resources are updated, until no more computing resource is available.
8: Jobs not added to waiting lists are rejected and rejection notification sent to job owners.
9: Job owners receiving job rejection notification remove the corresponding computing center from their preference list, otherwise they wait for further notifications.
10: **end**
11: Computing centers accept jobs in their waiting lists and send acceptance notifications.
12: **end**

Fig. 2.11 shows an example matching process. In the example there are m jobs and n com-

puting centers. Owners of jobs J_1 and J_m send request first to computing centers C_n and C_1, respectively. C_1 adds J_m to waiting list, while C_n rejects job J_1 and sends a rejection notification to J_1 job owner. Then J_1 owner proposes to C_1, which in this case is added to waiting list by C_1. When there are no further jobs being rejected, all the jobs in the waiting lists are accepted and acceptance notifications are sent to the job owners. Then a new matching phase starts.

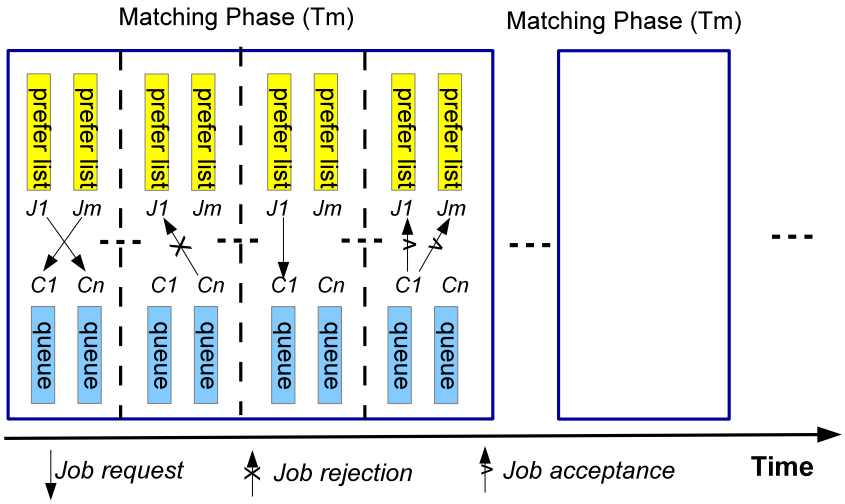


Figure 2.11: Illustration of jobs to computing centers matching process.

It is noted that the matching period time T_{m} is a key design parameter for the matching theory based algorithm. A smaller T_{m} finds less jobs to be matched to the computing centers in one phase, which may not be good for computing resource utilization. If T_{m} is set to be too small, the matching theory based algorithm behaves as the user preference based algorithm. On the other hand, a too large T_{m} may cause unwanted delay to the job completion time. A solution to avoid the service start delay due to matching process is to request service a certain time earlier before the job is generated.

2.6 Evaluation of ACRA Algorithms

In this section the feasibility of HMCM based data analytics is assessed and the ACRA algorithms are evaluated.

2.6.1 Simulator Design

A discrete-event driven system level simulator is developed, which can be used to perform simulations with a large scale of computing resources in the A-Fogs and D-Fogs. Based on the simulator extensive experiments are run to obtain simulation results, which are analyzed and discussed to get insights into the performance of the proposed ACRA algorithms and the feasibility of HMCM based data analytics services.

While there are several simulators available for simulations of cloud computing and cloudlet

computing, to the best of our knowledge, no fog computing simulator with analytics services and QoS support has been reported. The finite state machines of our fog computing simulator is presented in Fig. 2.12. The core of the simulator is an event scheduler, which schedules events according to the event arrivals: new jobs to be generated or jobs being completed. If a new job is generated, the job goes through the admission control procedure, where various ACRA algorithms are supported. After the ACRA process, jobs are either accepted or rejected. Then the simulator returns to the event scheduling state. If a job is completed, computing resources allocated to that job are released.

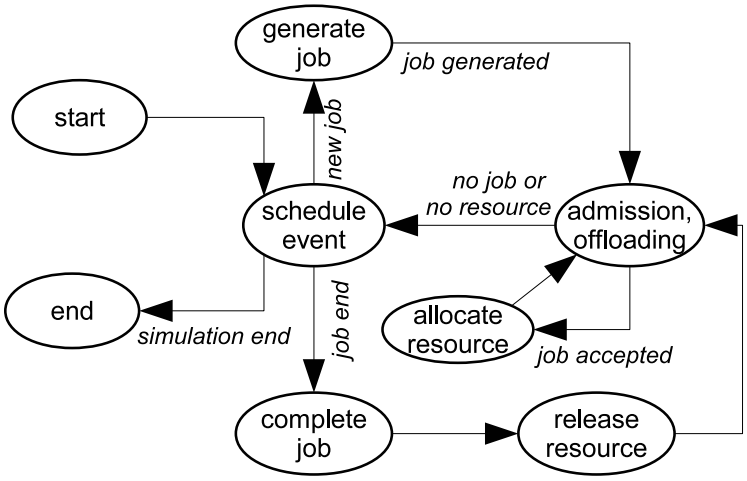


Figure 2.12: Simulator finite state machines.

2.6.2 Experiment Setup

While there are a vast space for system parameter setting, only typical ones are considered for our experiments, which are summarized in Table 2.6.

Analytics job arrival rates $\lambda_{g,s}$ are set to decrease with dataset type s, but remain the same for any given algorithm type g. Each job has a configurable target completion time, which is the main indicator of QoS requirement. In the simulations the target job completion time for a job is uniformly distributed in the range of 1 to 3 times the communication time to the cloud. The job service charge in dollars is assumed to be randomly distributed in the range of 1 to 1.5 times the communication cost to the cloud.

With the above system configuration, the communication costs for the transfer of job data to the computing centers, and the computing cost can be computed for the sample jobs running with all the computing settings in advance. The total costs versus the computing resource settings are sorted in increasing order and stored in a cost table for each job type. In simulations the possible options are simply selected for a given job as the available computer settings from the corresponding cost table.

For the large scale data analytics service and the QoS aware service and resource management scheme, the major concern from the analytics service users is on the service quality such as job blocking probability; while for the service operators of the fogs and the clouds they are more

Table 2.6: System parameter settings.

Variable	Values	Meaning
L (Km)	1	Length of the side of a squared network area
N_{a}	20 (default)	Number of A-Fogs
$N_i^{\mathrm{a,c}}$	[10,20]	Number of computers at A-Fog i
$N_i^{\mathrm{a,m}}$	[40,60]	Number of Raspberry Pi at A-Fog i
$N_i^{\mathrm{d,c}}$	500	Number of D-Fog cores
$N_i^{\mathrm{d,m}}$ (GB)	1000	Size of D-Fog memory
T_{m} (Sec)	60 (default)	Matching phase interval
$B_{\mathrm{a,c}}$ (Mbps)	20	Job to A-Fog cellular bandwidth
$B_{\mathrm{a,w}}$ (Mbps)	50	Job to A-Fog WiFi bandwidth
B_{d} (Mbps)	30	Job to D-Fog bandwidth
B_{c} (Mbps)	20	Job to cloud bandwidth
P_{wifi} ($/GB)	0.01	WiFi price
P_{cell} ($/GB)	1	cellular price
$P_i^{\mathrm{a,d}}$ ($/hour)	[1.6,2.4]	A-Fog i computer usage price
$P_i^{\mathrm{a,p}}$ ($/hour)	[0.8,1.2]	A-Fog i Raspberry Pi usage price
$P^{\mathrm{d,c}}$ ($/hour)	0.1 / 0.06	D-Fog / cloud CPU usage price
$P^{\mathrm{d,m}}$ ($/hour)	0.04 / 0.03	D-Fog / cloud memory usage price per GB

concerned on the utility (or revenue) generated from the analytics services. Additionally we proposed a metric called user satisfaction to measure the level of user satisfaction with the analytics service, on a scale of 0 to 1, with 1 being most satisfied, which means the matched fog object is the user's most preferred one.

In the next two subsections, experiment results over pure A-Fogs and HMCM are presented respectively.

2.6.3 Experiment Results with A-Fogs only Model

In this set of experiments, only jobs for A-Fogs presented in Table 2.2 are generated and processed. Typical results of job blocking probability, user satisfaction level and service utility with the distributed ACRA algorithms over 20 A-Fogs are measured and presented in Fig. 2.13. It is noted that each result shown in the figure is obtained by averaging over 50 simulations. Each simulation stops until 20000 jobs are successfully completed.

According to the results in Fig. 2.13, it can be observed that compared to the two baseline distributed ACRA algorithms (random and user preference based algorithms), the proposed matching theory based algorithm has the best overall performance, including the lowest job blocking probability, the highest service utility and high user satisfaction level. In general the job blocking probability and service utility increase proportionally to the job arrival rate while the service utility shows the opposite. When the job arrival rate is no more than 1 job per second, the matching theory proposed method has a blocking probability of less than 0.05, which is much smaller than the 0.18 and 0.25 of the user preference based and random algorithms, respectively. In addition the service utility of matching and user preference based algorithms is much larger than the random algorithm. The results demonstrate the effectiveness of the prefer-

ence computation method and superior performance of the matching algorithm, with improved data analytics service capacity and resource utilization.

In the above experiments the number of A-Fogs is fixed to 20. Next we run a set of experiments to check the impact of the number of A-Fogs on system performance. We vary the number of A-Fogs from 5 to 25 with a step of 5. Again the A-Fogs are uniformly distributed in the network area. Typical results are presented in Fig. 2.14 with job arrival rate of 1 job per second. It can be observed that with increasing number of A-Fogs, the computing system performance is improved significantly from all the three investigated aspects. For example, the blocking probability is more than 0.68 for all algorithms with 5 A-Fogs, while the service utility is smaller than 0.13 dollar/second. With 25 A-Fogs, the job blocking probability drops significantly to 0.01 for matching algorithm and 0.1 for random algorithm;and the service utility for matching algorithm increases to 0.44 dollar/second. The results demonstrate the feasibility of data analytics with A-Fogs and HMCM. In addition, matching algorithm shows superior performance over the two baseline algorithms over all the settings of the number of A-Fogs.

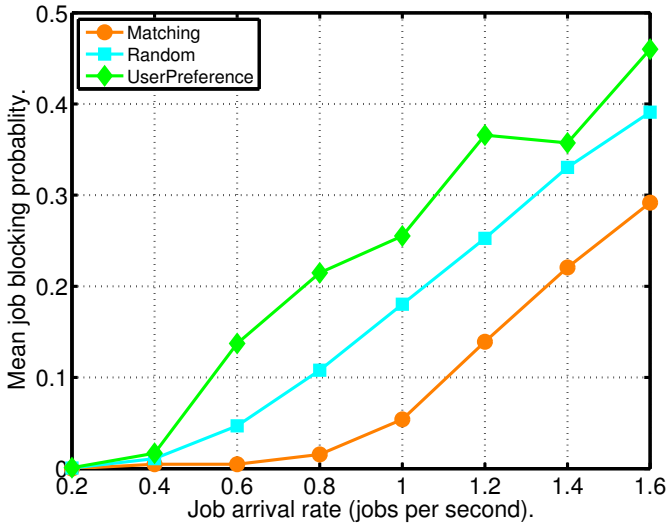
As mentioned previously, the matching period T_{m} has a large impact on service quality and resource utilization. In order to quantify the impact, experiments are run with varying matching period T_{m} = 30, 60, 90, 120 seconds. Then the performance in terms of job blocking probability and service utility with the proposed matching theory based ACRA algorithm is compared with different T_{m}. Experiment results with 20 A-Fogs are presented in Fig.2.15. It can be observed that the job blocking rates with 30 and 60 seconds matching period are very close and much lower than the longer matching period. In addition the service utility rate with 60 seconds matching period is higher than the 30 seconds one. Therefore 60 seconds is set as the default value for matching period in the experiments.

2.6.4 Experiment Results with hybrid mobile Computing Model

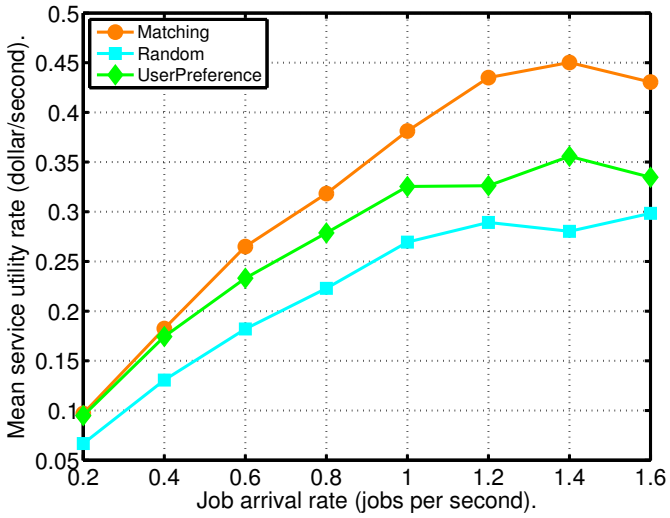
Next the feasibility and effectiveness of data analytics over HMCM are assessed. The HMCM system has 20 A-Fogs, 1 D-Fog and 1 cloud. In this case, only matching theory based ACRA algorithm is included for investigation as it largely outperforms the two baseline algorithms. In this set of experiments both A-Fog datasets and D-Fog datasets shown in Table 2.2 and Table 2.5 are generated and processed. For performance comparison purpose, four different hybrid mobile computing environments are investigated: 1) with A-Fogs only (labelled 'A-Fog' in the following figures); 2) with all the three types of computing centers (A-Fogs, D-Fog and cloud, labelled 'All'); 3) with only D-Fog and cloud (labelled 'D-Fog+cloud') and 4) with only cloud (labelled 'cloud').

Typical results of blocking probability and service utility with these computing environments are presented in Fig. 2.16(a) and Fig. 2.16(b), respectively. Further details on the proportion and distribution of the jobs completed at the different computing environments under the computing environment 'All' are presented in Fig. 2.17.

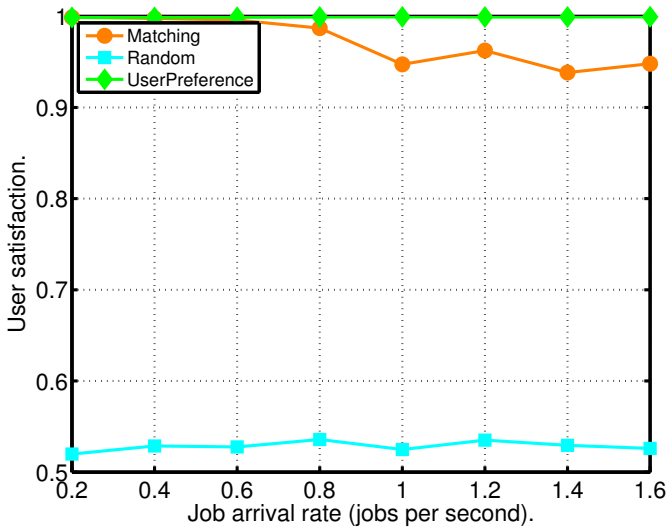
From the results presented in Fig. 2.16 and Fig. 2.17, we have the following observations:



(a) Mean job blocking probability.

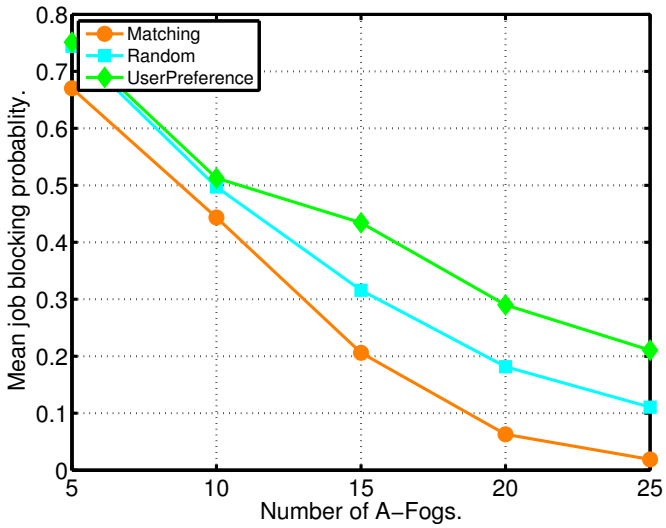


(b) Mean analytics services utility.

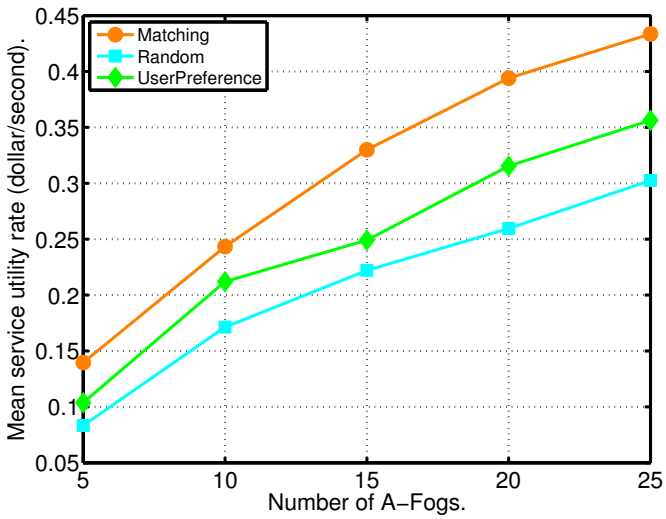


(c) Mean user satisfaction.

Figure 2.13: Analytics services performance with different matching methods versus job arrival rate over A-Fogs only environment. a) job blocking probability; b) service utility; c) user satisfaction

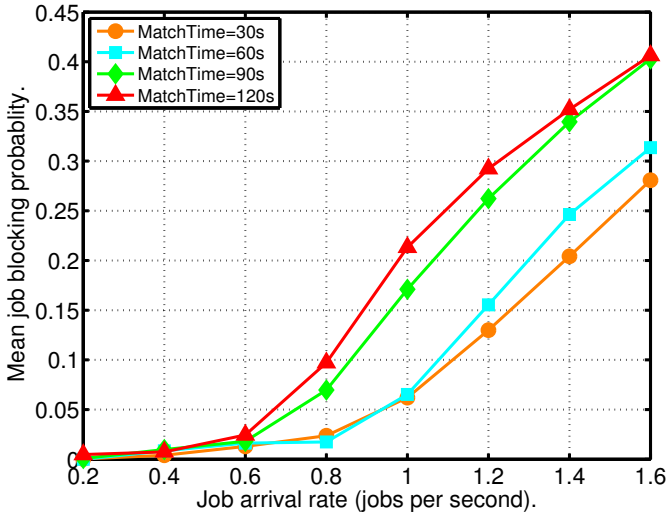


(a) Mean job blocking probability.

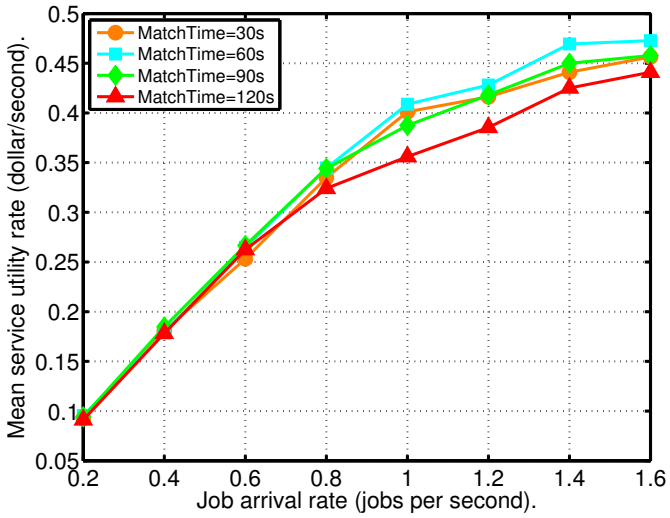


(b) Mean analytics services utility.

Figure 2.14: Analytics services performance with different matching methods versus number of A-Fogs. a) job blocking probability; b) service utility



(a) Mean job blocking probability.



(b) Mean analytics services utility.

Figure 2.15: Performance with different matching methods versus matching period. a) job blocking probability; b) service utility

- For the data analytics services, HMCM can provide good service quality. Although the cloud has virtually unlimited computing resources, it does not provide satisfactory service quality with the investigated system configurations. The job blocking probability with the cloud only environment is around 0.2 irrespective of job arrival rates, while it is less than 0.02 for the HMCM computing environment.

- Due to the limited computing resources at the A-Fogs, the blocking probability with only A-Fogs increases fast with job arrival rates larger than 0.2, mainly because A-Fogs are not powerful enough to process the larger D-Fog datasets.

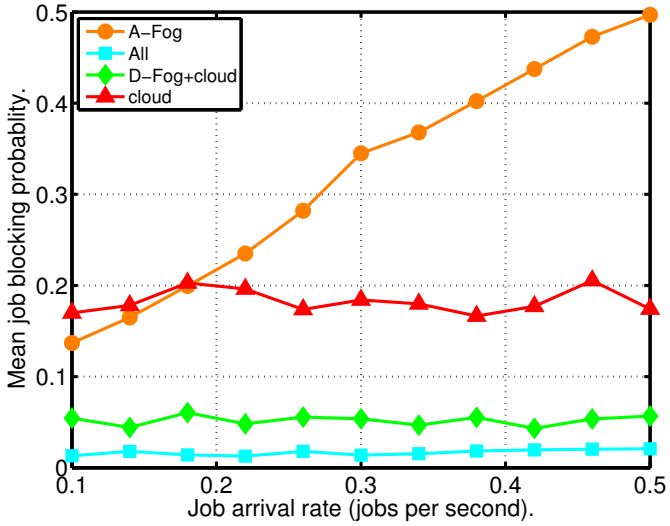
- With the hybrid mobile computing environment (including A-Fogs, D-Fogs and clouds), the overall analytics service has the lowest blocking probability and the highest service utility for all the job arrival rates. The service utility is more than double of the cloud only system. The A-Fogs only computing system also achieves a very high service utility, which is much higher than the D-Fogs and clouds based computing system. The results demonstrate that A-Fogs can play an very important role in providing fast and economical solution for data analytics services in CAV applications.

- Compared to the A-Fogs only architecture, the 'D-Fog+cloud' architecture has lower utility rate, which is close to that of cloud only solution. But its job blocking rate is also very low, which is around 0.05 for most job arrival rates. Therefore we can say D-Fogs with dedicated resource is a key and efficient in guarantee data analytics quality with small job blocking probability, but it may not be an economic solution compared to A-Fogs.

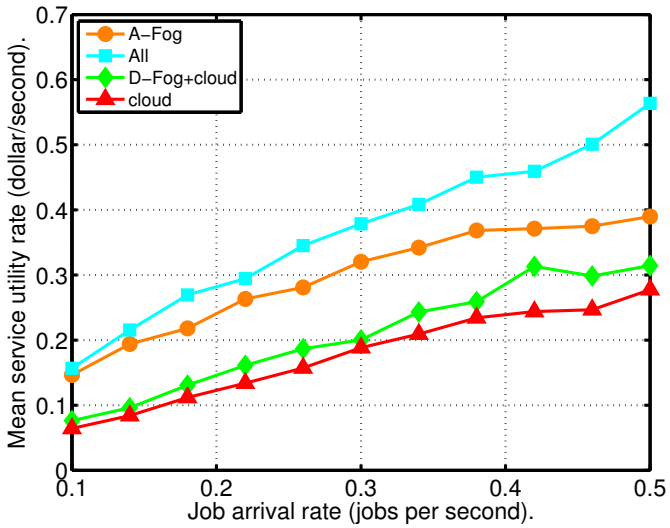
According to the above experiment results, it is demonstrated that data analytics for CAV applications is feasible over HMCM. The A-Fogs with opportunistic computing and communication resources can provide economic solution to data analytics services, while D-Fogs with dedicated computing resource can provide excellent service quality for the services. The important roles of A-Fogs and D-Fogs can also be observed from the proportions of these fogs used to execute the data analytics jobs from Fig. 2.17. With the proposed HMCM which integrates A-Fogs and D-Fogs and efficient matching based ACRA algorithm, satisfactory data analytics services and high service utilities could be achieved. It is noted that if more powerful computing devices are available for the A-Fogs and D-Fogs, which is highly likely to happen, even better performances can be delivered for HMCM.

2.7 Conclusion

In this chapter we proposed a hybrid mobile computing model (HMCM) based data analytics service for CAV applications. In the HMCM there are A-fogs with opportunistic computing resources, D-fogs with dedicated computing resources and traditional clouds. A-Fogs and D-Fogs can cooperate to increase improve resource utilization and data analytics capacity in terms of the number and size of processed analytics jobs. The benefits, key enabling technologies



(a) Mean job blocking probability.



(b) Mean analytics services utility.

Figure 2.16: Analytics services performance over hybrid mobile computing model against job arrival rate. a) job blocking probability; b) service utility.

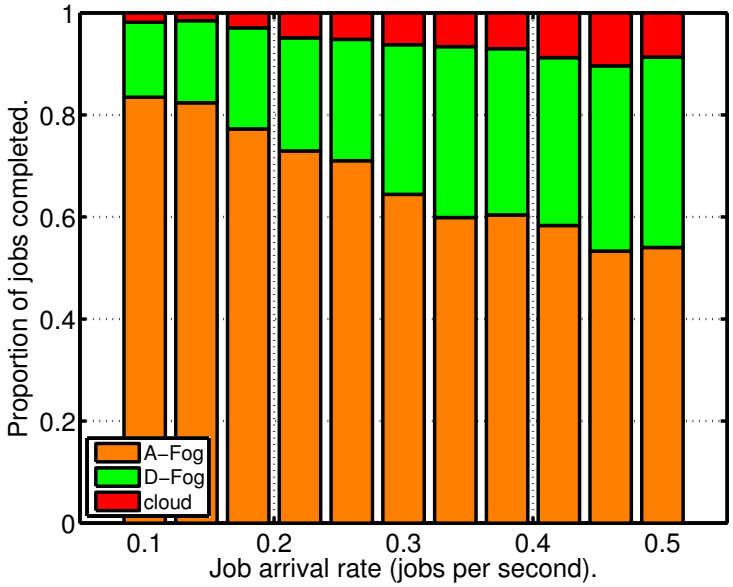


Figure 2.17: Proportion of jobs completed in different computing environments to those in the 'All' computing environment.

and use cases with HMCM model were discussed. Then we developed a framework for data analytics service and resource management, which is a key research challenge for HMCM. An optimization problem for QoS aware admission control and resource allocation (ACRA) was formulated. Distributed ACRA algorithms with and without fog cooperation were proposed to solve the ACRA problem, including two baseline non-cooperative algorithms and a matching theory based cooperative ACRA algorithm. In the ACRA algorithms, constraints and prices of computing and network resources were jointly considered. Experiment results demonstrated the feasibility of large scale data analytics services with HMCM and significant improvement on the data analytics capacity and service utility with the cooperative ACRA algorithm. We developed a system level simulator with workload models created from benchmark experiments over fogs and distributed compute engine Spark. It is observed from experiments that in the HMCM system A-Fogs is very effective in service cost reduction, while D-Fogs plays a key role in maintaining a very low job blocking probability. The results demonstrate the feasibility and efficiency of large scale data analytics over HMCM and show the superior performance of the matching theory based ACRA algorithm. In our future works mobility management and more design options will be investigated.

3 Object Detection with Mobile Local Computing

3.1 Introduction

Generally many driving safety applications such as collision detection, lane keeping and emergency braking, are all based on real time environment sensing. Two major distinct features of these applications are that vehicle sensors are the sources of data generation and expected response time is fast. If the processing and analytics of the data are performed in the clouds or fogs, the raw data from vehicles needs to be transported over the Internet to the clouds or fogs for analysis, which can incur high traffic load, large process and response latency and costs. In addition, it is possible that the connections of vehicles to the Internet may not exist or have very limited network bandwidth, such as in the scenarios of remote area and severe weather. Under theses conditions the expected data analytics services for CAV applications may not be provided through the public clouds or fogs. On the other hand, vehicles are equipped with increasing computing and storage resources to support autonomous driving and advanced driving assistance. In order to solve the problems for these real time CAV applications, a simple and straightforward solution is based on mobile local computing, which performs the data analytics at vehicles directly.

Detecting surrounding objects such as lane, other vehicles, pedestrians and traffic signs is the basis for most driving safety applications. Because of the strict real time requirements, these object detection tasks are better processed by vehicles themselves. In this chapter we focus on the design and optimization of visual object detection model with mobile local computing.

Visual object detection is a long standing and important research problem for computer vision, with a wide range of real world applications, such as robotic vision, surveillance, advanced driving assistance systems (ADAS) and autonomous driving [98]. Its main task is to predict the position and category of interested objects from images or videos. Traditionally hand-crafted features have been used to detect multiple classes of objects, e.g., over challenge datasets PASCAL [99] and COCO [100]. Deformable parts model (DPM) is one of the most successful traditional object detection approaches [101]. However, since AlexNet achieved huge success in the Imagenet challenge in 2012 [102], convolutional neural networks (CNN) quickly becomes the dominant object detection approach.

Despite fast growth of CNN in object detection over datasets with a large number of object classes, real time visual object detection in driving environment is still very challenging. It

is observed that popular CNN detectors including Faster-RCNN [103] and SSD [104] do not perform very well over the KITTI benchmark datasets [98]. In addition to radar and Lidar based object detection, camera based visual object detection, which is the focus of this work, provides an economic solution and is also a critical component of hybrid solution for ADAS and autonomous driving. There are many key challenges on visual object detection for autonomous driving as discussed below, which may not present in the other object detection datasets.

- Most object detection applications for ADAS and autonomous driving require extremely high detection accuracy and fast response time. While high false positive ratio (non-targets are falsely detected as targets) or excessively delayed detections are annoying, which may lead to close of the detection based safety applications, high false negative ratio (targets are not detected) can have fatal consequences and should be avoided as much as possible.

- Driving environment is very harsh for visual object detection with poor illumination and weather conditions. Unlike that there are only a few large target objects in an image in datasets such as PASCAL, there can be many occluded and truncated objects with large object scale variations in ADAS images. Example images with occluded and truncated cars are shown in Fig. 3.1.

- Apart from the accuracy performance requirement, computation speed is also a big concern for ADAS object detection with mobile local computing. Vehicles are unlikely to be equipped with GPU computers as powerful as used in research environments. Accuracy often has to be compromised due to computation complexity of advanced CNN detectors.

(a)

(b)

Figure 3.1: Example difficult images for object detection.

In view of the above research challenges, in this chapter we propose the following methods to the multi-scale CNN (MS-CNN) model [105] to improve visual object detection performance for ADAS.

- In the existing MS-CNN model, feature map from feature output scales are processed separately to predict existence of objects at fixed scales. In this chapter deconvolution of CNN features is applied at smaller feature output scales, which is further fused with features at larger feature output scales, to provide richer context for object detection at individual feature output scale. Such a method can effectively address the large object scale variation challenge.
- In most of existing CNN detectors, non-maximal suppression (NMS) method is used for suppression of overlapping object proposals. With such process there is very little chance for proper detection of occluded objects. But in driving environment occluded objects are normal and are potential driving hazards. To address the object occlusion challenge soft-NMS is applied at object proposals from different feature output scales to strike a balance on the number and quality of object proposals.
- In the existing CNN detectors, default anchor boxes with certain sizes are used to generate object proposals. In the driving environment the interested objects have strong features in shape, for example, the width of a car should not exceed lane width. The distributions of the object aspect ratio can be utilized for anchor box settings. We measure the aspect ratio statistics of objects from KITTI training samples and find proper anchor box settings by exploiting the statistics for better object localization and prediction.

The proposed CNN methods are individually and jointly evaluated with various image input sizes by extensive experiments over KITTI benchmark dataset. Good detection performance improvement is observed with both individual and combined CNN methods. Compared to the published works over KITTI benchmark test dataset our proposed method ranks the first for pedestrian detection category “Easy” and second for categories “Moderate and “Hard”, and is the fastest among the top ten ranked published methods. The object detection time with a local GPU computer is 0.08 second per 384×1280 sized image, which can satisfy the real time requirements of driving safety applications.

3.2 Related Works

Visual object detection is a long term research problem. Classic object detectors use hand-crafted features, such as histogram of oriented gradients (HOG) [106], integral channel features (ICF) [107] and aggregated channel features (ACF) [108]. From the aspect of feature enhancement, [109] introduces spatially pooled features to improve the feature robustness. [110] proposes a pedestrian detector by computing features at multiple image scales. A graph-based algorithm in [111] generates proposals of vehicles with better quality than other traditional region proposal approaches [112, 113]. DPM is the latest successful classic object detector with

significantly improved detection accuracy. However the computation complexity of DPM is still very high and it does not perform well for driving object detection.

While classic object detection gets stuck in a bottleneck, there is a huge breakthrough on visual object detection with deep learning models, especially CNN models. Powered by the powerful GPU computers and huge object detection samples, CNN models can automatically learn complex and efficient features from sample images. Widely successful CNN models and applications have been reported within the past several years. In general CNN based object detectors fall into two frameworks: one-stage and two-stage.

Currently two-stage detectors produce the state-of-the-art performance in object detection tasks like PASCAL, COCO. In the line of two-stage CNN detectors, RCNN [114] is a pioneer CNN model, which improves object detection performance over classic detectors by a large margin. In the first stage, RCNN applies selective search method [112] to generate sufficient proposal candidates that contain all the objects. In the second stage, RCNN forwards each proposal through convolutional networks, followed by classifying the proposals with SVMs and predicting bounding boxes offsets with linear regression. Fast-RCNN [115] extends RCNN by using one single convolution network to perform shared computation in the second stage, which increases the speed significantly. Furthermore, Faster-RCNN [103] proposes region proposal network (RPN) to replace selective search method in RCNN and makes the whole network trainable in an end to end approach. In addition, many other variants of RCNN-style approaches are proposed [116–118].

On the other hand, one-stage detectors are faster and easier to train while yielding inferior performance. SSD [104] skips the region proposal stage and directly uses multiple feature maps with different resolutions to perform object localization and classification. YOLO [119] is another one-stage detector that can achieve even faster speed at the expense of accuracy. By introducing improvements of batch normalization, high resolution classifier, convolutional with anchor boxes and dimension clusters to YOLO, YOLOv2 [120] achieves higher accuracy and higher speed.

In the latest research on CNN models, there are increasing interests on exploiting multiple scales feature maps. Based on the conventional pyramidal feature hierarchy in convolutional networks in Faster-RCNN, [117] adds a top-down pathway and lateral connections to merge feature maps from different level. The objective is to strengthen the representational power of low-level feature maps with the semantics conveyed from high-level ones. With this adaptation in Faster-RCNN, [117] shows considerable improvements on the COCO detection benchmark. Similar idea is applied to SSD in [121], where DSSD is proposed to utilize feature maps from smaller scales with more semantics. A fully evaluation of DSSD is conducted with different feature map concatenation approaches, including feature maps pooling and deconvolution [122].

The huge success of deep learning and CNN technologies significantly boost research and development of autonomous driving. The popular models are applied and enhanced for object detection in driving environment. It is noted that the popular models including Faster-RCNN, SSD, YOLO, YOLOv2 did not produce good results over the KITTI test dataset. But with certain

modifications and adaptations, the variants of Faster-RCNN and SSD models are taking the top entries in the KITTI object detection leader board. For example, [123] improves the region proposal quality with resource to subcategory information. As it is hard for Faster-RCNN to handle the large object size variation, which is designed to detect all the objects on a single layer, MS-CNN [105] extends the detection over multiple scales of feature layers, which produces good detection performance improvement. Scale dependent pooling and cascaded rejection classifiers are used in [124]. In [125], authors propose a recurrent rolling convolution (RRC) architecture on top of SSD model, which produces top detection performance for pedestrian detection. However, it is noted that the RRC model is very complex and significantly increases computation time.

Our work presented in this chapter are different from the above reported enhancements over KITTI benchmark tests. We use MS-CNN as a baseline network model and add three enhancement building blocks, which show considerable object detection performance improvement but with negligible additional object detection time.

3.3 Network Architectures

In this section we present the overall architecture of the modified CNN model and the proposed methods.

3.3.1 Overall Architecture of the Modified CNN Model

3.3.1.1 General CNN Building Blocks

Generally speaking CNN is a class of deep feed-forward artificial neural networks that use multiple layers of neurons. A CNN consists of an input layer, an output layer and multiple hidden layers. For the considered visual object detection task, the CNN input layer is an image with size $H \times W \times D$, where H and W denote image height and width in pixels, and D denotes the number of color components. For a typical KITTI image, the value of H, W and D is 384, 1280 and 3, respectively. The output layer produces a number of detected objects with associated attributes of object category and bounding boxes. Fully connected layer is usually included in the output layer, used to implement high-level reasoning based on deep feature maps.

A CNN hidden layer typically consists of convolution layers, pooling layers, ReLU layers. Convolution layer is the core building block of CNN. It includes a set of learnable filters which have a small receptive field but are applied with convolution to the whole input. A convolution layer can be characterized by its filters in 3 dimensions: filter height h and width w, and depth (or channels) d, which denotes the number of filters. The filters are shared (each filter is conveyed across the entire visual fields) and have local connectivity (each neuron is connected to only a small region of the input volume). They can learn features with very little image pre-processing which are hand-engineered in traditional algorithms. By applying a filter to the whole input volume a so-called feature map can be produced.

Pooling layer is used for non-linear down sampling. It usually follows convolution layer. Max pooling is the most common pooling function, which is used in the CNN investigated in this chapter. With max pooling the input feature map is partitioned into a set of non-overlapping rectangles and for each rectangle outputs the maximum feature value. ReLU layer applies non-saturating activation function $f(x) = max(0, x)$, which is used to increase nonlinear property for decision making and the overall network.

3.3.1.2 Key Building Blocks of the Modified CNN Model

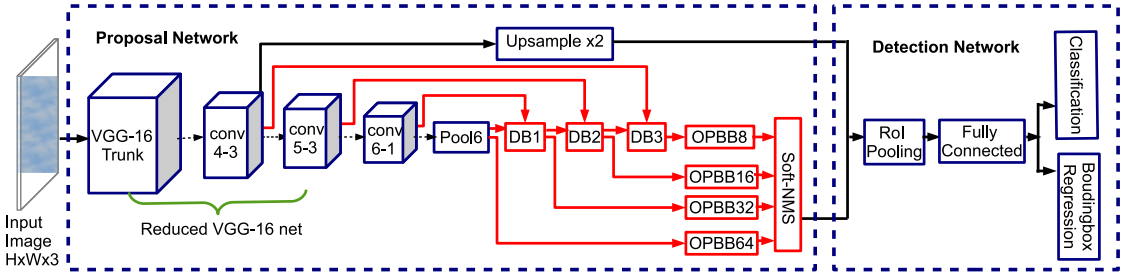


Figure 3.2: Overall pipeline of enhanced MS-CNN model.

The main building blocks of the modified CNN model is presented in Fig. 3.2. The baseline network is MS-CNN [105], which detects candidate objects at multiple feature output layers with different scales. To differentiate from the MS-CNN, the proposed methods are highlighted by red boxes in Fig. 3.2. It is noted that the proposed methods to MS-CNN are general and are applicable to other CNN models such as Faster-RCNN and SSD as well.

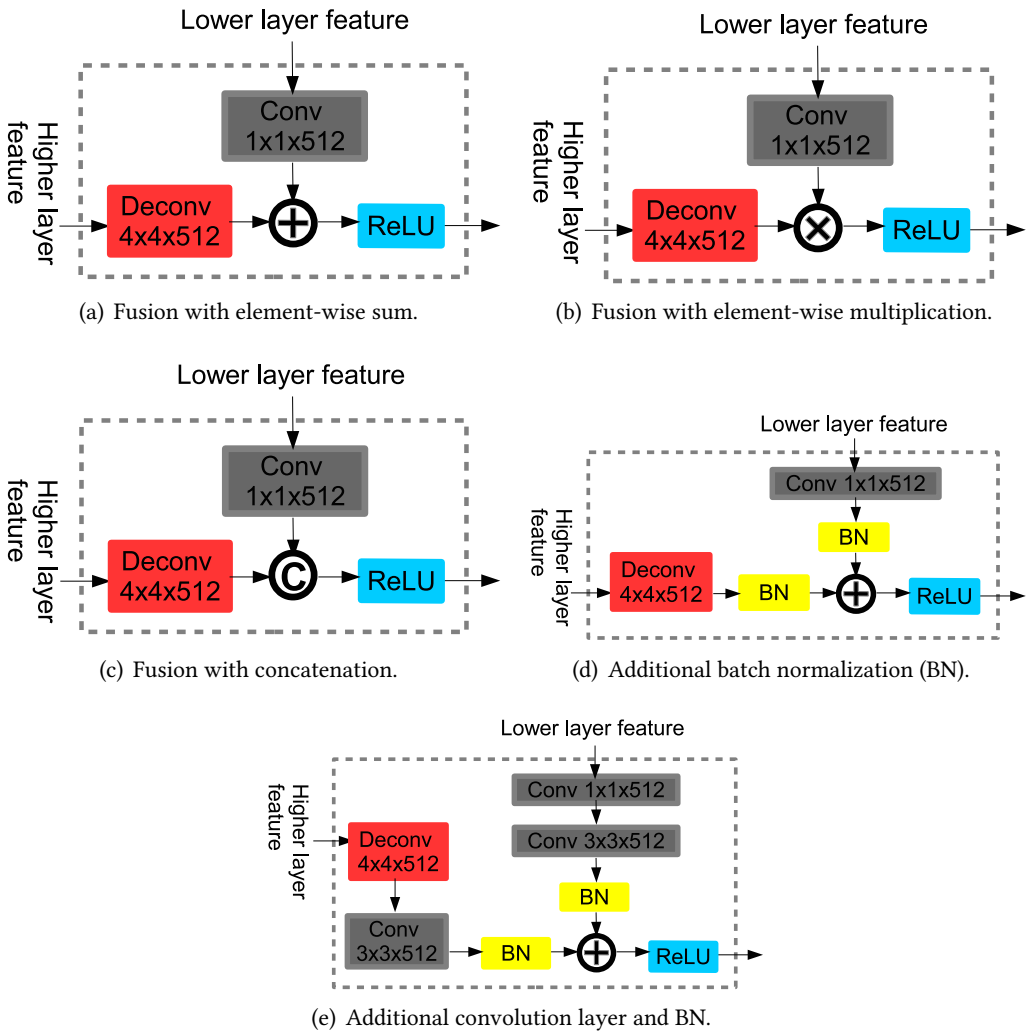
The proposed network follows the popular two-stage object detection network architecture, which consists of an object proposal network and an object detection network. The proposal network layers are based on the popular reduced VGG-16 net [126], which has 16 weight layers in its original form. Additional convolution layers, pooling layer, proposed deconvolution layers and object proposal layers are added on top of the reduced VGG-16 net. It is noted that only a few convolution and pooling layers from hidden layers are presented in Fig. 3.2 for better visualization. The feature outputs of these layers are directly used for object proposal. The layers selected as feature output layers are labeled as “conv4-3”, “conv5-3”, “conv6-1” and “Pool6”, respectively. The first number in the labels such as 4 and 6 represents the associated hidden layer in VGG-16 net, and the second number represents the ID of the convolution layer in a hidden layer. As the feature output layers are not directly connected (with separation by other convolution layers or pooling layers), dotted lines are used to connect them in Fig. 3.2.

The original feature outputs are further processed by the deconvolution building blocks (DBB), shown as “DB1”, “DB2” and “DB3” in Fig. 3.2, to aggregate feature maps from adjacent layers, before being used in object proposal building blocks (OPBB). Each OPBB produces a fixed-size set of proposals including coordinates with respect to the pre-defined anchors and scores of objectiveness. Then a soft-NMS building block is used to remove redundant proposals with heavy overlapping. In the original MS-CNN model, NMS is used to remove redundant proposals. The new building blocks (DBB, OPBB and soft-NMS building blocks) will be introduced

in details in the following subsections.

The object detection network has a region of interest (ROI) pooling layer and a fully connected (FC) layer. The outputs of upsampled feature maps from the lowest output feature layer (i.e. “conv4-3") and object proposals from soft-NMS building block in the proposal networks are used as input to the detection networks. The ROI pooling layer extracts the feature maps of the object proposals using these inputs. It is noted that the feature maps from “conv4-3” are upsampled twice to improve the capacity for location-aware bounding box regression. Then a fully connected layer maps the ROI feature maps into fixed vectors for classification and bounding box regression.

3.3.2 Deconvolution Building Block (DBB)



(a) Fusion with element-wise sum.

(b) Fusion with element-wise multiplication.

(c) Fusion with concatenation.

(d) Additional batch normalization (BN).

(e) Additional convolution layer and BN.

Figure 3.3: Various feature fusion methods for deconvolution building block (DBB).

MS-CNN exploits multi-scale features to produce predictions of different scales, which showed improved object detection performance over Faster-RCNN and SSD for KITTI datasets. It is a good idea to use the feature maps at larger scales (lower CNN layers) with smaller receptive

fields to detect smaller objects and those in smaller scales (higher layers) to detect larger objects. However, it is noted that shallow feature maps from the low layers of feature pyramid inherently lack fine semantic information for object recognition. There is an opportunity to augment the shallow feature maps with deeper feature maps from higher feature output layers and improve detection performance.

We propose to add DBB to the baseline MS-CNN model, with additional deconvolution layers and lateral connections to aggregate feature outputs from different layers. Using DBBs the semantics from higher layers can be conveyed into lower layers to increase the representation capacity. There are three DBBs used in the proposed CNN model. Fig. 3.3(a) illustrates the architecture of the DBB used in this chapter, which connects one feature output layer with its adjacent higher layer counterpart. Specifically, we first connect a convolution layer ("Conv $1\times1\times512$") with 512 1×1 filters to an output feature layer as shown in the Fig. 3.3(a). In addition, in the horizontal direction, a deconvolution layer ("Deconv $4\times4\times512$") with 512 4×4 filters is applied to upsample the corresponding higher-level feature maps. Then the outputs of these two associated feature layers, which have the same spatial size and depth, are merged by element-wise sum and processed by a ReLU layer to produce a new output feature layer. In order to maintain feature aggregation consistence, the number of channels is set to 512 in all DBBs.

It is noted that there are many possible architecture designs for DBBs. For example, for a given feature output layer, the output feature maps can be merged with those from both higher layers and lower layers. However the computation complexity and memory requirement can be increased significantly. We examined and compared four alternative DBB architectures, which are shown in Fig. 3.3. Two alternative architectures shown in Fig. 3.3(b) and Fig. 3.3(c) are very similar to the one adopted in this chapter as shown in Fig. 3.3(a), with the only difference of using element-wise multiplication and concatenation, respectively. Two more alternative DBB architectures are shown in Fig. 3.3(d) and Fig. 3.3(e). The implementation shown in Fig. 3.3(d) adds a batch normalization (BN) function block after the convolution and deconvolution layers in the DBB shown in Fig. 3.3(a), while the implementation shown in Fig. 3.3(e) adds a convolution layer "Conv $3\times3\times512$" and a BN function block. However, according to results from extensive experiments, it is found that the implementation shown in Fig. 3.3(a) has the best detection performance and low computation complexity. The architectures shown in Fig. 3.3(d) and Fig. 3.3(e) even have negative impact on detection performance. The design of DBBs is not straightforward and specific consideration is needed for different baseline CNN models.

3.3.3 Object Proposal Building Block (OPBB)

3.3.3.1 OPBB Architecture

The functionality of OPBB is to receive feature map output from the DBBs or Pool6 layer and produce high quality proposals to be further processed by the soft-NMS building block. In this chapter we have 4 OPBBs which have the same architecture but different parameters. These OPBBs are labeled as "OPBB8", "OPBB16", "OPBB32" and "OPBB64" as shown in Fig. 3.2. The

number in the OPBB labels is the ratio of the original image size to the spatial size of the feature map input to the OPBBs.

Inside each OPBB there are several similar process pipelines, each associated with one type of anchors. The overall architecture of an OPBB with two types of anchors is shown in Fig. 3.4. For the first anchor related pipeline, the input feature maps from DBB go through two separate processes: one for classification having a convolution layer with $h1 \times w1 \times (C+1)$ filters and a softmax module, and the other for bounding box regression with respect to the anchor having a convolution layer with $h1 \times w1 \times 4$ filters. The classification process path produces the softmax scores of C object classes and background class for each feature map location. The regression path produces a bounding box estimation for each feature map location. Then the anchors with estimated classification scores and the bounding box for each feature map location are processed to form good quality proposals.

At each feature map location l, there are two proposals, p_l^n for $n \in \{1, 2\}$, produced from the two anchor pipelines. Each proposal has $(4 + C + 1)$ dimensions, among which 4 dimensions are for bounding box coordinates and C+1 dimensions are for classification scores of each class. The 4 coordinates represent the offsets relative to the associated anchor coordinates. Let B_l^n denote the coordinates vector for proposal p_l^n, $n \in \{1, 2\}$. Let L denote the class label set, $L = \{0, 1, 2, ..., C\}$. Label 0 refers to the background class. Let $F_l^n = (f_l^{n,0}, f_l^{n,1}, ..., f_l^{n,C})$ be the classification score vector for p_l^n, where $n \in \{1, 2\}$, $f_l^{n,c}$ denotes the classification score for class c. Classification score measures the probability distribution over $C + 1$ classes. Then a proposal p_l^n at location l can be denoted by $p_l^n = (B_l^n, F_l^n)$.

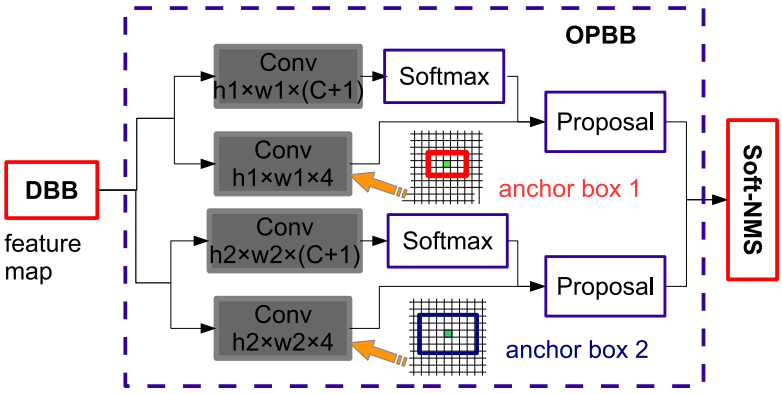
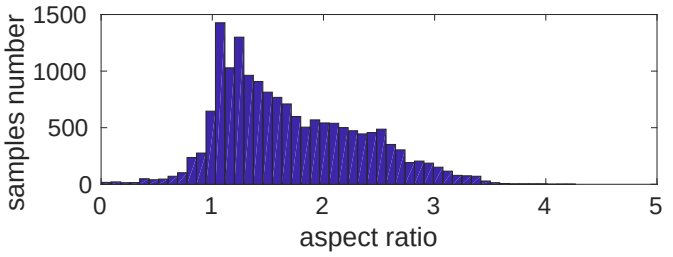


Figure 3.4: Object proposal building block.

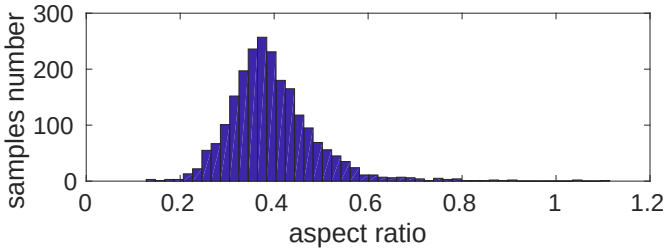
3.3.3.2 Anchor Boxes

Anchor boxes are critical component of the regional proposal networks for Faster-RCNN model and its variants such as MS-CNN. In the standard Faster-RCNN model there are 9 types of anchor boxes associated to one convolutional filter layer. In the baseline MS-CNN, in each OPBB, there are several convolutional filter layers and each is associated with only one type of anchor boxes. The associated convolutional filter layer and the type of anchor box correspond to one proposal pipeline in an OPBB. The aspect ratio of MS-CNN anchor boxes is set to 1 for cars and around

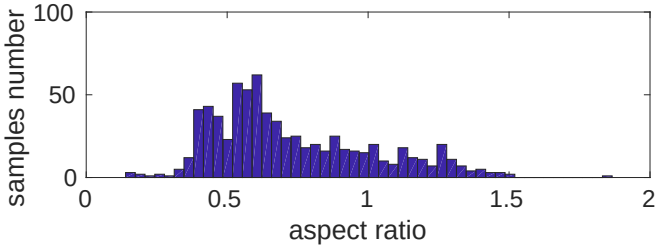
0.7 for both pedestrians and cyclists. Although the network can refine the bounding box of proposals by learning to predict the offsets to anchor boxes, a better anchor box setting will help object detection with improved matching to the ground truth bounding boxes, therefore improve both training and inference performance.



(a) Cars.



(b) Pedestrians.



(c) Cyclists.

Figure 3.5: Distribution of aspect ratios for different object classes in KITTI benchmark training set.

We develop a refined OPBB with better anchor box settings according to the object statistics analyzed over the KITTI training set. It is noted that although the experiments of the anchor box setting are conducted with MS-CNN as the baseline CNN model, the idea of setting anchor boxes with sample statistics is general and can be applied to other network architectures as well.

To get insights into better anchor box settings, we collect all the ground truth bounding boxes in the KITTI training set and generate a histogram of object aspect ratios for each object class (cars, pedestrians and cyclists). As shown in Fig. 3.5, the objects of different classes have distinct distributions of aspect ratios. Car samples have wider boxes with most aspect ratio values in the range of 1 to 3. On the contrary, the other two classes have much smaller aspect ratios. Cyclist samples show greater variation than pedestrians. Based on the observation, we resize the square anchor boxes for cars used in MS-CNN to rectangle ones, which are closer to the average aspect ratio of car samples. In addition, for pedestrians and cyclists objects detection

we add one more type of anchor box. So there are three types of anchor boxes in total for an OPBB in the refined OPBB architecture, compared to only two types of anchor boxes used in the baseline MS-CNN. The additional type of anchor box has an aspect ratio 0.5. The original anchor ratio set in the baseline MS-CNN is too narrow to efficiently cover the object variations. More details of anchor configurations are given in Section 3.4. It is noted that our new anchor settings are by no means the best fitting to the KITTI dataset. There may be optimal joint settings on the number, scale and aspect ratio of anchor boxes.

3.3.4 Soft-NMS Building Block

After the object proposal layers, soft-NMS building block is used to filter out highly overlapped proposals from the object proposal layers. As NMS algorithm has been applied to remove redundant neighbor proposals in most state-of-the-art object CNN detection models including MS-CNN, we have a brief introduction to NMS before the presentation of soft-NMS. For a proposal p, any other proposal that has an overlap more than a pre-defined threshold T with proposal p is called a neighbor proposal of proposal p. Mathematically, let $P_{\text{in}} = \{p_1, p_2, ...p_n\}$ denote an initial proposal set output from the object proposal layers, in which the proposals are sorted by their objectiveness scores. Here the objectiveness score S_i for proposal p_i is the maximum value in the classification score vector of p_i. The traditional NMS method works as follows:

- Algorithm input proposal set P_{in} and output proposal set P_{out}. P_{out} is initialized to an empty set.
- Step 0: Create a temporal proposal set P_{temp}, which is initialized to P_{in}.
- Step 1: Check if any proposal remains in proposal set P_{temp}.
- Step 2: If yes, go to Step 3; else, terminate the NMS process and return output P_{out}.
- Step 3: Move the first proposal (with the highest objectiveness score) in P_{temp} to P_{out}, which is called winning proposal, denoted by p_{win}.
- Step 4: Update set P_{temp} by removing all the neighbor proposals of proposal p_{win} from set P_{temp}.
- Step 5: Go to Step 2.

In many object detection challenge datasets neighbor proposals usually correspond to the same object. But due to heavy object occlusion in KITTI dataset, NMS may remove positive proposals unexpectedly. For example, there are two proposals p_1 and p_2 with large overlap in Fig. 3.6. The proposal p_2 for the occluded back car may be removed with high probability by the traditional NMS method. To address the NMS issue with occluded objects, we apply soft-NMS for suppression of overlapped objects [127]. With soft-NMS the neighbor proposals of a winning proposal are not completely suppressed. Instead they are suppressed according to updated objectiveness scores of the neighbor proposals, which are computed according to the

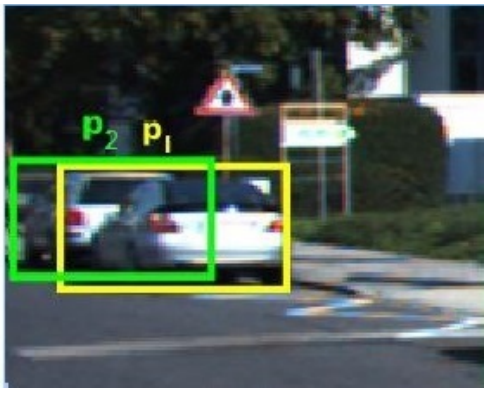


Figure 3.6: Example of overlapped proposals.

overlap level of the neighbor proposals and the winning proposal. NMS can be viewed as a specific case of soft-NMS, in which the updated objectiveness scores of the neighbor proposals of a winning proposal are simply set to zero.

Let p_i be a winning proposal and p_j be a neighbor proposal of p_i. Let S_j be the objectiveness score of p_j computed from object proposal layers. The update objectiveness score of p_j (denoted by S_j^{u}) is computed with a linear function by the following formula (3.1) [127]:

$$S_j^{\mathrm{u}} = S_j(1 - O_{p_i,p_j}), \tag{3.1}$$

where O_{p_i,p_j} represents the intersection of union (IoU) between p_i and p_j. O_{p_i,p_j} is computed by the following formula:

$$O_{p_i,p_j} = \frac{area(p_i \cap p_j)}{area(p_i \cup p_j)}. \tag{3.2}$$

As a whole, the term $1 - O_{p_i,p_j}$ acts as a weighting function with higher overlap leading to larger penalty to objectiveness score for neighbor proposals.

The operation of soft-NMS method is presented below.

- Algorithm input proposal set P_{in} and output proposal set P_{out}. P_{out} is initialized to an empty set.
- Step 0: Create a temporal proposal set P_{temp}, which is initialized to P_{in}.
- Step 1: Check if any proposal remains in proposal set P_{temp}.
- Step 2: If yes, go to Step 3; else, terminate the NMS process and return output P_{out} .
- Step 3: Move the winning proposal p_{win} in P_{temp} in this round to P_{out}.
- Step 4: Compute the updated score of the neighbor proposals of proposal p_{win} in P_{temp} according to (3.1).
- Step 5: Update set P_{temp} by removing the neighbor proposals of p_{win} if their updated scores are lower than a pre-defined threshold T_{s}.

- Step 5: Go to Step 2.

In this chapter, the neighbor proposal threshold T is set to 0.4 and the score updating threshold T_s is set to 0.001 for soft-NMS method by cross-validation.

3.3.5 Training and Inference

The whole network training includes two phases. Firstly, train the object proposal network with object proposal training samples. Secondly, train both the object proposal network and the object detection network. For both phases of network training, training samples with object classes and bounding boxes are needed. Next we introduce the construction of training samples, then present the loss function to be used for network training.

3.3.5.1 Training Samples

The class and bounding box of a proposal with regard to an anchor for a feature map location is mainly determined by the convolution layers in the OPBB. However, their weights are learned from training process with ground truth samples and configured anchors. Without loss of generality, let A_l denote an anchor with a given scale and aspect ratio from one type of anchor boxes centered at a feature map location l. The coordinates of the anchor includes its center (x_l, y_l), anchor width (w_l) and height (h_l). To create a training sample for this anchor, we first find the best matching ground truth box for it based on their IoU overlap. Let gt_l denote the best matching ground truth box for anchor A_l, and O_{A_l,gt_l} be the IoU overlap between anchor A_l and ground truth box gt_l. Then class label (denoted by c_l) for this anchor can be determined according to the IoU with the matched ground truth box. If O_{A_l,gt_l} is higher than 0.5, the anchor A_l is assigned a class label c_{gt_l}, which is the class label of the matched ground truth object. If O_{A_l,gt_l} is lower than 0.2, the anchor A_l is labeled as 0 (i.e., background class). Otherwise the anchor is assigned a class value of -1. The class label determination can be expressed in the following formula:

$$c_l = \begin{cases} c_{gt_l} & O_{A_l,gt_l} > 0.5 \\ 0 & O_{A_l,gt_l} < 0.2 \\ -1 & otherwise \end{cases}, \tag{3.3}$$

Note that anchors that labeled -1 will be discarded and are not used as training samples. The regression of the bounding box can be obtained from the anchor coordinates and the ground truth bounding box in a similar way presented in [103].

3.3.5.2 Training Loss Function

After the training samples are prepared, the network can be trained with properly designed loss function. In this chapter the objective loss function is to minimize the weighted sum of

localization loss L_{loc} and classification loss L_{cls} for the proposal and detection networks [105]:

$$\min\left[\sum_{l,c_l\geq 1}\lambda L_{loc}(loc_l, loc_{gt_l}) + \sum_{l} L_{cls}(F_l, c_l)\right], \tag{3.4}$$

$$L_{loc}(loc_l, loc_{gt_l}) = 0.25 * smooth_{L1}(loc_l - loc_{gt_l}), \tag{3.5}$$

$$L_{cls}(F_l, c_l) = -log(f_l^{c_l}), \tag{3.6}$$

$$smooth_{L1}(x) = \begin{cases} 0.5x^2 & if|x| < 1 \\ |x| - 0.5 & otherwise \end{cases}, \tag{3.7}$$

where l is the index of an anchor in the set of training samples, λ denotes a weight term, $F_l = (f_l^0, f_l^1, ..., f_l^C)$ is the classification score vector for proposal p_l, c_l is the anchor label class, loc_l is the bounding box coordinates for p_l, loc_{gt_l} is the coordinates of matched ground truth box. With the above objective function, the network can be trained by standard back-propagation and stochastic gradient descent strategies.

During inference process, a feed-forward pass of the network is run on the test images. The proposal network generates proposal candidates with bounding boxes and classification confidences and detection network further refines the location and class scores for proposals processed by soft-NMS.

3.4 Experiments

3.4.1 Dataset

We evaluate the enhanced CNN model over the KITTI 2D object detection benchmark dataset. The dataset contains 14999 images with 7481 for training and 7518 for testing. The image size is 384×1280 pixels. There are over 80000 annotated objects, which are divided into three categories (car, pedestrian and cyclist). Three object detection evaluation categories ("Easy", "Moderate" and "Hard") are set up for each object class, according to object height, occlusion and truncation level, which are presented in Table 3.1. For evaluation, average precision (AP) with different IoU thresholds (0.7 for car, 0.5 for pedestrian and cyclist) is used as the main metric of interest. The AP is computed as the mean precision at a set of equally spaced recall levels [99].

Table 3.1: Three object difficulty levels for KITTI dataset.

Levels	Description		
	Min. height	Max. occlusion level	Max. truncation
Easy	40 pixels	Fully visible	15%
Moderate	25 pixels	Partly occluded	30%
Hard	25 pixels	Difficult to see	50%

3.4.2 Implementation Details

As a widely adopted practice, the proposed network is fine-tuned on the reduced VGG-16 model, which is pre-trained on the ILSVRC CLS-LOC dataset [128]. We split the raw training dataset into training set and validation set for local performance evaluation.

As the number of samples for different object classes are highly imbalanced, detectors are trained separately for detection of cars and pedestrians/cyclists. The training procedure consists of two stages. In the first stage, only the proposal network is trained by 10000 iterations, with weight term λ of 0.05, initial learning rate of 0.00005, momentum of 0.9, weight decay of 0.0005. Following the proposal network training, in the second stage the whole network (including both proposal network and detection network) is trained for another 25000 iterations. The learning rate for the second stage is initially set to 0.0005 and is divided by 10 every 10000 iterations. The weight term λ is 1. The experiments are run with an Intel i7-7700k 4.20GHz server with 8 CPU cores and 32 GB memory and a Nvidia GeForce GTX 1080 GPU. Training time ranges from 6 to 10 hours for the models used in this chapter.

In order to examine the effectiveness of the proposed network methods, ablation experiments are designed and conducted. We let letters "D", "AR" and "S" denote the proposed network methods on deconvolution, anchor box resize and soft-NMS, respectively. The baseline MS-CNN network is denoted by letter "M". In the ablation experiments various methods are added on top of the baseline MS-CNN network. The network variants with baseline network and only one network method are denoted by "M+D", "M+AR" and "M+S", respectively. The network variants with baseline network and more than one network methods are denoted by"M+D+AR", "M+D+S", "M+AR+S" and "M+D+AR+S". As there are much smaller number of cyclist samples compared to those for car and pedestrian in the dataset, only car and pedestrian evaluation results are presented.

In addition to the various network methods, input layer image size impact is also investigated. We train the network with 3 input image sizes, small image 384×1280 (the original image size), medium image 576×1920 and large image 768×2560. It is noted that the enlargement of images does not increase image resolution. The experiments carried out with different input image size are denoted by the object class and the input image height. For example, experiments for car detection with image size 384×1280 are denoted by "Car-384". Anchor sizes are set differently for different types of experiments. The anchor and associated filter size configurations for different image sizes and different object classes are shown in Table 3.2. Note that the other parameters are kept unchanged through all the experiments.

3.4.3 Experimental Results on Validation Set

In this subsection we examine and compare the performance of the proposed CNN methods for object detection over KITTI benchmark dataset. As the ground truth of the KITTI test set is not publicised and only one submission of the KITTI test results to the benchmark website is allowed, performance comparison of the proposed methods is performed over the KITTI

Table 3.2: Configurations of anchor size and filter size (width×height) with different image size.

(a) car.

		OPBB8		OPBB16	
Car-384	anchor	40×24	56×36	80×48	112×72
	filter	5×5	7×7	5×5	7×7
Car-576	anchor	60×40	84×54	120×80	168×108
	filter	5×5	7×7	5×5	7×7
Car-768	anchor	60×40	84×54	120×80	168×108
	filter	5×5	7×7	5×5	7×7

		OPBB32		OPBB64	
Car-384	anchor	160×96	224×144	320×192	
	filter	5×5	7×7	5×5	
Car-576	anchor	240×160	336×216	480×320	
	filter	5×5	7×7	5×5	
Car-768	anchor	240×160	336×216	480×320	672×432
	filter	5×5	7×7	5×5	7×7

(b) pedestrian and cyclist.

		OPBB8			OPBB16		
Ped/cyc -384	anchor	28×40	28×56	36×56	56×80	56×112	72×112
	filter	3×5	3×7	5×7	3×5	3×7	5×7
Ped/cyc -576	anchor	40×60	40×84	56×84	80×120	80×168	112×168
	filter	3×5	3×7	5×7	3×5	3×7	5×7
Ped/cyc -768	anchor	40×60	40×84	56×84	80×120	80×168	112×168
	filter	3×5	3×7	5×7	3×5	3×7	5×7

		OPBB32			OPBB64		
Ped/cyc -384	anchor	112×160	112×224	144×224	224×320		
	filter	3×5	3×7	5×7	3×5		
Ped/cyc -576	anchor	160×240	160×336	224×336	320×480		
	filter	3×5	3×7	5×7	3×5		
Ped/cyc -768	anchor	160×240	160×336	224×336	320×480	448×672	
	filter	3×5	3×7	5×7	3×5	5×7	

training and validation set.

The AP results of the compared CNN models as configured in the previous subsection are reported in Table 3.3 for both car and pedestrian detection. The CNN models include the original MS-CNN with and without the proposed methods. The AP results for the detection categories "Easy", "Moderate" and "Hard" are presented in Table 3.3(a), 3.3(b) and 3.3(c), respectively. In the tables the maximal AP values from the compared CNN models for each image size are displayed in bold font. It is noted that as MS-CNN training with deconvolution building block and image size 768×2560 was not completed due to high GPU memory requirement, the results of related CNN models with DBB enhancement ("M+D+∗") are not presented for large input image size.

In addition the network inference time per image is reported in Table 3.4. It is noted that the inference speed of the original MS-CNN and the proposed CNN networks are very fast (0.08 second per image for small image size). The introduction of anchor box resize ("AR") and soft-NMS ("S") add negligible time. The deconvolution building block introduces a little extra computation time (0.01 second per image).

3.4.3.1 The effectiveness of proposed methods

First we check the effectiveness of the individual proposed methods. Comparing the results of CNN variants "M+D", "M+AR" and "M+S" to the baseline MS-CNN model "M", it can be observed that there are good performance improvement for most input image sizes and object classes. Among the individual methods, soft-NMS produces the largest and consistent performance gain for both car and pedestrian detection in most cases. The performance improvement with soft-NMS is more obvious for pedestrian detection with image size 384×1280. For example, the AP with soft-NMS increases from 76.35% for "M" to 78.96% for pedestrian detection category "Easy" with small image size. These results demonstrate the effectiveness of soft-NMS on tackling the object occlusion issues in ADAS environments. Anchor resize ("AR") method shows consistent performance gain over the baseline network as well. But the largest performance gain with "AR" comes mainly with the small image size, e.g., 5.1% performance gain with "AR" for car detection category "Easy". On the other hand, deconvolution ("D") method shows consistent performance gain for pedestrian detection and large performance gain for car detection category "Easy" with medium image size, but there is a slight performance loss for car category "Hard".

Next combinations of the proposed methods are examined. It is noted that the best AP performance is always achieved with combined network methods for all object classes, object detection categories and input image sizes. For example, for medium image size, the best network for both car and pedestrian detection is "M+D+S", "M+D+AR+S" and "M+D+AR+S" for category "Easy", "Moderate" and "Hard", respectively. These results show that the proposed methods can work together and effectively boost object detection performance.

An interesting observation is on the experiment results with combination of "AR" and "S" methods. For both car and pedestrian detection with small image size, both anchor resize and

Table 3.3: Performance comparison of CNN variants on validation set.

(a) Easy.

	Car			Pedestrian		
Image height	384	576	768	384	576	768
M	89.34	90.62	91.12	76.25	79.72	80.02
M+D	90.96	92.39	-	77.93	80.35	-
M+AR	94.44	90.47	91.38	77.97	79.92	80.25
M+S	91.77	91.09	91.50	78.96	79.82	80.41
M+D+AR	93.57	90.80	-	78.87	79.71	-
M+D+S	94.47	**94.20**	-	78.49	**80.37**	-
M+AR+S	**94.78**	92.72	**91.68**	**80.28**	79.98	**80.58**
M+D+AR+S	93.76	93.12	-	78.50	80.28	-

(b) Moderate.

	Car			Pedestrian		
Image height	384	576	768	384	576	768
M	88.84	89.86	90.04	70.57	74.68	76.49
M+D	89.00	89.74	-	71.39	75.92	-
M+AR	89.36	89.88	90.08	71.63	75.59	76.38
M+S	89.44	89.99	90.29	73.04	75.07	76.64
M+D+AR	88.93	89.92	-	72.66	76.26	-
M+D+S	89.43	89.81	-	71.50	75.80	-
M+AR+S	**89.57**	90.20	**90.35**	**73.05**	75.85	**76.93**
M+D+AR+S	89.37	**90.23**	-	72.42	**76.69**	-

(c) Hard.

	Car			Pedestrian		
Image height	384	576	768	384	576	768
M	77.59	79.04	79.86	62.58	66.55	68.02
M+D	77.22	78.80	-	63.53	68.06	-
M+AR	77.86	79.50	79.83	63.41	66.85	68.02
M+S	77.16	79.50	80.31	64.70	66.74	68.03
M+D+AR	77.26	79.56	-	64.53	67.63	-
M+D+S	**78.77**	79.20	-	63.37	67.68	-
M+AR+S	78.40	80.04	**80.39**	**64.88**	66.92	**68.41**
M+D+AR+S	78.23	**80.33**	-	64.15	**68.25**	-

Table 3.4: Average inference time for various network architectures.

	Car			Pedestrian		
Image height	384	576	768	384	576	768
M	0.08s	0.17s	0.24s	0.06s	0.14s	0.20s
M+AR+S	0.08s	0.17s	0.24s	0.06s	0.14s	0.20s
M+D+AR+S	0.09s	0.18s	-	0.07s	0.15s	-

soft-NMS methods bring performance gains: anchor resize has much larger gains for car detection, while soft-NMS has larger gains for pedestrian detection. The combination of "AR" and "S" methods have consistent and larger gains than the individual method.

3.4.3.2 The impact of input image size

Apart from the proposed network methods, it is also observed that increasing image size has a large positive impact on object detection. For any given studied MS-CNN variant, the AP performance improves with larger image size in most studied cases. There is a substantial performance gain with image size for the baseline network "M", especially for pedestrian detection. For example the AP increases from 70.57% with small image size to 76.49% with large image size.

However the performance gains with larger image size for some MS-CNN variants (such as "M+AR+S" and "M+D+AR+S") are much smaller. For the baseline MS-CNN network, the largest AP for car detection category "Easy" is 91.12% with large image size. But the enhanced network "M+AR+S" has 94.78% AP with small image size.

It is worth noting that the performance gains with large image size do not come without cost. According to Table 3.4, the average inference time per image for car detection increases from 0.08 second for small image size to 0.17 second for medium image size and 0.24 second for large image size. Similar inference time performance for pedestrian detection is observed. As the best detection performance with "M+D+AR+S" with medium image size is already very close to or even better than the best available performance with large image size, "M+D+AR+S" network model with medium image size is recommended for joint considerations on detection precision and speed. More specifically, the "M+AR+S" network architecture with small image size offers the highest speed and best detection AP (94.78% versus 91.68% with large image) for car detection category "Easy" and slightly lower AP (80.28% versus 80.58% for large image) for pedestrian. For some driving safety assistance applications with targets of detecting easy objects, such as forward collision warning, the "M+AR+S" network architecture with small image size can be the first choice.

To visually assess the effectiveness of the proposed method, some example KITTI images with annotations of detected objects by the baseline MS-CNN model (shown in the top half of each sub-picture) and our method (shown in the bottom half) are presented in Fig. 3.7. The first three rows Fig. 3.7(a)-Fig. 3.7(c) are for car detection and the last row Fig. 3.7(d) is for pedestrian detection. Compared the detection results with MS-CNN and our method, we can find that our method improves the detection performance from several aspects:

- Our method can reduce false proposals as shown in Fig. 3.7(a) and in Fig. 3.7(b). In the top half image of Fig. 3.7(a), there are two false proposals produced by MS-CNN around the orange car in the bottom left side. In Fig. 3.7(b) the MS-CNN method produce two false proposals, one in the right cluster of cars and one in the left cluster of cars. It can be found that our method can detect occluded objects more efficiently.

- Our method can detect more small objects that are missed by MS-CNN as shown in

Fig. 3.7(c). The MS-CNN method missed the remote small car on the road and a car in right shadow area. This highlights the effectiveness of deconvolution method which adds context and deeper features in the lower layers to detect small objects.

- Our method can avoid producing multiple bounding boxes for one object. For example in Fig. 3.7(d), the MS-CNN model produces two bounding boxes for each detected pedestrian.

3.4.4 Experimental Results on KITTI Test Set

Next we present the experiment results over the KITTI test set and compare our results with those of recently published approaches. As the KITTI leader board ranks the approaches based on the AP for "Moderate" detection category, we select the network "M+AR+S" with large image size (768×2560) for competition, which produced the best AP for "Moderate" category over validation set. The results are submitted to the KITTI test set evaluation server.

The AP and inference time results of our proposed method and other nine top ranked published approaches are presented in Table 3.5. A simple comparison of our own results on KITTI test data set to those on validation test shows that there are considerable performance loss possibly due to harder images in the test set. However similar performance loss was observed for the baseline MS-CNN model.

Comparing the AP and the inference time results in Table 3.5, it can be concluded that there is no absolute winner with dominant performance over all the comparison aspects. Among the compared approaches, our proposed method ranked the first in network inference speed, the best in the pedestrian category "Easy", second in pedestrian categories "Moderate" and "Hard", third in car detection category "Moderate". D_MANTA [132] ranked the first in car category "Easy". RRC [125] has four number one positions in all detection categories. However, it is noted that RRC has the second longest inference time (3.6 second), which is 15 times our inference time, even it is based on the fast SSD baseline and used much higher specification GPU computer.

It is noted that the highest AP for car category "Easy" achieved by "M+AR+S" with small image is 94.78% over the validation set, while the highest AP over the test set is only 90.49%. One reason for the performance gap is that "M+AR+S" with large image size is selected as the only model for competition. Therefore the good performance with "M+AR+S" model and small image size is compromised.

According to the object detection results presented in Table 3.5 and in KITTI benchmark website, it can be observed that the car detection performance for category "Moderate" is almost saturated with very little performance gap over the top 20 detection methods. However, there is still large performance improvement space for pedestrian and cyclist detection. For example the highest AP from the published works is 85.12% and 75.33% for pedestrian category "Easy" and "Moderate", respectively.

The main challenges of the pedestrian and cyclist detection still come from the small size, heavy occlusion or truncation of the objects. In addition other external factors like illumina-

(a)

(b)

(c)

(d)

Figure 3.7: Object detection examples on KITTI testing set with MS-CNN and our method.

Table 3.5: Performance comparison of recent published works and our method on the test set.

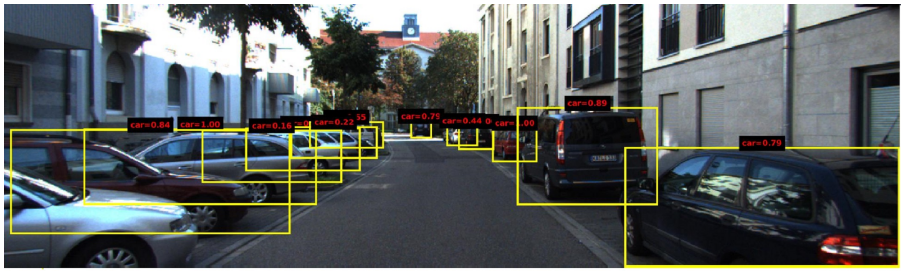
Method	Pedestrian			Car			Time (s)
	Easy	Mod	Hard	Easy	Mod	Hard	
spLBP [109]	-	-	-	80.16	77.39	60.59	1.5
Mono3D [129]	77.30	66.66	63.44	90.27	87.86	78.09	4.2
MS-CNN [105]	83.70	73.62	68.28	90.46	88.83	74.76	0.4
Deep3D [130]	-	-	-	90.47	88.86	77.60	1.5
SubCNN [123]	83.17	71.34	66.36	90.75	88.86	79.24	2.0
MV3D [131]	-	-	-	90.53	89.17	80.16	0.36
SDP+RPN [124]	79.98	70.20	64.84	89.90	89.42	78.54	0.4
D_MANTA [132]	-	-	-	**97.25**	90.03	80.62	0.7
RRC [125]	84.14	**75.33**	**70.39**	90.61	**90.22**	**87.44**	3.6
Our method	**85.12**	74.52	69.35	90.49	89.64	77.95	**0.24**

tion change and cluttered background can affect the accuracy of our detection method. And compared to the number of car samples in the KITTI dataset, the number of pedestrian and cyclist samples are much smaller, which may be another cause of the relatively poor detection performance for pedestrian detection.

We present some example images in which some samples are not correctly detected by our method in Fig. 3.8. These detection examples may help understand the existing detection challenges. In Fig. 3.8(a) the white car in the bottom left side is not detected due to heavy truncation. In Fig. 3.8(b) the cars are not detected due to their small sizes. In Fig. 3.8(c) one person nears the train is not detected due to occlusion and poor illumination conditions. Fig. 3.8(d) gives a false positive example where a motorcycle is falsely detected as a cycle.

3.5 Conclusion

Real time accurate object detection is one of the most critical problems for advanced driving assistance systems (ADAS) and autonomous driving. Recently CNN achieved huge successes on visual object detection over traditional object detectors, which use hand-engineered features. However, due to the challenging driving environment (e.g., large object scale variation, object occlusion and bad light conditions), popular CNN detectors including Faster-RCNN and SSD do not produce good detection performance over the KITTI driving benchmark dataset. In this chapter we proposed three methods on a multiple scale CNN network model for ADAS object detection. Firstly, CNN feature maps deconvolution and fusion was proposed to add context and deeper features for better object detection at lower scale of feature maps, to address the large object scale variation challenge. Then, soft non-maximal suppression (NMS) was applied across object proposals at different image scales to address the object occlusion challenge. As the cars, pedestrians and cyclists have distinct aspect ratio features, we measured their aspect ratio statistics and exploited them to set anchor boxes properly for better object matching and localization. The proposed CNN methods with various input image sizes were individually and jointly evaluated by extensive experiments over KITTI dataset. The effectiveness of the proposed methods was verified by experiment results with improved or comparable detection performance over KITTI test set. The average precision (AP) for pedestrian detection category "Easy" and the computation speed rank the first among the published works, the second for pedestrian category "Moderate" and "Hard", the third for car category "Moderate". And the network inference time for cars per 384×1280 image is only 0.08 second, much faster than the other top ranked published methods in KITTI leader board. In our future works we will investigate more CNN models and methods to improve object detection for safe and intelligent transport.



(a) Undetected car due to heavy truncation

(b) Undetected cars due to small size

(c) Undetected pedestrians due to occlusion and poor illumination conditions

(d) Motorcycle is falsely detected as bicycle.

Figure 3.8: Example images from KITTI testing set with false object detection by our method.

4 Recommendation System with Mobile Cloud Computing

4.1 Introduction

Besides driving safety and transport efficiency applications, CAV can support many entertainment applications at the same time. In this chapter we focus on the design of recommendation system, which plays a central role for many online entertainment applications and e-commercial services, such as multimedia streaming, recommendation of products such as movies, musics and articles [71, 76, 84]. Many big companies such as Amazon, eBay and Netflix have adopted recommendation techniques to their systems to estimate the potential preferences of customers and recommend relevant products or items to them. Recommendation performances have huge impact on the commercial success of these companies in terms of revenue and user satisfactory.

Online recommendation is a typical service for mobile cloud computing. The historical information of users and items are all stored in the cloud and recommendation process is also performed in the cloud. When a user's request is received, the user's information can be retrieved from the cloud and utilized by the recommendation system to recommend relevant items that the user may like. The large volume of raw data is not needed to be transported to the user side and only final results are sent to the user via wireless access network.

According to the type of data being collected and the ways of using them in recommendation systems, the approaches for recommendation can be classified as content-based (CB), collaborative filtering (CF) and hybrid one [74].

CB filtering is widely used for recommendation systems design, which utilizes the content of items to create features and attributes to match user profiles. Items are compared with items previously liked by the users and the best matched items are then recommended. One major issue of CB filtering approach is that recommendation system needs to learn user preferences for some types of items and apply these for other types of items.

CF approach is the most popular approach for recommendation systems design. It utilizes a large amount of data collected from user behavior in the past and predicts which items users will like. It does not need to analyze the content of the items. Instead, it relies on the relationships between users and items, which are typically encoded in a rating feedback matrix with each element representing a specific user rating on a specific item. An illustration of the CF based recommendation is shown in Fig. 4.1. The left of Fig. 4.1 shows a relationship graph of 3 users and 4 movies, which are connected by 5 edges. Each edge is associated with a rating of 1 to 5

stars, representing the level of user preference of the connected movie. The matrix in the right of Fig. 4.1 is generated according to the relationship graph. The general CF recommendation task is to predict the missing ratings (such as those represented by the symbol "?" in the matrix) by given users or for given items by data mining and exploring the user-item rating matrix.

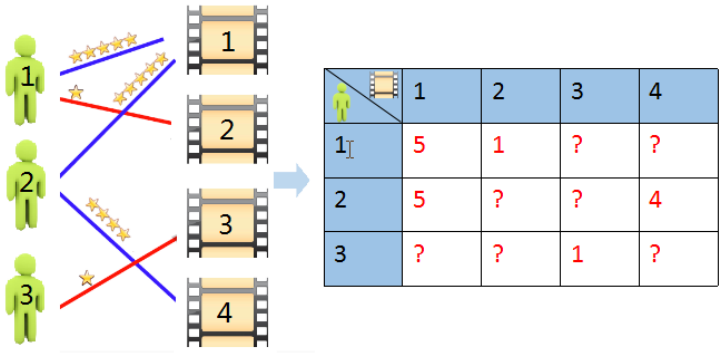


Figure 4.1: A simplified representation for movie CF recommender systems.

However it is widely known that CF approach suffers from sparsity and cold start (CS) problems. In the rating matrix only a small percentage of elements get values. Even the most popular items may have only a few ratings. For example, in a large Netflix rating dataset provided for Netflix Prize competition [69], there are about 100 million ratings given by over 480,000 users to about 18,000 movies. There is only around 1% of rating matrix elements receiving ratings. With a sparse rating matrix it is very challenging to estimate the relationships between items and users and make effective recommendation. Another well-known problem for CF approach is the CS problem, which can happen on new users or new items. CF approach requires a large number of ratings from a user or ratings on an item for an effective recommendation, which will not work for new users, new items or both due to few ratings available in the system. In addition, CS problem can be divided into CCS problem and ICS problem by whether number of rating records is zero or not. Generally, the sparsity of ratings for CS items is higher than 85% [85], and the sparsity of ratings for CCS items is 100%. Fig. 4.2 presents a simple illustration of the classification of CCS, ICS and non-CS items in recommendation systems.

The hybrid approach is one that combines CB filtering approach and CF approach attempting to overcome their shortcome and provide a more efficient result [68, 70, 73]. It is noted that the majority of the works on the CS recommendation problem are trying to provide recommendation of items that may be interesting to given users. Although a lot of works have been done with the hybrid approach to solve the sparsity and CS problems, recommendation of CS items is still an open research issue.

In this chapter we investigate the CS recommendation problem of providing prediction on the popularity of given CS items to the general users, and present a solution to predict the popularity of CCS items and ICS items. There are two main motivations for this work:

- CS items need to be recommended to get ratings for improved recommendation and they should be accurately recommended to give users better experiences with the recommen-

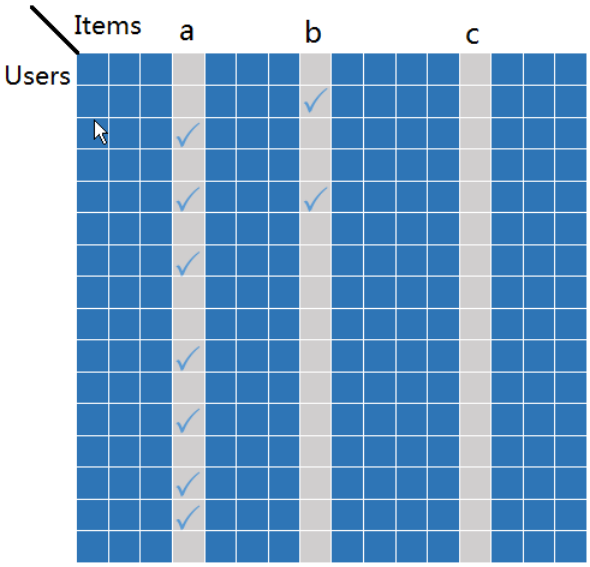


Figure 4.2: Illustration of non-CS item (a), ICS item (b) and CCS item (c), where $\surd$ indicates a known rating.

dation system. Otherwise the CS items may go to an undesirable cycle of receiving no ratings.

- The estimated ratings for CS items or items that are still under planning can give a measure of popularity of such items even before they are put into market (such as books, movies, etc), therefore help make right decisions on product planning and sale strategies. The accuracy of such estimation is critically important for this type of purposes.

We design two integrated recommendation models, in which item features are learned from a deep learning architecture SDAE [89] using the descriptions of items retrieved online, then these features are exploited and integrated into the timeSVD++ CF model [75]. timeSVD++ is one of the best performing CF models which tracks time changing behavior in the data and takes the temporal dynamics into account.

Our contributions are summarized as follows.

- We proposed a general framework of integrating the CF approach and machine learning algorithms to improve recommendation performance for CS items. In our proposed models, content features extracted from content descriptions (such as movie plots) by deep learning neural networks are used as the key item factor vectors in the recommendation model for CCS items and approximated by the item factor vectors in the model for ICS items. The content features are not only used loosely to determine item similarity as done in the existing hybrid approaches for CS items, but also become key component of the recommendation models, which affects both the training of the models and prediction of the unknown ratings for CS items.
- The framework of integrating the CF approach and machine learning algorithms for CS item recommendation is general. Various CF approaches and machine learning algo-

rithms can be used for general recommendation systems. The key integration point is on the extraction of item features by machine learning algorithms and embedding the item features into the CF recommendation models.

- Based on the general framework specific system design and models are presented, in which the state of the art CF model, timeSVD++ and an advanced deep learning neural network model, SADE, are used for CS items recommendation. Application of the models to Netflix movie recommendation with nearly 100 million ratings was investigated. The experiment results showed that tight coupling of the CF approach and content based approach for recommendation is feasible and very effective. For example, the rating prediction RMSE of the proposed model IRCD-CCS for CCS item recommendation is 0.045 lower than the second best performing approach, which represents a significant performance improvement in the research field of recommendation system design.

- In addition to the design and evaluation of recommendation models for CCS items and ICS items separately, we also compared the performance of IRCD-CCS model and IRCD-ICS model on rating prediction for ICS items. In practice, recommendation systems keep introducing new items into the systems over time. If a newly introduced item is a CCS item, the CF model can not provide rating prediction for it. If the item is an ICS item, the CF models may not give good recommendation. It may be beneficial to apply a CCS recommendation model for ICS item rating prediction. We proposed a scheme of switching recommendation models for ICS items and retraining the models to deal with the practical issues of transition of item status from CS to non-CS. To the best of our knowledge, this practical issue has not been studied before in the literature.

4.2 Related Works

Technically, matrix factorization (MF) method has been applied to CF by a variety of works. MF focuses on factorizing the rating matrix into low-dimension user latent vectors and item latent vectors. Training such a model can be effectively solved by using SGD [74] or alternating least squares (ALS) [94] to minimize the sum-squared distance. Authors [80] introduce the probabilistic matrix factorization (PMF) that scales linearly on large data sets and outperforms standard singular value decomposition (SVD) models. Based on PMF, several variants and generalizations are proposed like Bayesian PMF [81], generalized PMF [82].

As traditional CF algorithms only rely on the relations between users and items, which are typically encoded in a U-I matrix, the recommendation performance on sparsity problem and CS problem is largely limited. A large number of approaches incorporating additional information sources beyond U-I matrix have been developed to overcome the problems. Particularly auxiliary information of users or items and interaction related information are exploited to improve recommendation accuracy [84].

Auxiliary information refers to attributes about users and items. For user attributes, trust

network is incorporated into the raw ratings for prediction [88]. Authors [78] propose a probabilistic factor analysis framework which takes users' social trust relations into account. Authors [95] make use of users' social tags and design a diffusion-based recommendation algorithm which is only used in social tagging systems. Authors [77] adopt users' demographic data and apply a simple prediction rule by summing weighted ratings made by similar users to produce ratings for new users. Authors [87] combine attribute selection and local learning into the recommendation model for CS users. Both [77] and [87] use one of our baseline approaches (the ToU approach) in models to recommend products to CS users. Authors [96] try to learn user profiles through an additional interview process. For item attributes, collaborative topic regression (CTR) [90] applies topic model and latent Dirichlet allocation (LDA) to learn item content feature. However, this model only works on implicit rating prediction problem, and latent representation is not learned effectively with highly sparse content information.

With great successes in the fields of image, video and artificial intelligence, deep learning technology attracted large interests in the recommendation system field [72, 79, 83]. Collaborative deep learning (CDL) [91] is a representative example that applies deep learning to recommendation systems by integrating stacked denoising autoencoder (SDAE) into a simple latent factor based CF model for movie and article recommendation. Nevertheless, CDL only focuses on the situation of rare users and implicit interactions between users and items, and very simple CF model is considered. The main objective of CDL is recommending top-N items, not for the explicit ratings prediction.

On the other hand, the interaction-associated information refers to the information associated with U-I interaction behavior, like timestamps and locations of the ratings being made. Recently there is a strong interest in the utilization of time information for CF, which demonstrates superior recommendation performance [75, 92, 97]. TimeSVD++ [75] is a model that simulates the temporal dynamics of user interests by changing static biases and latent factors into time-dependent ones. Authors [92] introduce a set of additional time feature vector and use tensor factorization to learn the features. A different modeling scheme on user preferences is presented in [97], where a latent transition matrix is used to summarize the evolving preferences for each user. Authors [93] propose a time-dependent method to compute the similarity among different users. But they are not directly applicable to CS problem.

Generally, CS problem can be classified to CS user problem and CS item problem according to the completely missing ratings for the users or the items. For CS user problem, as system information like locations and gender does not describe user interest efficiently, several recent studies attempt to enrich user profiles with information from other channels, such as social trust network [78, 88], tagging system [95], interview process [96]. But these kinds of information are hard to collect in normal conditions. In addition, it is more difficult to acquire personal information of new users because of privacy issues. By the contrast the focus of this chapter is on the CS item problem with the motivations described previously. In order to alleviate the information scarcity of CS items, most research efforts so far have been devoted to profiling new items with additional information (e.g., collecting item attributes). However, there still exist a

number of limitations in the existing research works. Firstly, it is hard to dig out the specific features of new items with limited rough attributes. And gathering fine-grained attributes like tags, keywords and categories are always time-consuming and costly. Secondly, most studies that combine item content information with ratings data [86, 90, 91] adopt generative probabilistic models and tend to overfit easily on CS item situations. The last problem with these works is that they do not take time information into account. In this chapter a solution integrating deep learning and CF approach is proposed to address these limitations and largely improve recommendation performance for CS items.

4.3 Proposed Recommendation Model

In this section, we propose two integrated recommendation models with CF and deep learning , called IRCD-CCS and IRCD-ICS for CCS items and ICS items, respectively. A recommendation system is assumed with U users and V non-CS items. In addition it is assumed there are J CCS items, which receive no ratings from the users until the time of investigation, and I ICS items, which receive only a few ratings from the users. We let rating $r_{ui}(t)$ denote the rate by user u on item i at time t. The recommendation task considered in this chapter is to estimate the unknown ratings for both CCS and ICS items based on the known ones. We let $\hat{r}_{ui}(t)$ denote the predicted values of $r_{ui}(t)$.

4.3.1 Deep Learning of Content Features

As traditional CF models are not able to estimate the ratings for CS items, additional content descriptions for the items are obtained for the proposed model. Item features are extracted from the content descriptions and used with a CF model for CS item rating estimation.

Firstly the raw content information of all items are processed to generate vectors based on the bag of words approach. These item associated vectors are then learned by a SDAE to obtain item content features, which are then used in the CF models. SDAE is a deep network that is stacked by multiple denoising autoencoders (DAEs). Each layer of SDAE is trained as a DAE by minimizing the error in reconstructing its input (which is the output of the previous layer). Usually we consider the first half layers of the network as an encoding part and the last half layers as a decoding part. Encoding part tries to learn the feature representations of the noise-corrupted input, and decoding part tries to reconstruct the clean input itself in the output. An example structure of SDAE is shown in Fig. 4.3.

Formally, given a set C of vectors as raw content information of all items, an L-layer SDAE solves the following optimization problem:

$$\min_{W_l, b_l} \|C - C_L\|^2 + \lambda \sum_l \|W_l\|^2, \tag{4.1}$$

where C_L denotes the output of layer L of the network and W_l and b_l denote the weight matrix and bias vector of layer l of the network. More details on the SDAE structure and training are

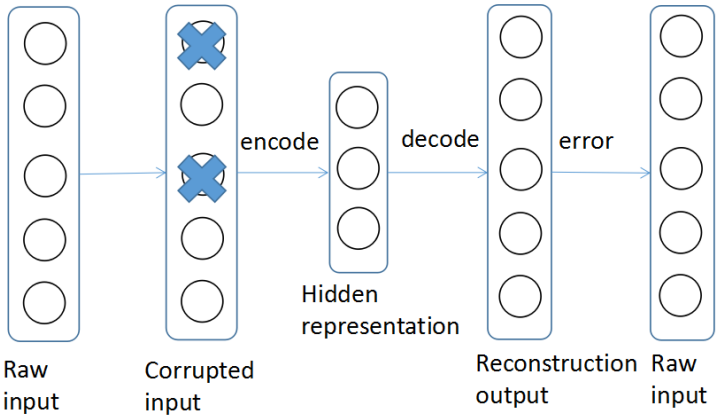


Figure 4.3: A graphic structure of SDAE.

referred to [91]. Once the model is trained, the item content features could be obtained from the hidden layer $C_{L/2}$ of the network. For a given item i, the feature representation, denoted by θ_i, is a vector with low dimensions.

It is noted that apart from the goal of learning the features from the rating records, another goal of using SDAE is to reduce the dimensionality of the item content-based vectors to be same with latent factor vectors, which can then be fused into the CF process.

4.3.2 timeSVD++ Model

The CF model used in the proposed IRCD is timeSVD++. There are several variants of timeSVD++ model. In this chapter the latent factor based variant is considered. For a latent factor based model, a rating by a user u on an item i is computed by the inner product of a vector q_i (the item factor for item i) and a vector p_u (the user factor for user u) using the following formula:

$$\hat{r}_{ui}(t) = q_i^T p_u. \tag{4.2}$$

In order to take the biases, additional implicit feedback and temporal effects into account, timeSVD++ uses a revised prediction rule by adding some baseline predictors to (4.2) as follows:

$$\hat{r}_{ui}(t) = \mu + b_i(t) + b_u(t) + {q_i}^T \left[p_u(t) + |N(u)|^{\frac{1}{2}} \sum\nolimits_{j \in N(u)} y_j \right]. \tag{4.3}$$

Here, μ denotes the overall mean rating, $b_i(t)$ and $b_u(t)$ indicate the time-aware biases of item i and user u respectively. Item factors do not change with time as they are more static in nature than humans. The set $N(u)$ contains the items rated by user u. The factor $|N(u)|^{\frac{1}{2}} \sum_{j \in N(u)} y_j$ indicates the perspective of implicit feedback, where y_j is a vector for item j related to implicit feedback and is to be learned from training process.

The biases $b_i(t)$ and $b_u(t)$ for items and users, respectively, are computed by the following formulae:

$$b_i(t) = b_i + b_{i,Bin(t)}, \tag{4.4}$$

$$b_u(t) = b_u + \alpha_u \cdot dev_u(t) + b_{u,t}. \tag{4.5}$$

It is noted that the time-aware item bias $b_i(t)$ is composed of a stationary part b_i and a time changing part $b_{i,Bin(t)}$, where the whole timeline is split into time-based bins $Bin(t)$. For user bias $b_u(t)$, b_u represents the stationary part, $\alpha_u \cdot dev_u(t)$ captures a possible gradual drift, in which the time deviation $dev_u(t)$ is defined as:

$$dev_u(t) = sign(t - t_u) \cdot |t - t_u|^{\beta}, \tag{4.6}$$

and $b_{u,t}$ denotes the day-specific sudden drift. Similar to user biases, user factors also become time-aware as $p_u(t)$.

$$p_u(t) = (p_{u1}(t), p_{u2}(t), ..., p_{ud}(t)). \tag{4.7}$$

$$p_{uk}(t) = p_{uk} + \alpha_{uk} \cdot dev_u(t) + p_{uk,t} \quad k = 1, ..., d. \tag{4.8}$$

Here d is the dimensionality of user factors.

In order to learn the model parameters, the system minimizes the regularized squared error on the training ratings:

$$\begin{aligned} \min_{q*,p*,b*} \sum_{u,i,t} (r_{ui} - \hat{r}_{ui})^2 + \lambda \Big[\|q_i\|^2 + \|p_u\|^2 + \|\alpha_u\|^2 + \|p_{ut}\|^2 \\ + \sum\nolimits_{j \in N(u)} \|y_j\|^2 + {b_i}^2 + {b_{i,Bin(t)}}^2 + {b_u}^2 + {\alpha_u}^2 + {b_{u,t}}^2 \Big]. \end{aligned} \tag{4.9}$$

According to the above regularized squared error function, a SGD optimization method is used to iteratively learn the model parameters. The algorithm loops through all ratings in the training set iteratively and updates each parameter according to the associated gradient until system converges.

4.3.3 Rating Prediction Model for CCS Items

Next we present the the IRCD-CCS model for rating prediction of CCS items. We first present the computation of content similarity. Two baseline rating prediction approaches based on content similarity are then presented. Finally the IRCD-CCS model for CCS items is presented, which integrates content similarities based approach and timeSVD++ model.

To predict the ratings for CCS items, we first use similarity measure to relate CCS items to the non-CS items, and predict the ratings for the CCS items from their most related non-CS items. Based on the item features obtained from the SDAE deep learning process, we use Pearson's correlation coefficient formula to compute the similarity between CCS items and non-CS items. For any two feature vectors θ_i and θ_j of items i and j, the similarity is computed as below:

$$s_{ij} = \frac{\sum_{k=1}^{d} \left(\theta_{ik} - \bar{\theta}_i\right) \cdot \left(\theta_{jk} - \bar{\theta}_j\right)}{\sqrt{\sum_{k=1}^{d} \left(\theta_{ik} - \bar{\theta}_i\right)^2 \cdot \sum_{k=1}^{d} \left(\theta_{jk} - \bar{\theta}_j\right)^2}}. \tag{4.10}$$

where $\bar{\theta}_i$ and $\bar{\theta}_j$ are the mean values of vectors θ_i and θ_j.

We consider two baseline approaches to predict ratings for CCS items. The first approach predicts the ratings for CCS items from their M most similar non-CS items within the whole non-CS item set after the missing ratings for the non-CS items are predicted, which is called Top-of-All (ToA) approach. Let $\mathcal{S}^M(j)$ denote the set of the M most similar non-CS items to a CCS item j, $j \in [1, J]$. For the ToA approach, the following formula is used to predict rating by user u on CCS item j:

$$\hat{r}_{uj} = \frac{\sum\limits_{i \in \mathcal{S}^M(j)} \hat{r}_{ui} s_{ij}}{\sum\limits_{i \in \mathcal{S}^M(j)} s_{ij}}. \tag{4.11}$$

It is noted that the ratings $\hat{r}_{ui}$ can be both real and predicted ones.

Alternatively, we can use the second approach to predict the ratings by a user for CCS items from their M most similar non-CS items within the set of non-CS items rated by the user, which is called Top-of-User (ToU) approach. Let $\mathcal{S}^M(u, j)$ denote the set of the M most similar non-CS items among the non-CS items rated by u to a CCS item j, $j \in [1, J]$. For the ToU approach, the following formula is used to predict rating by user u on CCS item j:

$$\hat{r}_{uj} = \frac{\sum\limits_{i \in \mathcal{S}^M(u,j)} r_{ui} s_{ij}}{\sum\limits_{i \in \mathcal{S}^M(u,j)} s_{ij}}. \tag{4.12}$$

Note that in this case, the ratings r_{ui} are real ratings in the training set.

The above two simple and straightforward approaches make rating prediction entirely based on similar non-CS items and ignore the other information in the rating matrix. According to our experiments, the ToU approach has higher accuracy than the ToA approach. Next we propose an integrated model by combining the ToU approach and timeSVD++ together.

As conventional CF methods are not directly applicable to the CCS problems, we set the item factor q_i in timeSVD++ model to item content feature θ_i and replace the overall average rating μ and item biases $b_i(t)$ with predicted ratings, which is generated by the ToU approach. We extend the ToU approach to generate predicted ratings for all the real ratings in the training set.

The following prediction rule is used in the proposed IRCD-CCS model:

$$\hat{r}_{ui}(t) = b_u(t) + {\theta_i}^T \left[p_u(t) + |N(u)|^{\frac{1}{2}} \sum\nolimits_{j \in R(u)} y_j \right] + \frac{\sum\limits_{j \in \mathcal{S}^M(u,i)} r_{uj} s_{ij}}{\sum\limits_{j \in \mathcal{S}^M(u,i)} s_{ij}}. \tag{4.13}$$

and the corresponding regularized squared error function is:

$$\min_{p*,b*} \sum_{u,i,t} (r_{ui} - \hat{r}_{ui})^2 + \lambda \left[\|p_u\|^2 + \|\alpha_u\|^2 + \|p_{ut}\|^2 + \sum\nolimits_{j \in N(u)} \|y_j\|^2 + {b_u}^2 + {\alpha_u}^2 + {b_{u,t}}^2 \right]. \tag{4.14}$$

Because all the parameters are user-associated, it can be used to predict ratings for CCS items after training.

It is noted that in the IRCD-CCS model for CCS items, CCS items did not receive any rating from users. Therefore the CCS items do not participate in the timeSVD++ model training. However, to enable prediction with the rule (4.13), we train the IRCD-CCS model with the rating matrix by setting the item factors q_i of non-CCS items to their content feature θ_i. It means that the content features learned from the SDAE is utilized instead of the item features hidden in the rating matrix in the IRCD-CCS model for the CCS items.

Fig. 3.4 shows a graphical framework of traditional MF model (left part) and IRCD-CCS model (right part). For each user u and item i, traditional MF model computes the predicted rating r_{ui} by adding the latent factors with biases predictor. Latent factor is the inner product of item factor q_i and user factor p_u. Biases predictor include the overall mean rating μ, item bias b_i and user bias b_u. Compared with traditional MF, the IRCD-CCS model first applies SDAE to learn the item content feature θ_i from the raw item content information C. Then the ToU approach is used to obtain a preliminary predicted rating r'_{ui} based on the similarity measure of item content feature. In the model training item factor q_i is set to item content feature θ_i. Finally ratings can be predicted by adding the two parts together with the rule shown in (4.13).

It is noted that the IRCD-CCS model needs pre-processing to generate the rating prediction for every real ratings in the training set offline, but the online prediction can be computed immediately with the prediction rule.

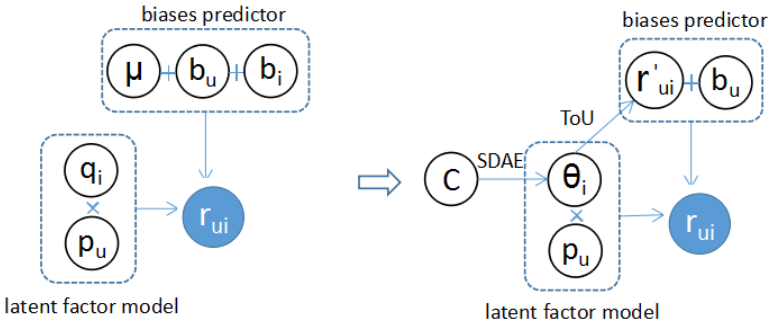


Figure 4.4: The graphical modification of IRCD-CCS framework.

4.3.4 Rating Prediction Model for ICS Items

The IRCD-ICS model modifies the timeSVD++ by applying content features learned from SDAE into item latent factor training process. In our proposed model IRCD-ICS for ICS items, the prediction rule (4.3) is reused, but the model parameters are learned by minimizing a different

regularized squared error function. The modified regularized squared error function is shown below:

$$\min_{q*,p*,b*} \sum_{u,i,t} (r_{ui} - \hat{r}_{ui})^2 + \lambda \Big[\|q_i - \theta_i\|^2 + \|p_u\|^2 + \|\alpha_u\|^2 + \|p_{ut}\|^2 + \sum\nolimits_{j \in N(u)} \|y_j\|^2 + {b_i}^2 + {b_{i,Bin(t)}}^2 + {b_u}^2 + {\alpha_u}^2 + {b_{u,t}}^2 \Big]. \tag{4.15}$$

Note that the dimensionality of both q_i and θ_i is d.

It is noted that the trained IRCD-ICS model is used for both ICS items and the general non-CS (NCS) items (with relatively larger number of ratings). We do not create separate recommendation models for the ICS items and the general NCS items. For the ICS items, it is desirable to learn the item factor from the content description as much as possible, as there is very little useful information to be learned for the ICS items from the rating matrix. Therefore the content features play a key role in the recommendation for ICS items. For the NCS items, both rating matrix and content description (content features) can be utilized and should be used to improve the rating prediction performance. The introduction of factor $\|q_i - \theta_i\|^2$ can help achieve the goal of learning item factors for both ICS and NCS items.

Compared with timeSVD++, the following main changes are made on the update rule of item factor vectors. For each given rating $r_{ui}(t)$, the prediction error is computed by $e_{ui}(t) = r_{ui}(t) - \hat{r}_{ui}(t)$. The original update equation of item factor q_i used in timeSVD++ is:

$$q_i \leftarrow q_i + \gamma \left\{ e_{ui}(t) \left[p_u(t) + |N(u)|^{-\frac{1}{2}} \sum\nolimits_{j \in N(u)} y_j \right] - \lambda \cdot q_i \right\}. \tag{4.16}$$

In the proposed model the following update rule for the item factor q_i is used:

$$q_i \leftarrow q_i + \gamma \left\{ e_{ui}(t) \left[p_u(t) + |N(u)|^{-\frac{1}{2}} \sum\nolimits_{j \in N(u)} y_j \right] - \lambda \cdot (q_i - \theta_i) \right\}, \tag{4.17}$$

where γ denotes the learning rate. The interested readers are referred to [75] for more details on the timeSVD++ model learning process. Fig. 4.5 shows a graphical framework for traditional MF model (left part) and IRCD-ICS model (right part). For the IRCD-ICS model the item content feature θ_i is utilized to learn item factor q_i in the training process according to rule (4.17).

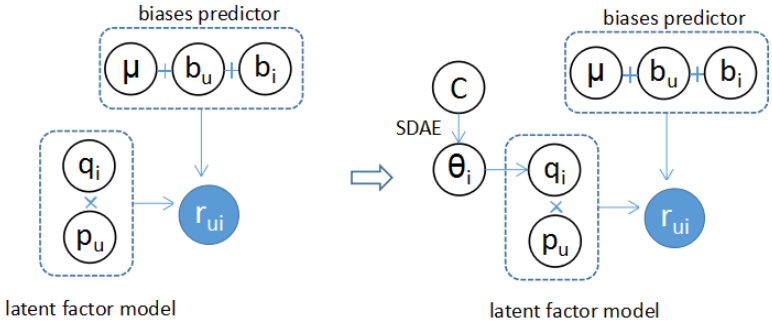


Figure 4.5: The graphical modification of IRCD-ICS framework.

4.4 Performance Evaluation

In this section, the recommendation performance of the proposed models IRCD-CCS for CCS items and IRCD-ICS for ICS items is evaluated. The experiment data preparation and results are presented and discussed.

4.4.1 Experiment Dataset and Settings

A large real-world dataset created by the Netflix Prize is used to evaluate the proposed models. The Netflix dataset contains more than 100 million explicit ratings on a scale of 1 to 5 stars for 17770 movies defined by 480189 anonymous users. The corresponding timespan ranges from Dec 12, 1999 to Dec 12, 2005.

In order to predict ratings for CS items we also collect the plots of the movies from IMDB to extract item content information. We first collected the corresponding movie plots by OMDb API[1], which is a free web service to obtain movie information. A Python-based program was written, which traverses the movies in the Netflix dataset and automatically sends search requests of plots according to the movie titles to the OMDb database. Then the collected movie plots were filtered by removing stop words, which refer to the most common words in a language offering little useful information. In the experiments the list of stop words was obtained with a built in Python package. Then we computed the term frequency-inverse document frequency (tf-idf) of each word in the corpus. Tf-idf is the product of term frequency and inverse document frequency. Term frequency refers to the number of occurrence of words in a document, and inverse document frequency is calculated by dividing the total number of documents by the number of documents containing the word. Generally tf-idf is used as a weighting factor to reflect how important a word is to a document. Based on tf-idf, the most important S words with the highest tf-idf values are chosen to form a dictionary. S is set to 20000 in our experiments. Each movie plot is then represented by a S-dimensional bag-of-words vector, in which each entry of the vector indicates the count of corresponding word occurs in the plot. The last step is the normalization for all the vectors. After the movies with missing plots were removed, the final dataset has 476691 users, 14657 movies, and 95975845 ratings.

A workflow of obtaining and processing the movie plots is presented in Fig. 4.6.

As we are interested in the performance of rating prediction for both CCS items and ICS items, the original dataset is partitioned into one training set and one test set for CCS items and ICS items, respectively. The movies are ordered by the timestamp of their first received rating. The histogram of movies on the date of their first rating is shown in Fig. 4.7. We divide the entire timespan of the dataset (2240 days) into 100 intervals, and count how many movies fall into each interval. Generally, new movies appeared late, and thus located at far right in the figure. For the preparation of training and test set for CCS experiments, we choose the most recent L movies for test set and the other movies for training set. The ratings of the L CCS movies are used for testing and it is guaranteed that none of these CCS movies could be seen

[1] http://www.omdbapi.com

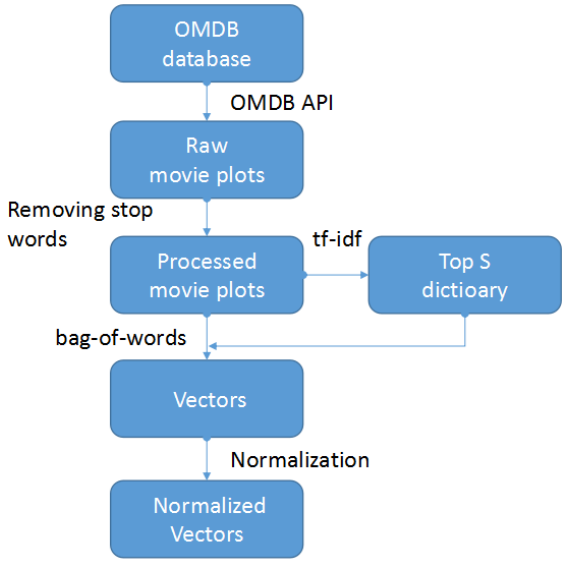


Figure 4.6: Workflow of data preprocessing on movie plots.

by any user in the training set. To prepare the ICS training and test sets, we choose the most recent K movies for test set and the other movies for training set. Meanwhile the earliest N ratings of each tested movie are added into training set and all the remaining ratings of these K ICS movies are used for test purpose. It is noted that the ratings in the training set which have timestamp later than the timestamp of the earliest rating in the test set are all discarded.

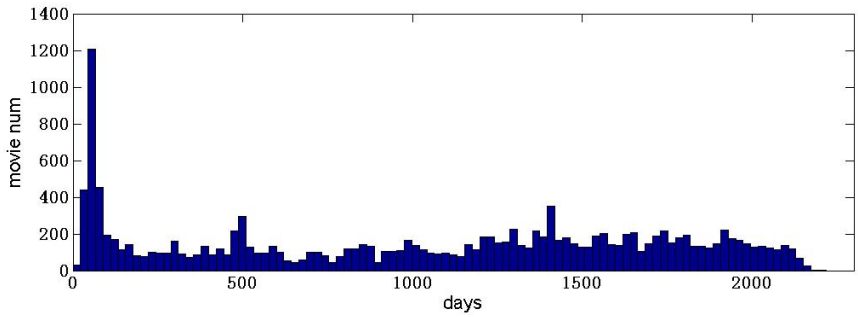


Figure 4.7: Histogram of movies on the date of their first ratings.

For the experiment setting of some parameters, we chose some typical values for performance evaluation. The proposed recommendation models can work with general configurations of the parameters. As there is a large space of values for the parameters, it is very time consuming to test all the combinations of the settings for the parameters. After quick check on the recommendation performance with a number of parameters settings, we determined the settings for model parameters (e.g., setting dimensionality d of feature vectors to 50) with those giving good performance.

For the experiment datasets, the rating statistics for the training and test sets are presented in Table 4.1 and Table 4.2 for CCS experiment and ICS experiment, respectively.

We first conduct experiments for CCS movies and compare our model IRCD-CCS against three recommendation models: ToA model, ToU model, and simple average (SA) model. ToA

Table 4.1: Statistics of the training and test datasets for CCS movie experiment.

	L=100		L=300	
	Training set	Test set	Training set	Test set
Number of users	476691	11764	476691	51171
Number of movies	14557	100	14357	300
Number of ratings	95959733	16112	95874146	101699
Mean rates	3.6042	3.4975	3.6043	3.5652

Table 4.2: Statistics of the training and test datasets for ICS movie experiment(N=5).

	K=100		K=200		K=300	
	Training set	Test set	Training set	Test set	Training set	Test set
Number of users	476691	11603	476691	31725	476691	50918
Number of movies	14657	100	14657	200	14657	300
Number of ratings	95960233	15612	95924771	51074	95875646	100199
Mean rates	3.6042	3.5135	3.6043	3.4288	3.6043	3.5718

model and ToU model are presented in Sec. 4.3.3. As the existing models for CCS item prediction use different content information and features, they are not directly comparable to the proposed model. Therefore the simple prediction model SA for CCS items is used: every CCS item rating by a user is set to the average of ratings of the user.

On the other hand, the baseline models for ICS movie experiment include ALS, SGD, timeSVD++ and CDL. ALS and SGD are two major algorithms for learning parameters. In our experiment, we use them to minimize the error function of (4.2). As described in the previous section, timeSVD++ is a time-aware CF model, taking the temporal effects into consideration. CDL is a model jointly performing deep representation learning and CF, which applies an ALS-style algorithm for learning. ALS and SGD provide recommendation based on plain ratings, while timeSVD++ and CDL utilize additional time information and item content information respectively.

The overall experiment procedure has four major steps, which are described below:

- Configure the system parameters: the parameters to be configured include learning rate, regularization and factor dimension for each model as described in Sec.4.4.1. These settings remain the same throughout the experiments.

- Prepare training and test sets: as described in Sec.4.4.1, training and test sets are generated respectively according to the configurations with different test sizes (L for CCS experiment and K for ICS experiment). Moreover, the number of training samples for tested ICS items N is also configured to different values in ICS experiments.

- Train the model by fitting the data in training set: specifically, ALS and CDL apply the ALS algorithm to update model parameters, while IRCD models, SGD and timeSVD++ learn model parameters by SGD algorithm. ToA and ToU approaches compute the prediction result directly using Eq. (4.11),(4.12).

- Predict the ratings in the test set: this is done by using the trained models. The prediction performance is then computed and recorded.

4.4.2 Results and Analysis

In this subsection, the proposed models are evaluated in terms of RMSE and compared with other baseline recommendation models. RMSE is an objective metric widely used for performance evaluation of recommendation system models, which is defined as:

$$RMSE = \sqrt{\frac{1}{N_p} \sum_{u,i} (\hat{r}_{ui} - r_{ui})^2}, \tag{4.18}$$

where N_p denotes the total number of predictions.

After tuning the parameters on the validation set, we compare our proposed IRCD models with other baseline models. Dimensionality d of the feature vectors is set to 50 for all the models.

4.4.2.1 Performance Evaluation of of IRCD-CCS Model on CCS Movies

Firstly we evaluate the prediction performance of IRCD-CCS model for CCS items. Table 4.3 presents the prediction result RMSE, against the number of most related items M, the size of the test dataset for CCS movies, L for CCS new movies. The SA model, ToA model, ToU model and IRCD-CCS model are compared. The number M of most related items is configured to 20 and 100, and the size $L$of the test dataset for CCS movies is configured to 100 and 300, respectively.

It is noted that the IRCD-CCS model performs the best for all the investigated scenarios. Its performance is significantly better than the baseline models. The result shows an improvement of about 0.05 on RMSE compared to the second best model ToU. The performance of both IRCD-CCS model and ToU model improves largely as M increases. On the contrary, the ToA model works well with small M (e.g., with only 20 most related ICS items for rating prediction), but with a large M such as 100 it has a poor performance, which is even worse than the SA model. This can be explained by that the influence of prediction error is accumulated as M increases.

Table 4.3: Performance comparison of prediction models for CCS movies with Netflix dataset.

Approaches	ToA		ToU		IRCD-CCS		SA
	M=20	M=100	M=20	M=100	M=20	M=100	
RMSE (L=100)	1.155	1.224	1.133	1.113	1.075	**1.053**	1.146
RMSE (L=300)	1.134	1.218	1.140	1.127	1.096	**1.082**	1.157

4.4.2.2 Performance Evaluation of IRCD-ICS Model on ICS Movies

Next we compare the prediction performance RMSE of the IRCD-ICS model with the existing models for ICS movies. Table 4.4 presents the experiment results. In the experiments the number of associated training ratings N is set to 5. The size K of the tested ICS movies is set to 100, 200 and 300. It is observed that the proposed method IRCD-ICS achieves the best accuracy in all cases. CDL performs the worst even though it applies SDAE for the content information learning. The main reason is that CDL is proposed on the use of implicit rating data instead

of explicit data. The large gap (around 0.05) between SGD and ALS demonstrates that SGD is effective on making better predictions for ICS items than ALS. By modeling the temporal dynamics, timeSVD++ outperforms SGD by a significant margin of 0.002 to 0.017. Compared to timeSVD++, IRCD-ICS model shows further consistent improvement of more than 0.004 with inclusion of content information and deep learning process.

Table 4.4: Performance comparison of prediction models for ICS movies with Netflix dataset.

Algorithms	RMSE (K=100)	RMSE (K=200)	RMSE (K=300)
ALS	1.124	1.112	1.097
SGD	1.070	1.076	1.058
timeSVD++	1.053	1.074	1.053
CDL	1.179	1.151	1.148
IRCD-ICS	**1.049**	**1.070**	**1.048**

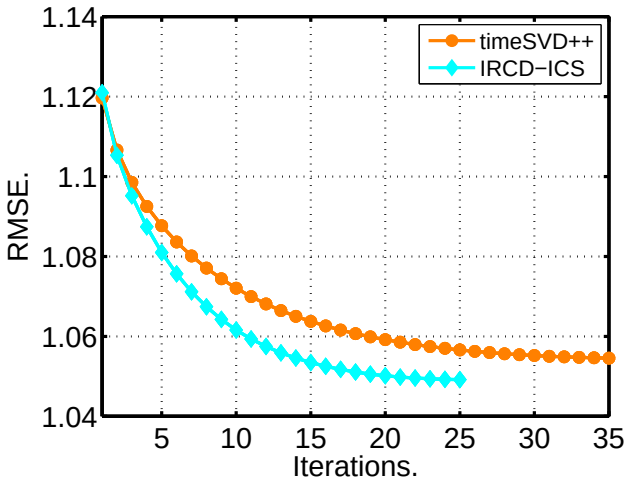
To have a deep comparison of timeSVD++ and the proposed model IRCD-ICS, we investigate how the prediction performance RMSE changes with the training iterations. Representative training curves are presented in Fig. 4.8. For both models the RMSE decreases monotonically without overfitting problem. The IRCD-ICS model converges faster than timeSVD++ in the training process. For the case of K being 100, the IRCD-ICS model takes only 25 iterations to end while timeSVD++ needs more than 35 iterations, which means more computation time.

4.4.2.3 Performance Evaluation of Both Models on ICS Movies

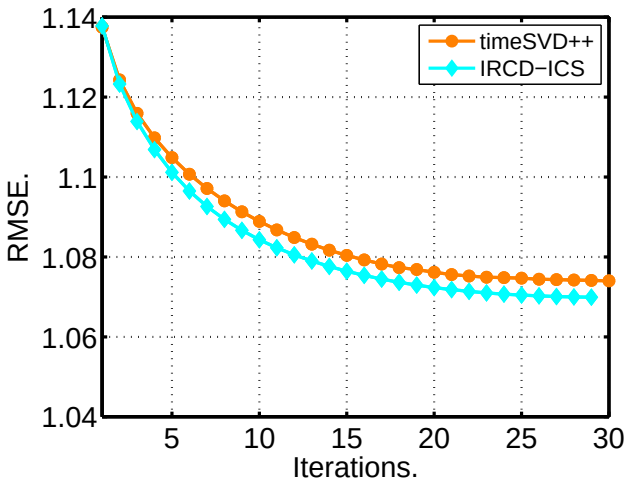
In general if an item has a sufficient number of ratings the IRCD-ICS model for this item will certainly outperform the IRCD-CCS model. But when the number of ratings for an ICS items is close to zero, it is not clear whether IRCD-ICS model or IRCD-CCS model performs better. Therefore in this subsection we compare the two models IRCD-CCS and IRCD-ICS on recommending ICS items. To evaluate and compare the models under different degrees of rating matrix sparsity, the number of associated training ratings N is set to 1, 3, 5 and 7 for ICS movies. Fig. 4.9 shows the RMSE results of timeSVD++ and the IRCD-ICS model with different N for the ICS movies. The red dashed line indicates the RMSE of IRCD-CCS model, which is irrelevant to N. As we can see from Fig. 4.9 , the RMSE value of both timeSVD++ and IRCD-ICS decreases as N increases. However, when N is less than 5, the ICS item based models including timeSVD++ and IRCD-ICS even perform worse than IRCD-CCS model. It is shown that ICS item based models do not make good prediction of ratings for items with only a small number of ratings, in which case the CCS item based model is preferred.

4.4.3 Discussions

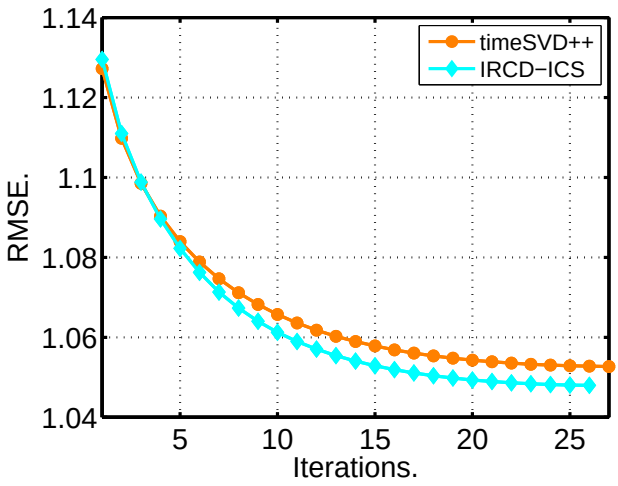
In the above experiments, different recommendation models are used for CCS items and non-CCS (ICS and NCS) items. However, in practical operation of recommendation systems, if ratings for a CCS item are received from users, the item becomes an ICS item. In this case, for this new ICS item rating prediction with the model trained and used for CCS items may be worse



(a) K=100.



(b) K=200.



(c) K=300.

Figure 4.8: Training curves of timeSVD++ and IRCD-ICS model.

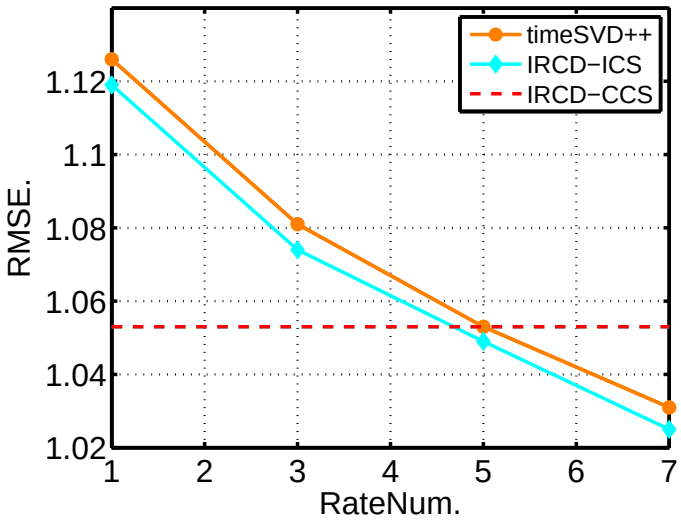


Figure 4.9: Performance comparison of IRCD-CCS and IRCD-ICS models for rating prediction of ICS models, K=100.

than with a model used for ICS items. However, the existing recommendation model for non-CCS items does not use any information (ratings and content description) related to this ICS item, which just changed its status from CCS item. Therefore there is a need to retrain the recommendation model for non-CCS items with extra rating and content description information from the new non-CCS items.

There are two key issues to consider with regards to the retraining of the model for non-CCS items: 1) how frequently the model should be retrained ? 2) which part of the model should be retrained ?

For the issue of retraining frequency, we do not need immediately retrain the ICS model whenever some items change their status from CCS to ICS. As the recommendation of items is not made continuously, there is no need to retrain the ICS model too frequently, which will incur a very high computation cost. Even if the IRCD-ICS model is not retrained when a CCS item changes to ICS item, the CCS model can be used temporally for recommendation of that item. According to the evaluation results, there is not large performance degradation when the CCS model is applied to ICS items. Therefore we can design a recommendation system, which collects and stores the new ratings made by the users, and regularly retrains the ICS and CCS recommendation models, for example, in the scale of days or weeks according to the rating activities.

For the issues of which parts of the models to be trained, there is very little impact in the trained model with inclusion of new ratings to the training set, as they take very small proportion of the total ratings in the training set. To reduce computation loads, in the model retraining, we can keep the values for trained model parameters (such as q_i, p_u and b_i) which are well trained in the previous round of training process and should be stable in short term, and only learn the parameter values for the new items.

4.5 Conclusion

Recommendation of cold start items is challenging and still an open research issue for recommendation systems. Cold start items can be classified to complete cold start (CCS) items which receive no ratings and incomplete cold start (ICS) items which receive more than zero but very few ratings. In this chapter we proposed two recommendation models to address the recommendation problems for CCS and ICS items, respectively. The models combine a time-aware collaborative filtering (CF) model timeSVD++ with a deep learning architecture SDAE. The deep learning neural network SDAE is responsible for the extraction of item content features, while the timeSVD++ model is responsible for prediction of unknown ratings. It considered temporal dynamics of user preferences and item features. A large number of experiments were run to evaluate the proposed models in terms of recommendation prediction error RMSE on Netflix dataset. The results showed that our models outperformed existing baseline approaches for cold start item recommendation. From our analysis and experiments, the impact of including the time and item content information is very large. Especially for CCS problem, our model can successfully takes the advantage of CF latent factor models to gain significant performance improvement. In addition, we also compared our proposed models on the ICS new items recommendation with different degrees of rating matrix sparsity. It was found out that the ICS item-based model does not make good recommendations for items that received very few ratings (e.g. 3 ratings). In that case the CCS item based model should be used instead of the ICS item based model.

In the future we plan to extend our recommendation models for cold start items and work on the following research directions. First, we are interested in the investigation of the recommendation performance for CCS and ICS items with more system configurations and parameters setting, in order to reveal more insights to their impact on recommendation performance and system optimization. Second, we create and maintain two separate recommendation models for CCS items and ICS items, respectively. This approach requires extra storage and computation resources. We plan to design a recommendation model, which is applicable to recommendation of both CCS and ICS items. Third, recommendation models are evaluated by the RMSE of rating predictions, which may not effectively reflect the performance of real recommendation systems. We are interested in the design of an additional performance evaluation approach, which can take item recommendation decisions into account and quantify the impact of the decisions on user acceptance of recommended items. Finally, in this chapter we run experiments of cold start item recommendation on Netflix movies. We are interested in the application of the models to the recommendation of other products such as online music.

5 Conclusion and Future Work

Increasing urbanization and the number of vehicles on road is creating enormous pressure on modern transport systems and exacerbating many global issues such as traffic congestion, accidents and pollution. Connected autonomous vehicle (CAV) is an ambitious technology to tackle the above problems. It is proposed to capitalize on the latest technology advances of autonomous driving and vehicles networking, to provide a wide range of driving safety applications, transport efficiency applications and entertainment applications. As computing is an extreme component for CAV systems, various mobile computing models are proposed in the literature. However it is noted that none of mobile computing models could fit all the CAV applications, which have highly diverse QoS requirements such as communication delay, data rate, accuracy, reliability and/or computing latency. In this thesis, a hybrid mobile computing model is investigated for CAV, in which three mobile computing models (namely mobile local computing, mobile edge computing and mobile cloud computing) and/or their combinations are chosen and applied to different CAV applications.

5.1 Conclusion and Summary

In this section, the research activities in this thesis are briefly summarized. In chapter 1, we start by discussing the technologies of autonomous driving and connected vehicles respectively. As both technologies have their own limitations and could complement each other, the integration of two technologies (CAV) is proposed. While CAV holds huge potential in a wide range of applications including driving safety, transport efficiency and user entertainment, the research of CAV is still on an early stage and there are many technical and non-technical challenges unsolved. Specifically we focus on the mobile computing challenges in this thesis, which is the cornerstone of CAV systems. A survey of relevant mobile computing models including mobile cloud computing, mobile edge computing and mobile local computing is presented. According to the survey, it can be found that there is no individual model which can support all the CAV applications. Therefore the main objective of this thesis is determined: proposing a hybrid mobile computing model (HMCM) to deliver the CAV applications. In the HMCM, the computing and communication resources at the local host and external entities are expected to be efficiently utilized according to the CAV applications requirements. Finally the thesis outline and the research contributions of this thesis are summarized.

Chapter 2 first presents the architecture of the HMCM, which is consisted of interconnected ad-hoc fogs (A-Fogs) with consumer computing devices, dedicated fogs (D-Fogs) with service

operator devices at edge and remote clouds. Detailed functional modules of fog nodes are designed. Then a framework for QoS aware service and resource management is designed to provide QoS support for large scale data analytics services for CAV applications. To support the QoS aware resource management framework, extensive benchmark experiments over fogs with distributed computing engine Spark are run to measure computing performance of various analytics tasks and create easy to use workload models for QoS management framework. We formulate an optimization problem for analytics job admission control and resource allocation (ACRA). Resource heterogeneity, networking, power and computing resource cost models are taken into account in ACRA optimization objective. Distributed ACRA algorithms are proposed including two baseline non-cooperative algorithms and a matching theory based cooperative ACRA algorithm. A system-level simulator is developed to evaluate the analytics services and the proposed QoS aware resource management framework. Various computing models with and without fogs are compared. Simulation results demonstrate the feasibility of large scale edge analytics services with HMCM and effectiveness of proposed QoS aware framework. The performance with HMCM is much better than that with existing individual models in terms of analytics job blocking probability and service utility. The matching algorithm also largely outperforms the baseline non-cooperative algorithms.

In Chapter 3, a case study on visual object detection with mobile local computing is conducted. Compared with traditional approaches using hand-engineered features, CNN achieved big performance improvement on object detection. However, due to the challenging driving environment (e.g., large object scale variation, object occlusion and bad light conditions), popular CNN detectors do not perform well over the KITTI autonomous driving benchmark dataset. We propose three methods for CNN based visual object detection for ADAS and autonomous driving. To address the large object scale challenge, deconvolution and fusion of CNN feature maps is proposed to add context and deeper features for better object detection at low feature map scales. In addition, soft-NMS is applied across object proposals at different feature scales to address the object occulusion challenge. As the cars, pedestrians and cyclists have distinct aspect ratio features, we measure their aspect ratio statistics and exploit them to set better anchor boxes for better object matching and localization. The proposed CNN methods are individually and jointly evaluated by extensive experiments over KITTI dataset. Experiment results demonstrate the effectiveness of the proposed methods with improved or comparable detection performance on KITTI test set.

In chapter 4, we take movie recommendation as an example for CAV entertainment applications with mobile cloud computing and focus on the design of recommendation system. Collaborative filtering (CF) is the most popular approaches used for recommendation systems, but it suffers from complete cold start (CCS) problem where no rating record are available and incomplete cold start (ICS) problem where only a small number of rating records are available for some new items or users in the system. We propose two recommendation models to solve the CCS and ICS problems for new items, which are based on a framework of tightly coupled CF approach and deep learning neural network. A specific deep neural network SADE is used

to extract the content features of the items. The state of the art CF model, timeSVD++, which models and utilizes temporal dynamics of user preferences and item features, is modified to take the content features into prediction of ratings for cold start items. Extensive experiments on a large Netflix rating dataset of movies are performed, which show that our proposed recommendation models largely outperform the baseline models for rating prediction of cold start items. The two proposed recommendation models are also evaluated and compared on ICS items, and a flexible scheme of model retraining and switching is proposed to deal with the transition of items from cold start to non-cold start status. The experiment results on Netflix movie recommendation show the tight coupling of CF approach and deep learning neural network is feasible and very effective for cold start item recommendation. The design is general and can be applied to many other recommendation systems for online shopping and social networking applications. The solution of cold start item problem can largely improve user experience and trust of recommendation systems, and effectively promote cold start items.

5.2 Future work

Based on the work in this thesis, we plan to work on the following directions in the future.

- More comprehensive evaluation of the hybrid mobile computing model and the QoS aware job management schemes will be performed in our future research work. Specifically, we are interested in more heterogeneous network and computing environments, and develop more effective schemes to allocate computing resources for given analytic tasks. In addition, vehicle mobility management and more design options will be investigated.

- We will investigate more CNN models and methods to improve object detection performance. The base network we used in this thesis is memory intensive and could be replaced with other light-weighted candidates.

- The future works could be extended to 3D object detection which utilizes data from multiple sensors such as radar, Lidar and camera. Not only RGB images, 3D voxel grids of objects could be exploited to achieve higher performance and safety for autonomous cars.

- In this thesis, we only focus on the object detection in a local vehicle. It is believed that connecting and exchanging the detection results between multiple nearby vehicles through CV technology could enlarge the detection area and improve the detection accuracy of each single vehicle. In the future, we are interested in the collaborative detection of multiple vehicles.

- Our recommendation models could be evaluated with more system configurations, parameters setting and metrics, in order to reveal more insights to their impact on recommendation performance and system optimization. The two separate recommendation models could be combined together to deal with both CCS and ICS problems. Moreover, it

will be interesting to apply our models to the recommendation of other products besides movies.

Bibliography

[1] J. Dargay, D. Gately and M. Sommer. "Vehicle ownership and income growth, worldwide: 1960-2030." *The Energy Journal*, 143-170, 2007.

[2] Mobileye. https://www.mobileye.com.

[3] Apollo. http://apollo.auto.

[4] N. Lu, N. Cheng, N. Zhang., X. Shen and J. W. Mark. "Connected vehicles: Solutions and challenges." *IEEE internet of things journal*, 1(4), 289-299, 2014.

[5] J. B. Kenney. "Dedicated short-range communications (DSRC) standards in the United States." *In IEEE*, 99(7), 1162-1182, 2011.

[6] ETSI http://www.etsi.org.

[7] A. Asadi, Q. Wang and V. Mancuso. "A survey on device-to-device communication in cellular networks." *IEEE Communications Surveys and Tutorials*, 16(4), 1801-1819, 2014.

[8] Y. Zhang, E. Pan, L. Song, W. Saad, Z. Dawy and Z. Han. "Social network aware device-to-device communication in wireless networks." *IEEE Transactions on Wireless Communications*, 14(1), 177-190, 2015.

[9] J. Stankovi. "Research directions for the internet of things." *IEEE Internet of Things Journal*, Vol. 1, No. 1, 3-9, March 2014.

[10] A. Zanella *et al.* "Internet of things for smart cities." *IEEE Internet of Things Journal*, Vol. 1, No. 1, 22-32, February 2014.

[11] F. Bonomi, R. Milito, P. Natarajan and J. Zhu. "Fog Computing: A Platform for Internet of Things and Analytics." *Big Data and Internet of Things: A Roadmap for Smart Environments*, 169-186, March 2014.

[12] Cisco. "Fog computing and the Internet of Things: extend the cloud to where the things are." White Paper, 2015.

[13] A. Fox. "A view of cloud computing." *Communications of the ACM*, Vol. 53, No. 4, 50-58, April 2010.

[14] C. Magurawalage, K. Yang, , L. Hu, J. Zhang. ''Energy-efficient and network-aware offloading algorithm for mobile cloud computing." *Computer Networks*, Vol. 74, 22-33, Dec. 2014.

[15] H. Dinh, C. Lee, D. Niyato, P. Wang. "A survey of mobile cloud computing: architecture, applications, and approaches." *Wireless Communications and Mobile Computing*, Vol. 13, No. 8, 1587-1611, Dec. 2013.

[16] M. Satyanarayanan, P. Bahl, R. Caceres, N. Davies. "The case for vm-based cloudlets in mobile computing." *IEEE Pervasive Computing*, Vol. 8, No. 4, Oct.-Dec. 2009.

[17] F. Bonomi, R. Milito, J. Zhu, S. Addepalli. Fog computing and its role in the internet of things." *In ACM MCC*, 13-16, 2012

[18] M. Gerla. "Vehicular cloud computing." *In IEE Med-Hoc-Net*, June 2012.

[19] M. Whaiduzzaman, M. Sookhak, A. Gani, R. Buyya. "A survey on vehicular cloud computing." *Journal of Network and Computer Applications*, Vol. 40, 325-344, April 2014.

[20] E. Lee, E. Lee, M. Gerla, S. Oh. "Vehicular cloud networking: architecture and design principles." *IEEE Communications Magzine*, Vol. 52, No. 2, February 2014.

[21] J. Wei, J. He, K. Chen, Y. Zhou. "Benchmarking of Distributed Computing Engines Spark and GraphLab for Big Data Analytics." *In IEEE BigDataService*, 2016.

[22] J. He, J. Wei, K. Chen, Z. Tang, Y. Zhou and Y. Zhang. "Multi-tier Fog Computing with Large-scale IoT Data Analytics for Smart Cities." *IEEE Internet Journal of Things Journal*, No. 99, July 2017.

[23] L. M. Vaquero, L. Rodero-Merino. "Finding your Way in the Fog: Towards a Comprehensive Definition of Fog Computing." *ACM SIGCOMM Computer Communication Review*, Vol. 44, No. 5, Oct. 2014.

[24] W. Shi, S. Dustdar. "The Promise of Edge Computing." *Computer*, 78-81, 2016.

[25] W. Shi, *et al.* "Edge Computing: Vision and Challenges." *IEEE Internet of Things Journal*, Vol. 3, No. 5, 637-645, October 2016

[26] M. Chiang, T. Zhang. Fog and IoT: An Overview of Research Opportunities *IEEE Internet Journal of Things Journal*, Vol. 3, No. 6, 854-864, Dec. 2016

[27] S. Sarkary, S. Chatterjee, S. Misraz. "Assessment of the Suitability of Fog Computing in the Context of Internet of Things." *IEEE Transactions on Cloud Computing*, 2015.

[28] X. Hou, *et al.* "Vehicular Fog Computing: A Viewpoint of Vehicles as the Infrastructures." *IEEE Transactions Vehicular Technology*, Vol. 65, No. 6, 3860-3913, June 2016.

[29] J. Ni, *et al.* "Security, Privacy, and Fairness in Fog-Based Vehicular Crowdsensing." *IEEE Communications Magazine*, 146-152, June 2017.

[30] S. Lien, *et al.* "Collaborative Radio Access of Heterogeneous Cloud Radio Access Networks and Edge Computing Networks." *In IEEE ICC 2016*, 3909-3914, 2016.

[31] S. Park, O. Simeone, and S. Shamai. "Joint Optimization of Cloud and Edge Processing for Fog Radio Access Networks." *IEEE Transactions on Wireless Communications*, Vol. 15, No. 11, 7621-7632, Nov. 2016.

[32] Y. Shih, it. "Enabling Low-Latency Applications in Fog-Radio Access Networks." *IEEE Network*, Jan. 2017.

[33] K. Liang, *et al.* "An Integrated Architecture for Software Defined and Virtualized Radio Access Networks with Fog Computing." *IEEE Network*, 80-87, Jan. 2017.

[34] N. Chen, *et al.* "Dynamic Urban Surveillance Video Stream Processing Using Fog Computing." *In IEEE Second International Conference on Multimedia Big Data*, 2016.

[35] M. Yannuzzi, *et al.* "A New Era for Cities with Fog Computing." *IEEE Internet Computing*, 54-67, 2017.

[36] J. Liu, *et al*, "Secure intelligent traffic light control using fog computing." *Future Generation Computer Systems*, 78:817âĂŞ824, 2018.

[37] I. Stojmenovic, S. Wen, X. Huang and H. Luan, "An overview of Fog computing and its security issues." *Concurrency Computatation: Practice and Experience*, 28:2991âĂŞ3005, 2016.

[38] F. Mehdipour, B. Javadi, A. Mahanti. "FOG-engine: Towards Big Data Analytics in the Fog." *In IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing*, 2016.

[39] R. Deng, *et al.* "Towards Power Consumption-Delay Tradeoff by Workload Allocation in Cloud-Fog Computing." *In IEEE ICC 2015 - Mobile and Wireless Networking Symposium*, 3909-3914, 2015.

[40] R. Deng, *et al.* "Optimal Workload Allocation in Fog-Cloud Computing Toward Balanced Delay and Power Consumption." *IEEE Internet of Things Journal*, Vol. 3, No. 6, 1171-1181, Dec. 2016.

[41] N. Wang, *et al.* "ENORM: A Framework For Edge NOde Resource Management." *IEEE Transactions Service Computing*, Jan. 2017.

[42] X. Chen and J. Zhang. "When D2D Meets Cloud: Hybrid Mobile Task Offloadings in Fog Computing." *In IEEE ICC 2017 Ad-Hoc and Sensor Networking Symposium*, 2017.

[43] H. Zhang, *et al.* "A Hierarchical Game Framework for Resource Management in Fog Computing." *IEEE Communications Magazine*, 52-57, August 2017.

[44] Y. Xiao and M. Krunz. "QoE and Power Efficiency Tradeoff for Fog Computing Networks with Fog Node Cooperation." *In IEEE INFOCOM*, 2017.

[45] R. Silva, J. Silva, F. Boavida. "Opportunistic Fog Computing: Feasibility Assessment and Architectural Proposal." *In IEEE Conference on Integrated Network and Service Management (IM)*, May 2017.

[46] Spark. http://spark.apache.org.

[47] A. Roy, I. Mihailovic and W. Zwaenepoel. "X-stream: edge-centric graph processing using streaming partitions." *In Proc. of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pp. 472-488, 2013.

[48] M. Zaharia, M. Chowdhury, M. Franklin, S. Shenker and I. Stoica. "Spark: cluster computing with working sets." *In Proceedings of the 2nd USENIX conference on Hot topics in cloud computing.*, June, 2010.

[49] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, I. Stoica. " Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing." *In NSDI*, 2-2, 2012.

[50] Raspberry Pi Foundation. http://www.raspberrypi.org.

[51] Y. Guo, *et al*, "How well do graph-processing platforms perform? an empirical performance evaluation and analysis." *In IEEE IPDPS*, 395-40, 2014.

[52] T. White. "Hadoop: The definitive guide". O'Reilly Media, Inc., 2012.

[53] M. Zaharia, M. Chowdhury, M. Franklin, S. Shenker, I. Stoica. "Spark: cluster computing with working sets." *In Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*., June, 2010.

[54] J. Dean, S. Ghemawat. "MapReduce: simplified data processing on large clusters." *Communications of the ACM*, 51(1), 107-113, 2008.

[55] Y. Low. "GraphLab: A Distributed Abstraction for Large Scale Machine Learning." *Doctoral dissertation, University of California, Berkeley*, 2013.

[56] J. E. Gonzalez, R. S. Xin, A. Dave, D. Crankshaw, M. J. Franklin, I. Stoica. "Graphx: Graph processing in a distributed dataflow framework." *In OSDI*, 599-613, 2014.

[57] A. Kyrola, G. E. Blelloch, C. Guestrin. "GraphChi: Large-Scale Graph Computation on Just a PC." *In OSDI*, 31-46, 2012.

[58] Raspberry Pi Foundation. http://www.raspberrypi.org.

[59] K. Shvachko, H. Kuang, S. Radia, R. Chansler. "The hadoop distributed file system". *In IEEE MSST*, 1-10, 2010.

[60] Cassandra. http://cassandra.apache.org.

[61] Hbase. http://hbase.apache.org.

[62] Amazon S3. https://aws.amazon.com/cn/s3.

[63] Spark. http://spark.apache.org.

[64] Y. Guo, M. Biczak, A. L.Arbanescu, A. Iosup, C. Martella, and T. L. Willke. "How well do graph-processing platforms perform? an empirical performance evaluation and analysis." *In IEEE IPDPS*, 395-40, 2014.

[65] M. Zaharia, M. Chowdhury, M. Franklin, S. Shenker, I. Stoica. " Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing." *In NSDI*, 2-2, 2012.

[66] MNIST. http://yann.lecun.com/exdb/mnist/.

[67] S. Zhao, R. Xiang, Y. Shi, P. Gao, and W. J. Li. "SCOPE: Scalable Composite Optimization for Learning on Spark." *arXiv preprint*, 1602.00133, 2016.

[68] D. Agarwal, and B. C. Chen. "Regression-based latent factor models." *In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 19-28, 2009.

[69] J. Bennett, and S. Lanning. "The netflix prize." *In Proceedings of KDD cup and workshop*, 2007, 35, 2007.

[70] T. Chen, W. Zhang, Q. Lu, K. Chen, Z. Zheng, and Y. Yu. "SVDFeature: a toolkit for feature-based collaborative filtering." *The Journal of Machine Learning Research*, 13(1), 3619-3622, 2012.

[71] P.G. Campos, F. DÃŋez, and I. Cantador. "Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols." *User Modeling and User-Adapted Interaction*, 24(1-2), 67-119, 2014.

[72] K. Georgiev, and P. Nakov. "A non-iid framework for collaborative filtering with restricted boltzmann machines." *In Proceedings of the 30th International Conference on Machine Learning*, 1148-1156, 2013.

[73] L. Hu, J. Cao, G. Xu, L. Cao, Z. Gu, and C. Zhu. "Personalized recommendation via cross-domain triadic factorization." *In Proceedings of the 22nd international conference on World Wide Web*, 595-606, 2013.

[74] Y. Koren, R. Bell, and C. Volinsky. "Matrix factorization techniques for recommender systems." *Computer*, (8), 30-37, 2009.

[75] Y. Koren. "Collaborative filtering with temporal dynamics." *Communications of the ACM*, 53(4), 89-97, 2010.

[76] G. Linden, B. Smith, and J. York. "Amazon. com recommendations: Item-to-item collaborative filtering." *IEEE Internet Computing*, 7(1), 76-80, 2003.

[77] B. Lika, K. Kolomvatsos, and S. Hadjiefthymiades. "Facing the cold start problem in recommender systems." *Expert Systems with Applications*, 41(4), 2065-2073, 2014.

[78] H. Ma, I. King, and M. R. Lyu. "Learning to recommend with explicit and implicit social relations." *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3), 29, 2011.

[79] R. Salakhutdinov, A. Mnih, and G. Hinton. "Restricted Boltzmann machines for collaborative filtering." *In Proceedings of the 24th international conference on Machine learning*, 791-798, 2007.

[80] R. Salakhutdinov, and A. Mnih. "Probabilistic matrix factorization." *In NIPS*, 20, 1-8, 2007.

[81] R. Salakhutdinov, and A. Mnih. "Bayesian probabilistic matrix factorization using Markov chain Monte Carlo." *In Proceedings of the 25th international conference on Machine learning*, 880-887, 2008.

[82] H. Shan, and A. Banerjee. "Generalized probabilistic matrix factorizations for collaborative filtering." *In 2010 IEEE 10th International Conference on Data Mining (ICDM)*, 1025-1030, 2010.

[83] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran. "Low-rank matrix factorization for deep neural network training with high-dimensional output targets." *In Acoustics, Speech and Signal Processing (ICASSP)*, 6655-6659, 2013.

[84] Y. Shi, M. Larson, and A. Hanjalic. "Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges." *ACM Computing Surveys (CSUR)*, 47(1), 3, 2014.

[85] D. Zhang, C. H. Hsu, M. Chen, Q. Chen, N. Xiong, and J. Lloret. "Cold-start recommendation using bi-clustering and fusion for large-scale social recommender systems." *IEEE Transactions on Emerging Topics in Computing*, 2(2), 239-250, 2014.

[86] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. "Generative models for cold-start recommendations." *In Proceedings of the 2001 SIGIR Workshop on Recommender Systems*, 6, 2001.

[87] U. Ocepeka, J. Rugeljb, and Z. Bosnica. "Improving matrix factorization recommendations for examples in cold start." *Experts Systems with Applications*, 42(19), 6784-6794, 2015.

[88] P. Victor, C. Cornelis, A. M. Teredesai, and M. De Cock. "Whom should I trust?: the impact of key figures on cold start recommendations." *In Proceedings of the 2008 ACM symposium on Applied computing*, 2014-2018, 2008.

[89] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. A. Manzagol. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." *The Journal of Machine Learning Research*, 11, 3371-3408, 2010.

[90] C. Wang, and D. M. Blei. "Collaborative topic modeling for recommending scientific articles." *In Proceedings of the 17th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, 448-456, 2011.

[91] H. Wang, N. Wang, and D. Y. Yeung. "Collaborative deep learning for recommender systems." *In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1235-1244, 2015.

[92] L. Xiong, X. Chen, T. K. Huang, J. G. Schneider, and J. G. Carbonell. "Temporal collaborative filtering with Bayesian probabilistic tensor factorization." *In SDM*, 10, 211-222, 2010.

[93] Y. Xiao, P. Ai, C. H. Hsu, H. Wang, and X. Jiao. "Time-ordered collaborative filtering for news recommendation." *China Communications*, 12(12), 53-62, 2015.

[94] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan. "Large-scale parallel collaborative filtering for the netflix prize." *In Algorithmic Aspects in Information and Management*, 337-348, 2008.

[95] Z. K. Zhang, C. Liu, Y. C. Zhang, and T. Zhou. " Solving the cold-start problem in recommender systems with social tags." *EPL (Europhysics Letters)*, 92(2), 28002, 2010.

[96] K. Zhou, S. H. Yang, and H. Zha. "Functional matrix factorizations for cold-start recommendation." *In Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, 315-324, 2011.

[97] C. Zhang, K. Wang, H. Yu, J. Sun, and E. P. Lim. "Latent factor transition for dynamic collaborative filtering." *In SDM*, 452-460, 2014.

[98] A. Geiger, P. Lenz, and R. Urtasun. "Are we ready for autonomous driving? the kitti vision benchmark suite." In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 3354–3361, 2012.

[99] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. "The pascal visual object classes (voc) challenge." *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[100] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. "Microsoft coco: Common objects in context." In *European conference on computer vision*, pp. 740–755, Springer, 2014.

[101] P. Felzenszwalb, R. B. Girshick, and D. McAllester. "Cascade object detection with deformable part models." In *CVPR*, pp. 2241-2248, 2010.

[102] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks." In *NIPS*, pp. 1097-1105, 2012.

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

[104] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. "Ssd: Single shot multibox detector." In *European conference on computer vision*, pp. 21–37, Springer, 2016.

[105] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos. "A unified multi-scale deep convolutional neural network for fast object detection." In *European Conference on Computer Vision*, pp. 354–370, Springer, 2016.

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

[107] P. Dollár, Z. Tu, P. Perona, and S. Belongie. "Integral channel features." 2009.

[108] P. Dollár, R. Appel, S. Belongie, and P. Perona. "Fast feature pyramids for object detection." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 8, pp. 1532–1545, 2014.

[109] Q. Hu, S. Paisitkriangkrai, C. Shen, A. van den Hengel, and F. Porikli. "Fast detection of multiple objects in traffic scenes with a common detection framework." *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1002–1014, 2016.

[110] R. N. Rajaram, E. Ohn-Bar, and M. M. Trivedi. "Looking at pedestrians at different scales: A multiresolution approach and evaluations." *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 12, pp. 3565–3576, 2016.

[111] X. Yuan, S. Su, and H. Chen. "A graph-based vehicle proposal location and detection algorithm." *IEEE Transactions on Intelligent Transportation Systems*, 2017.

[112] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. "Selective search for object recognition." *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[113] C. L. Zitnick and P. Dollár. "Edge boxes: Locating object proposals from edges." In *European Conference on Computer Vision*, pp. 391–405, Springer, 2014.

[114] R. Girshick, J. Donahue, T. Darrell, and J. Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

[115] R. Girshick. "Fast r-cnn." In *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

[116] J. Dai, Y. Li, K. He, and J. Sun. "R-fcn: Object detection via region-based fully convolutional networks." In *Advances in neural information processing systems*, pp. 379–387, 2016.

[117] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. "Feature pyramid networks for object detection." In *CVPR*, 2017.

[118] K. He, G. Gkioxari, P. Dollár, and R. Girshick. "Mask r-cnn." In *International conference on computer vision*, 2017.

[119] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. "You only look once: Unified, real-time object detection." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779–788, 2016.

[120] J. Redmon and A. Farhadi. "Yolo9000: Better, faster, stronger." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[121] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. "Dssd: Deconvolutional single shot detector." *arXiv preprint arXiv:1701.06659*, 2017.

[122] J. Jeong, H. Park, and N. Kwak. "Enhancement of ssd by concatenating feature maps for object detection." *arXiv preprint arXiv:1705.09587*, 2017.

[123] Y. Xiang, W. Choi, Y. Lin, and S. Savarese. "Subcategory-aware convolutional neural networks for object proposals and detection." In *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*, pp. 924–933, IEEE, 2017.

[124] F. Yang, W. Choi, and Y. Lin. "Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2129–2137, 2016.

[125] J. Ren, X. Chen, J. Liu, W. Sun, J. Pang, Q. Yan, Y.-W. Tai, and L. Xu. "Accurate single stage detector using recurrent rolling convolution." In *CVPR*, 2017.

[126] K. Simonyan and A. Zisserman. "Very deep convolutional networks for large-scale image recognition." *NIPS*, 2015.

[127] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis. "Improving object detection with one line of code." *arXiv preprint arXiv:1704.04503*, 2017.

[128] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*. "Imagenet large scale visual recognition challenge." *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[129] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun. "Monocular 3d object detection for autonomous driving." In *CVPR*, 2016.

[130] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka. "3d bounding box estimation using deep learning and geometry." In *CVPR*, 2017.

[131] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. "Multi-view 3d object detection network for autonomous driving." In *CVPR*, 2017.

[132] F. Chabot, M. Chaouch, J. Rabarisoa, C. TeuliÃĺre, and T. Chateau. "Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image." In *CVPR*, 2017.