Cross-Layer System Design for Autonomous Driving

by

Shih-Chieh Lin

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in The University of Michigan
2019

Doctoral Committee:

Assistant Professor Jason Mars, Co-Chair
Assistant Professor Lingjia Tang, Co-Chair
Assistant Professor Steve Oney
Associate Professor Thomas F. Wenisch

Shih-Chieh Lin

shihclin@umich.edu

ORCID iD: 0000-0002-8218-6253

To my parents, Yu-Chin Shih and Chi-Chou Lin.

ACKNOWLEDGEMENTS

Pursuing the doctoral degree is one of the most ambitious decisions in my life that has shaped me to who I am today. This dissertation would not have been possible without the guidance of my advisors and colleagues. My advisors, Jason and Lingjia, I can not thank you enough for how much I have learned from you. Jason, you encourage me to explore my ability and inspire me to confront any challenge with confidence. Lingjia, you drive me to pursue my excellence and teach me to defend my work. My dissertation committee, Jason, Lingjia, Tom and Steve, I thank you for your insights and guidance in constructing this dissertation.

I am grateful to be one of the members of Clarity Lab. I could not have enjoyed working with you more, which enlightens me to be a better researcher. We had countless nights debating and brainstorming ideas. Specifically, I want to thank Yunqi for being my mentor for both research and life. You teach me how to be confident in my research, be proud of my achievement, and be a humble and resilient person. You are and will always be the role model that I chase.

Lastly, nobody has been more important to me in the pursuit of this journey than my significant other, Ya-Wen, and my family, Yu-Chin, Chi-Chou, Hsiang-Lin and Man-Lin. You have been staying with me through all the ups and downs in the life. You will always be my Sirius in my starry sky at night, thank you. I am fortunate enough to have the greatest family in the world. You always remind me of being a kind and humble person. Thank you for always being by my side and share those good and bad times with me. A Ph.D. life could be tough, but becomes better because

of all of you. You are the reason why I became who I am today. To all my family, friends and all those who wished me well, thank you from the bottom of my heart.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

ABSTRACT

Autonomous driving has gained tremendous popularity and becomes one of the most emerging applications recently, which allows the vehicle to drive by itself without requiring help from a human. The demand of this application continues to grow leading to ever increasing investment from industry in the last decade. Unfortunately, autonomous driving systems remain unavailable to the public and are still under development even with the recent considerable advancement achieved in our community. Several key challenges are observed across the stack of autonomous driving systems and must be addressed to bridge the gap.

This dissertation investigates cross-layer autonomous driving systems from hardware architecture, software algorithms to human-vehicle interaction. In the hardware architecture layer, we investigate and present the design constraints of autonomous driving systems. With an end-to-end autonomous driving system framework we built, we accelerate the computational bottlenecks identified and thoroughly investigate the implications and trade-offs across various accelerator platforms. In the software algorithm layer, we propose an accelerating technique for object recognition, which is one of the critical bottlenecks in autonomous driving systems. We exploit the similarity across frames in streaming videos for autonomous vehicles and reuse the intermediate outputs computed in the algorithm to reduce the computation required and improve the performance. In the human-vehicle interaction layer, we design a conversational in-vehicle interface framework which enables drivers to interact with vehicles by using natural human language to improve the usability of autonomous driving features. We also integrate this framework into a commercially available vehicle and conduct a real-world driving study.

CHAPTER I

Introduction

Autonomous driving has attracted a significant amount of interest in the past few years as many companies in technology industry and automotive industry, such as Google, Tesla, Uber, Ford and BMW, have invested large amount of capital and engineering power on investigating and developing such application. Autonomous driving is an emerging application powered by Artificial Intelligence (AI) techniques to enable vehicles to understand the environment and move toward the destination without little or even no human input. Despite the significant advancements in our community to boost the power of autonomous driving in the last decade, vehicles equipped with autonomous driving capability are still largely under experiements and several cruicial hurdles observed prevent such applications from being available in the real-world use cases.

Particularly, three key challenges observed across the stack of autonomous driving systems from hardware architecture, software algorithms to human-vehicle interaction and must be addressed to bridge the gap between this emerging application and our community. First, the design constraints of architecting autonomous driving systems from hardware perspective remain unclear when applying state-of-the-art AI techniques (e.g., computer vision, machine learning and robotics), which recently leverage more sophisticated computation. To better understand the architectural im-

plications of such systems, an in-depth investigation to explore the viability of using different hardware platforms must be conducted. Second, the significant improving accuracy these algorithms and techniques are able to achieve introduces increasingly amount of computation resulting in higher processing latency even with modern hardware accelerators. To tackle the performance bottleneck, these techniques used in autonomous driving applications must be carefully redesigned from algorithmic perspective to explore the accelerating opportunities and improve the performance. Finally, an estimated 73% of drivers with autonomous driving features-enabled vehicles have not even attempted to use these features [22], which is mainly due to drivers have difficulty understanding these features and they are unsure how to activate and use autonomous driving features. To improve the usability of autonomous driving systems built in cars, it is important to investigate the gap and build an effective in-vehicle interface that help to bridge the gap.

This dissertation investigates these key challenges in leveraging state-of-the-art machine learning, deep learning algorithms to architect cross-layer autonomous driving systems from hardware architecture, software algorithms to human-vehicle interaction and proposes novel autonomous driving designs to address these key challenges.

1.1 Motivation

This section motivates the need for investigating autonomous driving systems across the stack from the perspective of hardware architecture, software algorithm and human-vehicle interaction in the context of the critical challenges facing modern autonomous driving applications.

1.1.1 Architectural Constraints

Many automotive companies like Google, Uber, Tesla, Mobileye and others have recently invested significantly in the future application known as autonomous driving

systems. The autonomous driving system allows the vehicle to drive by itself without requiring help from a human. The vehicle equipped with autonomous driving capability detects the environment, locates its position, and operates the vehicle to get to the specified destination safely without human input. Demand of this application continues to grow leading to ever increasing investment from industry. Intel recently acquired Mobileye, a leader in computer vision-based autonomous driving technology, for $15.3 billion [155]. Reports show that by 2035, automobiles with autonomous driving features are expected to capture 25% of the automotive market, which translates to 18 million vehicles [47], and the size of the autonomous driving vehicle market is expected to leap to $77 billion by 2035 [47].

Despite the recent advancements in autonomous driving systems contributed by industry leaders like Google, Tesla and Mobileye, autonomous driving vehicles are still largely under experimentation and research. As such, architecting the right autonomous driving system still largely remains an open research question.

Architecting autonomous driving systems is particularly challenging for a number of reasons. These systems must make the "correct" operational decision at all times to avoid accidents, thereby advanced machine learning, computer vision and robotic processing algorithms, typically computationally intensive, are employed to deliver the required high precision. Despite the large amount of computation, it is critical for such mission critical system to be able to react to the traffic condition at real-time, which means the processing always needs to finish at strict deadlines. Furthermore, the system needs to perform the necessary computation under certain power budget to avoid negatively impacting the driving range and fuel efficiency of the vehicle by large amounts.

To address these challenges as a research community, there are several key research questions arise:

1. What are the design constraints for building autonomous driving systems?

2. What are the computational profile and bottlenecks of a state-of-the-art end-to-end autonomous driving system?

3. What architecture should we use when building such systems to meet all the design constraints?

1.1.2 Algorithmic Bottlenecks

With the recent advances in computer vision especially leveraging machine learning techniques, video understanding has become one of the emerging applications drawing increasing amount of attention in both academia and industry. Automatic video understanding facilitates analysis on volumes of videos that are so large that it is infeasible for human to process (e.g., YouTube has more than 5 billion videos uploaded and users upload more than 400 hours of videos every minute [30]), power applications like video summarization, video captioning. In addition to analyzing large volumes of data offline, it presents the opportunity of using computer vision to assist human beings in various tasks like driving in an online real-time fashion (i.e., autonomous driving).

To better support such applications, precision is one of the critical factors that our research community has put in a lot of effort in the last couple decades, resulting in a huge boost in video understanding which in turn significantly increased the feasibility of these applications. As opposed to hard coding decision rules in conventional expert systems, much of the recent advance can only be achieved thanks to the data-driven approach we have been taken, which learns from large corpus of real-world samples. Specifically, Convolutional Neural Networks (CNNs) have been proven to be one of the most effective machine learning techniques in video understanding. In particular, Faster R-CNN [136] along with quite a few other R-CNN based algorithms have been consistently standing on the top of the PASCAL VOC object detection benchmark leaderboard, outperforming many other conventional techniques.

However, the constant improving accuracy these CNN-based algorithms can achieve does not come free. They have become increasingly complex with more number of layers in the network architecture as well as more complex structure within each layer, which drives up the computational power required to process such networks, resulting in higher and higher processing latency. With such computationally expensive algorithms, it is nearly infeasible to perform any meaningful analysis at large scales especially at the rate these videos get produced. Furthermore, such long processing latency makes it extremely difficult, if possible, for video understanding to perform any real-time tasks such as object detection and tracking in autonomous driving.

Researchers have proposed various ideas to reduce the processing latency by trading off accuracy. Network pruning [56, 59, 107, 57, 72, 143, 177, 2] is one common technique that prunes out redundant parameters to reduce the size of the neural networks. Weight quantization [70, 134, 69, 175, 29, 101, 83, 28] utilizes data types with reduced accuracy, like fixed-point numbers, when processing such neural networks to cut down the cost of algorithmic operations. However, these techniques all sacrifice a certain amount of accuracy to achieve the performance improvement, which makes them less attractive for real-world applications especially mission-critical ones like autonomous driving.

1.1.3 Human-Vehicle Interaction

Advanced Driver Assistance Systems (ADAS) are one of the fastest growing autonomous driving applications and have recently started to become widely deployed across the newer vehicle fleet. ADAS features are designed to either warn or assist in the control of a vehicle to help reduce the effects of human error while driving. For example, Lane Keeping System (LKS), which vibrates the steering wheel to alert drivers when their vehicles drift out of the lane, and Adaptive Cruise Control (ACC), which adjusts a vehicle's speed to maintain a certain distance from other vehicles, are

two widely-used ADAS features.

While the available data on the influence of ADAS features is limited, some studies have estimated a promising influence on the real-world driving experience. One study estimates that ADAS may help drivers prevent 28% of all crashes if all new car purchases included ADAS [53]. The market for ADAS has grown significantly and nearly all major automakers (including Ford, Chrysler, BMW, and Audi) integrate ADAS into their vehicles. The market for ADAS is projected to continue growing: McKinsey & Co. researchers have predicted the ADAS market will double its size in the next three years, reaching $35 billion in annual revenue [41].

However, an estimated 73% of drivers with ADAS-enabled vehicles have not even attempted to use these features [22]. This is due to a number of factors [35, 89], including the fact that ADAS are relatively new and constantly evolving. Many drivers are not familiar with ADAS and rarely read their owner's manuals on how to use them [124]. Specifically, there are gulfs of evaluation [123] (i.e., drivers have difficulty assessing the state of ADAS, such as if they are activated) and gulfs of execution [123] (i.e., drivers are unsure how to activate and use ADAS). Because ADAS features could help the driver maintain control of their vehicle, it is important to build user interfaces that help to bridge these gulfs.

Building an effective interface for ADAS is challenging for several reasons.

1. It is unclear what information drivers need to effectively use ADAS features.

2. The interface can not require any complex visual-manual interactions because asking the driver to perform such operations while driving could reduce safety.

3. These are complicated systems whose behavior depends on a complex combination of system states and vehicle contexts. For example, ACC might appear to be deactivated because the feature is malfunctioning, the system is in a state where it is not used, or the feature is inactive and the driver must be able to

distinguish between these three states.

4. Drivers often feel unclear what their roles entail when ADAS features are activated.

1.2 Cross-Layer System Design for Autonomous Driving

In this section, we summarize the proposed cross-layer system design for emerging autonomous driving applications.

1.2.1 Architectural Implications of Autonomous Driving

This dissertation first investigates the design constraints for autonomous driving systems and six classes of constraints are identified including performance, predictability, storage, thermal, power and others, and has arrived at some unique observations about autonomous driving systems. For instance, we discover it is critical for the system to be able to finish the end-to-end processing at a latency less than 100 ms and a frame rate higher than 10 frames per second, to react fast enough to the constantly changing traffic condition. To capture the stringent performance predictability requirements of such mission critical real-time system, tail latency (i.e., high quantiles of the latency distribution) should be used to quantify the performance of the system. We also find the power consumption of such system is heavily magnified (almost doubled) by the increased cooling load to remove heat generated by the computing system to keep the passenger cabin within a tolerable temperature, making it challenging to leverage power-hungry computational platforms without significantly degrading the vehicle driving range and fuel efficiency.

To understand the computational profile of such systems, we first need to address the challenge of there being a lack of publicly available end-to-end experimental frameworks that are representative of the state-of-the-art autonomous driving

systems, which places a significant obstacle for our community to investigate these systems. Therefore, we build an end-to-end autonomous driving system based on most recently published system designs from academic research [86] and industry practitioners [162]. The algorithmic components we use represent the most up-to-date advancements in relevant fields, as they have won the corresponding machine learning challenges recently (e.g., YOLO [135] won the multi-object detection challenge [37]). In this architecture (detailed in Section 3.1.3), the video captured by cameras are streamed into an object detection engine to detect interested objects and a localization engine to locate the vehicle based on the nearby landmarks in parallel. The detected object coordinates are then fed into an object tracking engine that tracks the moving objects to predict their moving trajectories. The output of the object detection engine and the localization engine is then fused onto the same 3D coordinate space to plan out the operational decisions.

With this end-to-end experimental framework, we discover three computational bottlenecks, namely object detection, object tracking and localization, dominate the majority of the computation. These computationally expensive bottlenecks prevent conventional multicore CPU systems from meeting the design constraints of such systems. Thus, we conduct an in-depth investigation to explore the viability of accelerating these algorithmic components using various accelerators including GPUs, FPGAs and ASICs, and we provide insights on how future architecture designs should evolve for this emerging and fast growing application.

1.2.2 Accelerating Object Recognition for Streaming Videos

In this work, we explore the opportunities of reducing the amount of data needs to be processed by the same neural networks without sacrificing the accuracy of the neural networks themselves. One unique insight we observe in this work is large fraction of the pixels remain the same across adjacent frames in continuous video

recordings. Exploiting this insight, we design a technique with an emphasis on Faster R-CNN [136] that reuses the understanding (e.g., objects detected or recognized) we have already gained in areas that did not change across frames, and only feeds the changing areas, which is significantly smaller, to the neural networks to process. For instance, like in an autonomous driving system, intermediate outputs like the feature maps and the region of interests (ROIs) extracted from the previous frame can be reused for the next frame as long as the pixels in the corresponding area haven't changed. By avoid having to send the entire frame to the costly neural networks to process, we can significantly reduce the total area the complex and time-consuming neural networks need to process, thereby reducing the overall processing latency, while still achieving the same accuracy.

1.2.3 Conversational In-Vehicle Digital Assistant

Currently drivers interact with ADAS features through controls on their steering wheel and indicators displayed on the dashboard. In this work, we propose to add a third modality: a speech-based conversational interface for ADAS features. We designed and built **Adasa**, the first speech-based conversational interface for ADAS. Adasa's features are based on our analysis of over 9,000 conversations between drivers and Ford's customer service division and includes additional training data generated by crowd workers. We built Adasa upon the state-of-the-art conversational machine learning platform, Lucida [61], which allows drivers to interact with Adasa in unconstrained natural language in real-time. Drivers can simply ask questions or issue commands after enabling Adasa by pressing a single button on the steering wheel.

Adasa can handle *queries* that do not require the current vehicle's status (e.g., the meaning of a symbol on the dashboard), *system diagnostic* questions related to the vehicle and ADAS state (e.g., "why is my wheel vibrating?"), or *commands* to control ADAS features via natural language. We integrated Adasa into a commercially

available vehicle and conducted a user study on 15 drivers in a real-world driving environment.

1.3 Summary of Contributions

This dissertation identifies the crucial bottlenecks in autonomous driving systems across the stack and proposes a thorough cross-layer system design to improve autonomous driving applications. A summary of specific contributions is as follows:

- **Architectural Implications of Autonomous Driving -** We identify and present the design constraints of autonomous driving systems in terms of performance, predictability, storage, thermal and power. Despite the stringent real-time performance constraints, the power consumption of such system is heavily magnified by the increased cooling load to remove the heat generated by the computing system, resulting in significant impact on the fuel efficiency and driving range of the vehicle. To investigate the design of autonomous driving systems, we build an end-to-end system whose architecture aligns with latest industry products. The models and algorithmic components we employ in this system are representative of the state-of-the-art techniques, as they recently won the corresponding machine learning challenges and awards. We identify three computational bottlenecks, including object detection, object tracking and localization, in the end-to-end system that must be accelerated to meet all the design constraints. To thoroughly investigate the implications and trade-offs across devices when accelerating these bottlenecks, we implement them on three different accelerator platforms including GPUs, FPGAs and ASICs and provide a quantitative analysis. We explore the landscape of acceleration-based autonomous driving design choices across the three accelerator platforms and discuss the implication trade-offs for future architecture among performance,

power and scalability.

- **Computation Reuse of Object Recognition for Videos -** We identify performance bottlenecks and acceleration opportunities in Faster R-CNN in real-world video understanding applications via system characterization. Based on the insight that majority of the frames in continuous videos do not change, we propose a novel method to reuse intermediate results of unchanged regions across frames, to reduce the amount of complex neural network computations, thereby improving the processing latency. The proposed method could be applied to existing two-stage object detectors (e.g., Faster R-CNN) directly without the need to re-train models or depend on specific hardware. We demonstrate the proposed idea can achieve substantial performance improvement (i.e., up to 1.27×) without sacrificing the accuracy of the algorithm.

- **Adasa: A Conversational In-Vehicle Digital Assistant -** We conduct an analysis of over 9,000 discussions between drivers and customer service representatives from a major auto manufacturer about ADAS to better understand drivers' information needs. We then propose Adasa, an in-vehicle digital assistant, that allows the driver to ask questions or command and control in human natural language while driving to help drivers understand the features and improve the usability. An evaluation of Adasa deployed on a production vehicle is conducted, which demonstrates the effectiveness of the proposed system in improving the usability of these ADAS features. Insights gained through our user study in terms of how the digital assistant system design affects the overall user experience, and how traffic conditions impact user interaction, including response length and query completeness.

CHAPTER II

Background and Related Work

In this Chapter, we survey the related literature and provide the background relevant to the topics covered in this dissertation. This includes prior works on autonomous driving system architectures, autonomous driving in-vehicle interfaces, and deep learning techniques for autonomous driving applications.

2.1 Autonomous Driving System Architectures

2.1.1 Autonomous Driving Systems

Prior work has surveyed the common algorithmic components for autonomous driving systems [86]. However, the algorithms presented by Kato et al. are quite outdated, which do not represent the state-of-the-art autonomous driving systems. To address this, we design and develop an end-to-end autonomous driving system with recently developed algorithms that offer significantly higher accuracy. Geiger et al. design and develop a benchmark suite of computer vision applications for studying and evaluating autonomous driving systems [45]. Amer et al. present a review of the state-of-the-art algorithmic components used for path tracking in autonomous driving systems [4].

2.1.2 Architectural Acceleration on Autonomous Driving

There is also a large body of work accelerating machine learning-based applications using various accelerator platforms [74, 55, 18, 62, 60, 108, 43, 148, 17, 16, 34, 105, 21, 3, 33, 88, 84, 178, 77, 139]. Specifically, GPUs have been shown to offer orders of magnitude performance improvement over multicore CPUs [65, 62, 60, 139]. This is because many machine learning algorithms spend a large fraction of their execution time performing matrix multiplication, which can be parallelized on the large number of threads offered by GPUs. The commonality exists in these machine learning algorithms, especially DNNs, allows researchers to design ASICs to accelerate them, which offer even higher performance benefits and energy efficiency [55, 18, 17, 16, 34, 105, 3, 33, 84]. FPGAs have been discussed as another alternative, which also provide high performance and energy efficiency with the additional capability of reconfiguring the fabric programmatically [108, 43, 148]. In addition to computation, prior work also explored novel memory architectures to bring memory closer to processors [21, 3, 88].

2.2 Accelerating DNN Techniques for Autonomous Driving

The computational requirements and applicability of deep neural networks (DNNs) and convolutional neural networks (CNNs) [95, 96] have prompted researchers to design novel DNN architecture to improve the efficiency [75, 65, 104, 69, 70, 57, 56, 82, 18, 127, 163]. Besides, many prior works have investigate how to accelerate object detection applied on video recognition, which usually incorporates temporal and contextual information in videos [76, 12, 176, 100, 184, 185, 186, 126]. We then provide a brief survey on related works.

2.2.1 DNN Architecture Acceleration

Network Pruning - There has been a large body of works improving the DNN efficiency via neural network pruning, which is a technique that iteratively prunes redundant parameters from the neural network and reduce the number of neurons in the DNNs [56, 59, 107, 57, 72, 143, 177, 2]. Han *et al.* [56, 57] investigated the crucial connections in the DNNs while learning and proposed a method to prune the unimportant connections, which is able to reduce the number of parameters by a factor of 9× on AlexNet [92]. Yang *et al.* [177] introduced energy-aware pruning which exploit the energy consumption of CNNs to guide the pruning process. Luo *et al.* [107] proposed a filter level pruning method called ThiNet, which identifies the importance for each filter and discard those trivial filters to improve the efficiency. Our method aims to investigate on reusing intermediate features of DNNs, which can be further optimized by network pruning approaches.

Weight Quantization - Many prior works investigate on reducing the precision for both floating-point and fixed-point formats of weights [70, 134, 69, 175, 29, 101, 83, 28]. Courbariaux *et al.* [28] investigated the neural network performance with three different precisions (i.e., floating-point, fixed-point and dynamic fixed-point) and found that using low precision is able to achieve sufficient performance. Hubara *et al.* [69] proposed Binarized Neural Networks (BNNs) where a neural network consists of binary weights and activations (i.e., +1, -1) and demonstrated significant speedup achieved. Rastegari *et al.* [134] presented XNOR-Net and BWN which apply binary values on both weights and inputs and are able to perform accurately and efficiently on ImageNet [140]. However, quantizing weights lead to loss of accuracy and models are required to be retrained depending on the methods applied whereas our method is able to achieve almost lossless accuracy and can be applied directly on pretrained models.

2.2.2 Object Detection Acceleration in Videos

Sparsity - Exploiting sparsity in DNNs achieves remarkable success by reducing the redundancy of matrix multiplication [170, 15, 54, 104, 149, 50, 51] and is widely used for video recognition tasks [55, 126, 94]. Liu *et al.* [104] introduced Sparse Convolutional Neural Networks (SCNNs) to reduce over 90% of parameters via maximum sparsity. Wen *et al.* [170] then proposed Structured Sparsity Learning (SSL) to regularize the structures of DNNs and identify and compact structure from complex DNNs to improve sparsity. Han *et al.* [55] investigated on compressed network model and proposed a sparse matrix-vector multiplication compute unit to improve the performance by orders of magnitudes. Pan *et al.* [126] proposed to generate sparse inputs by subtracting two consecutive frames which is then processed with a sparsity-aware hardware accelerator to achieve computational savings [55]. Nonetheless, specialized hardware is required to execute such sparse matrix-vector multiplication, which limit the design spaces to specific CNN models as well as hardware platforms.

Optical Flow - Various optical flow-based approaches have been utilized for video recognition tasks and achieve substantial success [40, 13, 71, 146]. Optical flow has also been utilized to solve vision tasks such as pose estimation, frame prediction and attribute transfer [128, 130, 183]. Fischer *et al.* [40] first applied deep CNNs to estimate the flow motion and demonstrate competitive accuracy. For improving video recognition performance, Zhu *et al.* [184, 185, 186] then based on FlowNet [40] proposed Deep Feature Flow (DFF), which executes expensive CNNs only on sparse key frames and propagates the corresponding deep feature maps to other frames via a flow field to advance the performance of video recognition tasks. However, it requires to re-train the proposed DNNs (i.e., FlowNet) when applying various object detection algorithms, which makes DFF less feasible to such rapid changing eras. In contrast, our proposed method can be plugged easily with existing two-stage object detectors (e.g., Faster R-CNN) to improve the efficiency for video recognition tasks.

Overall, Our method focuses on factorizing the size of inputs and targets at reusing features of DNNs in videos can be employed to various DNN models directly without the need to retrain models to further improve the inference performance.

2.3 Autonomous Driving In-Vehicle Interfaces

In general, the number and complexity of systems available to drivers has increased significantly [87]. HCI researchers have investigated how to improve in-vehicle user interfaces for systems like driving assistance, infotainment, entertainment and car-integrated mobile devices [11, 42, 106, 152, 171].

2.3.1 User Interfaces for Cars

To help drivers use these systems without being too distracted while driving, different types of in-vehicle interfaces have been built and studied [44, 87, 93, 98, 125, 144, 180]. However, most of these interfaces have been visual or tactile. Kern et al. explore the design space of driver-based automotive user interfaces, including a set of inputs (e.g., button, touchscreen and pedals) and outputs (e.g., multi-functional display, digital and analog speedometer) modalities [87]. Visual user interfaces in vehicles have been investigated, but the results show that driving experience and behavior would be affected notably, and these interfaces may distract drivers from the primary driving tasks (i.e., driving and focusing on the traffic) [85, 150]. On the other hand, Ohn-Bar et al. investigate how gesture interfaces for the in-vehicle systems should be designed to improve the driving experience. They present the feasibility to have gestural interfaces deployed in the cars for a wide range of in-vehicle functionalities [125]. Lee et al. study the implications for drivers when using voice interfaces and touch interfaces on semi-automated systems and find that drivers who use the voice interface to control automated driving have lower nervousness and make fewer driving mistakes than those who use the touch interface [98].

2.3.2 In-vehicle Voice Interfaces

There is a large body of work showing that voice interfaces allow drivers to effectively focus on driving and the environment. Among all the interface modalities, voice interfaces help improve driving safety compared to other interface modalities [8, 52, 90, 97, 98, 106, 121, 161]. Researchers also find that voice interfaces affect human behavior, as well as emotion. Graham et al. evaluate users' experience of using voice interfaces to perform secondary tasks while driving. Despite the fact that voice interface is slower, less accurate, and leading to lower task performance, the results show that users still consider it easy to learn and logical, expressing the preference over manual interface [52]. In addition, prior works present that drivers spend more time keeping their eyes on the road when using a speech interface than a manual interface [8, 121]. From the industrial point of view, in-vehicle voice interfaces are now widely deployed in commercialized vehicles such as Ford Sync [23], Toyota Entune [27] and GM MyLink [24]. Even mobile devices are designed to be able to connect to in-vehicle infotainment and entertainment systems via CarPlay [5] or Android Auto [49]. These technologies leverage advanced speech recognition techniques to allow users to interact with systems like navigation or multimedia entertainment via voice commands. However, recent commercial products mostly rely on constrained speech (i.e., using specific terms or formats) which can be distracting while driving as it may require a higher cognitive demand than unconstrained natural speech. In fact, interacting with conversational systems using natural human language is still challenging and remains a crucial problem [58, 171, 172]. In this work, we employ advanced machine learning techniques to design Adasa and enable drivers to interact with vehicles in unconstrained natural language in real-time. None of these aforementioned speech-based systems are particularly designed for ADAS features, which we found to be a gap and may directly affect the drivers' in-vehicle experience while driving as more ADAS features are introduced.

CHAPTER III

The Architectural Implications of Autonomous Driving: Constraints and Acceleration

Autonomous driving systems have attracted a significant amount of interest recently, and many industry leaders, such as Google, Uber, Tesla and Mobileye, have invested large amount of capital and engineering power on developing such systems. Building autonomous driving systems is particularly challenging due to stringent performance requirements in terms of both making the safe operational decisions and finishing processing at real-time. Despite the recent advancements in technology, such systems are still largely under experimentation and architecting end-to-end autonomous driving systems remains an open research question.

To investigate this question, we first present and formalize the design constraints for building an autonomous driving system in terms of performance, predictability, storage, thermal and power. We then build an end-to-end autonomous driving system using state-of-the-art award-winning algorithms to understand the design trade-offs for building such systems. In our real-system characterization, we identify three computational bottlenecks, which conventional multicore CPUs are incapable of processing under the identified design constraints. To meet these constraints, we accelerate these algorithms using three accelerator platforms including GPUs, FPGAs and ASICs, which can reduce the tail latency of the system by 169×, 10×, and

93× respectively. With accelerator-based designs, we are able to build an end-to-end autonomous driving system that meets all the design constraints, and explore the trade-offs among performance, power and the higher accuracy enabled by higher resolution cameras.

3.1 Autonomous Driving

To investigate autonomous driving systems, we first present a formal taxonomy of such systems defined at different levels of automation ranging from no automation to full automation, as well as where the current industry stands based on this taxonomy. We then introduce the computational pipeline of the state-of-the-art highly automated autonomous driving systems. Based on this pipeline, we investigate and formalize the design constraints of architecting such systems.

3.1.1 Level of Automation

To facilitate the development of highly autonomous vehicles (HAVs), the National Highway Traffic Safety Authority released a guideline for autonomous driving systems [166] in which they referred to the six levels of automation defined by SAE International [141].

- **No Automation (Level 0)** – The human driver must complete all driving tasks even with warnings from vehicles.
- **Driver Assistance (Level 1)** – The automated system shares steering and acceleration/deceleration responsibility with the human driver *under limited driving conditions* (e.g., high speed cruising), and the driver handles the remaining driving tasks (e.g., lane change).
- **Partial Automation (Level 2)** – The automated system fully controls the

steering and acceleration/deceleration of vehicles *under limited driving conditions*, and the human driver performs remaining driving tasks.

- **Conditional Automation (Level 3)** – The automated system handles all driving tasks *under limited driving conditions*, and expects that the human driver will respond to requests to intervene (i.e., resume driving).
- **High Automation (Level 4)** – The automated system handles all driving tasks *under limited driving conditions* even if the human driver does not respond to requests to intervene.
- **Full Automation (Level 5)** – The automated system takes full control of all driving tasks under *all driving conditions* that can be managed by a human driver.

In summary, level 1 and 2 of automation are still mostly driving assistance, where the human driver still handles a substantial portion of the driving tasks at all times under all conditions. Autonomous driving systems can take full driving responsibility at level 3-5 of automation under certain driving conditions, which are typically referred as HAVs. As they represent the future of autonomous driving systems, we focus on HAVs at level 3-5 for the rest of the work.

3.1.2 Current Industry Status

To understand where current industry stands, we survey the industry leaders in the level of automation, the computing platform and sensors they leverage as presented in Table 3.1. As shown in the table, even leading industry companies like Tesla and Waymo can only achieve level 2 or 3 of automation, where the human driver is still heavily involved in the control of the vehicle. It demonstrates the challenges in building autonomous driving vehicles and motivates our research community to investigate this emerging application.

Table 3.1: Summary of autonomous driving vehicles under experimentation in leading industry companies.

Manufacturer	Mobileye [114]	Tesla [159, 36]	Nvidia/Audi [156]	Waymo [169, 157, 48]
Automation	level 2	level 2	level 3	level 3
Platform	SoCs	SoCs + GPUs	SoCs + GPUs	SoCs + GPUs
Sensor	camera	camera, radar	lidar, camera, radar	lidar, camera, radar

Looking at the computing platforms and sensors these industry leaders use, most of them leverage a combination of SoCs and GPUs to provide the large amount of computational capacity needed for autonomous driving systems. Another interesting observation is both Nvidia/Audi and Waymo, who are able to build experimentation autonomous driving vehicles at level 3 of automation, use Light Detection and Ranging (LIDAR) as part of the sensing devices, which is a remote sensing device used to examine surroundings of the vehicle at high precision by sending light beams. Although the high precision makes LIDAR a great fit as a sensing device for autonomous driving systems, the extreme high cost of LIDAR has been one of the primary reasons that prevent such systems from being commercially available on the market. Commercially available LIDAR devices are as expensive as $75,000 USD [167], which is much higher than the cost of the vehicle itself, even for some luxury cars. As a result, the industry has been trying to move away from LIDAR devices, and build vision-based autonomous driving systems instead, using only cameras and radars that are much cheaper for sensing the surroundings. For instance, companies like Mobileye [115, 117] and Tesla [158] have recently announced their plan for focusing on vision-based autonomous driving systems which are composed of mainly cameras and radars as sensing devices. Therefore, we focus on vision-based autonomous driving systems in this work.

3.1.3 Autonomous Driving Pipeline

The task of autonomous driving is to operate the vehicle to reach a given destination, with the data captured on various real-time sensors such as video cameras, laser scanners, and milliwave radars. The autonomous driving system then performs the necessary processing to recognize the driving environments and makes operating decisions. Such system is often composed of three major components: scene recognition for localizing the vehicle at decimeter-level and tracking nearby objects, path planning for generating the future paths, and vehicle control for physically operating the vehicle to follow the planned paths [86]. These algorithm components are the basis of most modern autonomous driving systems, which has been confirmed by the self-driving car Udacity built [162], and also aligns with how Mobileye designs their autonomous driving systems [115]. A detailed diagram of these components is presented in Figure 5.1.

The captured sensing data is first fed to an object detector (step 1a in Figure 5.1) and a localizer (step 1b in Figure 5.1) in parallel. The object detector detects the objects of interest around the vehicle, such as other vehicles, pedestrians, and traffic signals. The detected objects are then passed to an object tracker (step 1c in Figure 5.1) to associate the detected objects with their movements in the past, to predict the trajectories of moving objects. In parallel, the localizer determines the location of the vehicle at high precision. Subsequently, the object movement information from object tracker and the vehicle location information from localizer is combined and projected onto the same 3D coordinate space by a sensor fusion engine (step 2 in Figure 5.1).

The fused information about the vehicle location and moving objects is then passed to the motion planning engine to assign path trajectories (step 3 in Figure 5.1), such as lane change and setting the vehicle's velocity. The mission planning engine calculates the detailed operating motions to realize the planned paths and determine the routing

path from source to destination (step 4 in Figure 5.1). The vehicle control engine simply follows the planned paths and trajectories by operating the vehicle (step 5 in Figure 5.1).

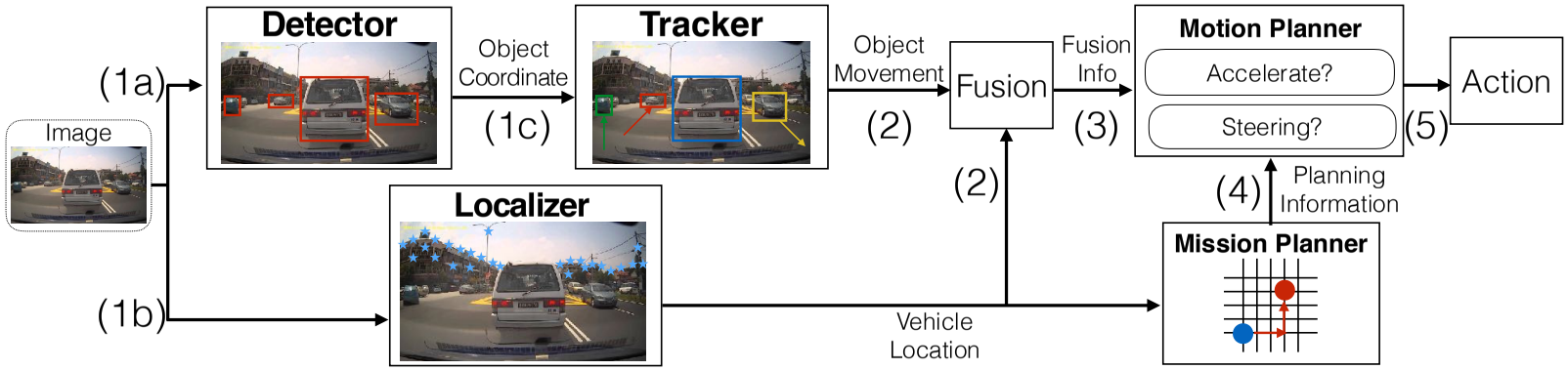


Figure 3.1: Overview of a state-of-the-art autonomous driving system, which is designed based on recent publications [86] and specifications released by industry companies [162, 115]. The video captured by cameras are streamed into both the object detection engine to detect objects (1a) and the localization engine to locate the vehicle (1b) in parallel. The detected objects then are passed to the object tracking engine to track moving objects (1c). The vehicle location and the tracked objects are projected into the same 3D coordinate space by the fusion engine (2), which will be consumed by the motion planner (3) to make operational decisions (5). The mission planner is only invoked when the vehicle deviates from the original routing plan generated by the navigation services like Google Maps (4).

3.1.4 Design Constraints

Despite the extremely detailed regulations on conventional automobiles (e.g., crash test, fuel economy, vehicle inspection), regulatory authorities have only recently started forming these regulations regarding autonomous driving vehicles. In the Federal Automated Vehicle Policies published by the U.S. Department of Transportation [166], it was only mentioned that "significant emphasis should be placed on software development, verification and validation" without any specific details. Therefore, many of the design constraints we discuss in this section are derived from published materials by industry practitioners like Toyota [80], Udacity [162] and Mobileye [115].

3.1.4.1 Performance Constraints

To avoid car accidents, the autonomous driving system needs to be able to "understand" the real-time traffic condition and react to it fast enough. While autonomous vehicles have the potential to reduce traffic casualties, the actual performance requirement for autonomous driving system is still largely undefined. According to prior work in driver-assistance systems [118], the reaction time of an autonomous driving system is determined by two factors.

- **Frame rate:** The frame rate determines how fast the real-time sensor data can be fed into the process engine.
- **Processing latency:** The processing latency of recognizing scenes and making operational decisions determines how fast the system can react to the captured sensor data.

Human drivers take varying amount of time to respond based on the level of expectation and action chosen. For example, human drivers take 600 ms to react when they are expecting a possible interruption and 850 ms otherwise [79]. A typical driver takes

0.96 s to release accelerator, 2.2 s to reach maximum breaking, and 1.64 s to begin steering to avoid an accident [110]. The fastest possible action by a human driver takes 100–150 ms [122, 160]. To provide better safety, autonomous driving systems should be able to react faster than human drivers, which suggests the latency for processing traffic condition should be within 100 ms. This aligns with the industry standards recently published by Mobileye [147] and the design specifications from Udacity [162].

In addition to processing latency, autonomous driving systems also need to frequently update their "understanding" to keep up with the continuously changing real-time traffic condition. In other words, the frame rate needs to be high, in case the real-time traffic condition changes drastically between two neighboring frames. To react quickly to the constantly changing traffic condition, the system should be able to react faster than human reaction time, which suggests a frequency of once every 100 ms. This also aligns with the frame rate of the collision prevention assistant systems built by Mobileye [116].

Performance Constraints: Autonomous driving system should be able to process current traffic conditions within a latency of 100 ms at a frequency of at least once every 100 ms.

3.1.4.2 Predictability Constraints

Autonomous driving is one of the mission critical applications that must be performed at real-time. What this means is the processing fails if not completed within a specific deadline, thereby the performance predictability is critical. Not being able to process in real-time can put the passengers in danger, and sometimes result in fatal accidents. Therefore, the performance of autonomous driving systems needs to be extremely predictable for them to be widely adopted. The predictability is defined as both the temporal aspects (i.e., meeting the specified timing deadline) and the func-

tional aspects (i.e., making the correct operational decisions). From an architect's point of view, we focus on the predictability of the temporal aspects.

Specifically, the predictability of the processing latency is critical for the autonomous driving system to quickly react to the real-time traffic condition reliably. To capture the non-determinism of large-scale distributed systems, tail latency, defined as the high quantiles of the latency distribution (e.g., 95th-, 99th- percentile latency), is often used to evaluate the performance of such systems instead of mean latency. As we will show in Section 3.2.2, the localization algorithm has large performance variability, which is challenging for the autonomous driving system to react to the real-time traffic. As a result, tail latency, high quantiles like 99.99th- percentile or even worst case latency, should be used to evaluate the performance of such systems to reflect the stringent predictability requirements. We will also empirically demonstrate why tail latency should be used in Section 3.4.1.2.

Predictability Constraints: Due to the large performance variability of autonomous driving systems, tail latency (e.g., 99th-, 99.99th- percentile latency) should be used as the metric to evaluate the performance, in order to capture the stringent predictability requirement.

3.1.4.3 Storage Constraints

While GPS technology has been commonly adopted to identify the vehicle location for navigation systems, it does not provide the necessary level of precision (e.g., precision at decimeter-level is needed to keep the vehicle staying in certain lanes [99]) and the spacial accessibility [9] to localize the vehicle (step 1b in Figure 5.1) for autonomous driving tasks. Therefore, prior map-based localization has been widely used to provide localization capability at centimeter-level precision [173, 174, 182, 111, 151, 120], where the surrounding view is transformed into feature descriptions to map the feature points stored in the prior map to identify the location of the

vehicle. However, it is infeasible to transmit the prior map from the cloud all the time, because the vehicle does not always have access to the Internet and the vehicle still needs to perform the necessary autonomous driving tasks even under limited accessibility. Therefore, the prior map needs to be stored on the autonomous driving vehicle.

However, prior maps of large environments (e.g., an entire country) consume significant amount of storage space. For example, a prior map of the entire United States takes 41 TB of storage space on an autonomous driving system [165].

Storage Constraints: Tens of TBs of storage space is needed to store the prior maps in large environments required by autonomous driving systems to localize the vehicle (e.g., 41 TB for an entire map of the U.S.).

3.1.4.4 Thermal Constraints

There are two aspects of thermal constraints in autonomous driving systems: 1) the temperature of the space to hold the computing system needs to be within the operating range the system can operate under; 2) the heat generated by the computing system should have relatively small impact on the thermal profile of the vehicle (e.g., not heat up the engine, which can potentially affect the reliability of the vehicle).

There are typically two temperature zones in modern autonomous driving vehicles: in the climate controlled passenger cabin or outside of [80]. Outside the passenger cabin, the operating temperature can get up to +105°C ambient [80], which is higher than most general-purpose computer chips could safely operate under (e.g., a typical Intel processor can only operate at temperature lower than 75°C [73]). Therefore, the autonomous driving system should be placed inside the climate controlled passenger cabin to avoid having to build climate control functionality for additional zones.

However, the passengers may no longer be able to tolerate the increased temperature when the computing system is placed inside the passenger cabin without

additional cooling infrastructure. For instance, a computing system that consumes 1kW power (e.g., 1 CPU and 3 GPUs operating at full utilization) will raise the temperature by 10°C in a minute if no additional cooling is added in the passenger cabin [39]. In conclusion, additional air conditioning load needs to be added to remove heat generated by the autonomous driving systems to keep the passenger cabin temperature tolerable.

Thermal Constraints: The computing system of autonomous driving vehicle needs to be put into the climate controlled passenger cabin to be able to operate safely, which means additional cooling capacity needs to be added to remove the additional heat generated by the computing system to maintain a tolerable temperature in the passenger cabin.

3.1.4.5 Power Constraints

In gasoline powered cars, the electrical system is typically on the order of 1-2 kW provided by the car's alternator [113, 129], which could be increased at the cost of reduction in fuel efficiency of the vehicle [142]. The exact reduction varies depending on the fuel efficiency of the car, but a rule of thumb for gas powered cars is that the miles per gallon (MPG) rating will be reduced by one for every additional 400 W of power consumption [38] (e.g., an additional 400 W power consumption translates to a 3.23% reduction in MPG for a 2017 Audi A4 sedan with 31 MPG originally [7]). Similarly, the additional power consumption will reduce the total driving range of electric vehicles (EVs) [158] due to the limited battery capacity.

The total power consumption of the autonomous driving system includes the consumption of the computing system, storage overhead (Section 3.1.4.3) and cooling overhead (Section 3.1.4.4). While the power consumption of the computing system heavily depends on the computing platform (e.g., CPUs, GPUs), a typical storage system consumes around 8 W to store every 3 TB data [145]. To remove the addi-

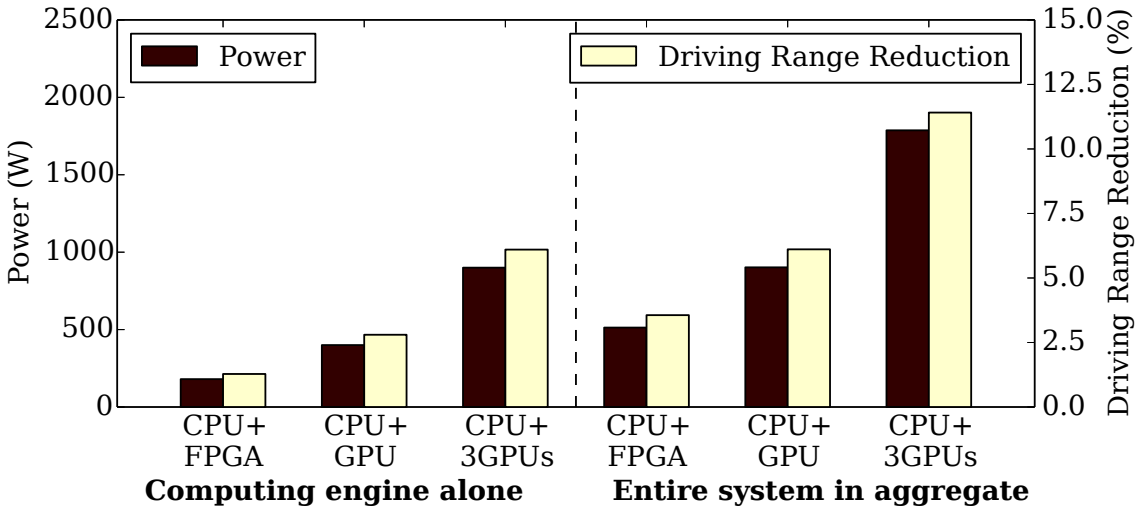


Figure 3.2: Driving distance per charge reduction contributed by the computing engine alone (on left) and the entire system in aggregate (on right) with respect to the additional power consumption generated by autonomous driving systems. The power consumption is heavily magnified by the storage engine and especially the cooling overhead resulting in driving range reductions as much as 11.5%.

tional heat generated by the system, a typical automobile air conditioner consumes around 77% of the cooling load to dissipate the heat (i.e., a coefficient of performance of 1.3, representing the ratio of useful cooling provided to work required [81]). That is to say, a 100 W system imposes 77 W cooling overhead to remove the additional heat generated by the system.

In Figure 3.2, we analyze the reduction in driving range of a Chevy Bolt [19] as additional power needs are placed on an electric vehicle. The first three sets of bars on the left represent the power consumption and driving range reduction contributed by the computing engine only, and the ones on the right show the corresponding metrics for the entire system in aggregate (i.e., including storage and cooling overhead). Surprisingly, the computing engine only contributes about half of the power consumption and the driving range reduction, where the storage engine and especially the cooling overhead almost double the impact. For example, a computing engine equipped with 1 CPU and 3 GPUs operating at full utilization alone only reduces the driving range by 6%, while the entire system experiences almost doubled reduction (i.e., 11.5%).

Power Constraints: The power consumption of autonomous driving system consists of the consumption of the computing engines and the storage engine, and is heavily magnified by the required cooling capacity to remove the additional heat. Having a power-hungry system can significantly degrade the vehicle fuel efficiency (e.g., as much as 11.5%).

3.1.4.6 Other Constraints

Additionally, there are also other constraints that we are not focusing on in this work. Any equipment used in the car should be able to sustain the impulse and vibration of the vehicles. Sudden impulses can range somewhere between 50 g to 500 g (g is used to measure impulse and stands for gravitational force) and vibrations can be up to 15 g and 100–2000 Hz [80]. Hardware reliability is also a constraint in real-time systems, where airplanes typically employ triple-modular-redundancy to provide the safety guarantee [179]. However, autonomous driving vehicles experience much less environment variability (e.g., temperature, atmospheric pressure) than airplanes, which makes it less likely for rare events like radiation-induced soft errors to occur.

3.2 End-to-End System

Due to the lack of a publicly available experimental framework, we build an end-to-end workload to investigate the architecture implications of autonomous driving vehicles. In this section, we detail the state-of-the-art algorithmic components that constitute this end-to-end autonomous driving system and the design decisions in choosing these algorithmic components. We then characterize the end-to-end system to understand its computational profile and the potential bottlenecks for meeting all the design constraints.

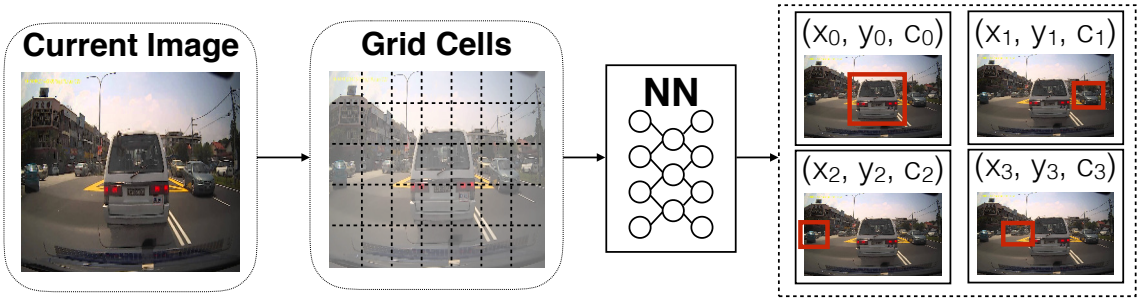


Figure 3.3: Overview of the object detection engine (DET). It partitions the input image into sub-regions and predicts the coordinates of detected objects and the confidence for each sub-region using Deep Neural Networks (DNNs).

3.2.1 Algorithmic Components

To build a representative end-to-end system for the state-of-the-art autonomous driving system, we select the best publicly available algorithms from the corresponding machine learning benchmark suites (e.g., VOT benchmark [91] for object tracking tasks).

3.2.1.1 Object Detection

For object detection (DET), we use YOLO [135], a DNN-based detection algorithm, which outperforms all the other multiple object detection algorithms in both accuracy and speed on VOC benchmark suite [37]. Figure 3.3 presents an overview of the algorithm. It first partitions the image into sub-regions and predicts the coordinates of detected objects and the confidence for each region through fully convolutional networks. A threshold is then used to filter out the objects detected with lower probabilities. In the output. we focus on four categories that we care the most in autonomous driving, including vehicles, bicycles, traffic signs and pedestrians.

3.2.1.2 Object Tracking

For object tracking (TRA), we use GOTURN [64], a DNN-based single object tracking algorithm, which outperforms most state-of-the-art trackers in speed and

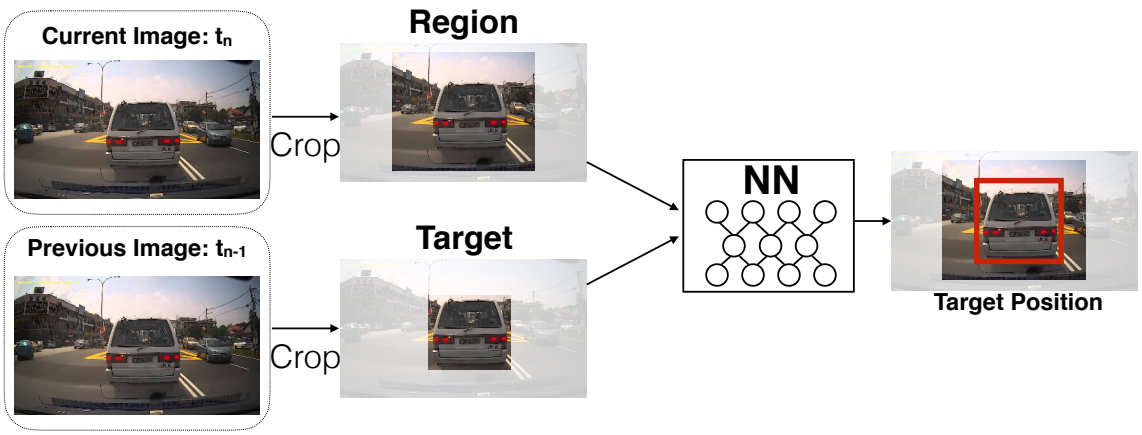


Figure 3.4: Overview of the object tracking engine (TRA). It crops the next image into a search region and the previous image into the tracking target only based on previous results. It then uses Deep Neural Networks (DNNs) to locate the target object in the search region.

accuracy in the VOT object tracking benchmark [91]. As shown in Figure 3.4, it crops the current image into a search region and the previous image into the tracking target only based on the previous result. Both the search region and the target are then fed as inputs to the neural networks, which generates the bounding box of the target to be tracked in the current image.

A pool of trackers is launched initially to wait for incoming tracking requests to avoid the initialization overhead. To record the number of objects tracked, we implement a tracked object table to store the objects that are being tracked currently. A threshold is set to determine whether an object is passing or leaving if it does not appear in ten consecutive images. We remove it from the tracked object table and set the tracker idle to wait for more incoming tracking requests.

3.2.1.3 Localization

For localization (LOC), we use ORB-SLAM [119], which has been ranked top of the available open source algorithms for localizing the vehicle on the KITTI datasets [45], as well as on the TUM benchmark suite [153]. Besides the high accuracy, the algorithm is also capable of localizing the vehicle regardless of viewpoints, which makes

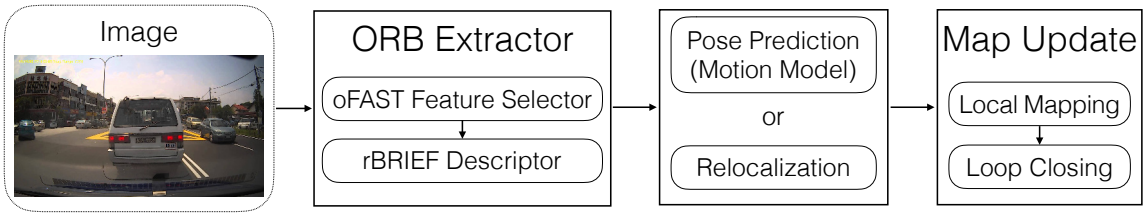


Figure 3.5: Overview of the localization engine (LOC). The algorithm takes images as input and extracts interesting features such as landmarks, and then generates a descriptor for each of the extracted features. The feature descriptors are consumed to locate the vehicle and predict vehicle poses.

it a great fit for autonomous driving systems. Figure 3.5 presents the details of the algorithm. The incoming video stream is fed into the ORB extractor to detect feature points using oFAST algorithm. The rBRIEF algorithm is then invoked to generate descriptions of the extracted feature points.

ORB-SLAM then attempts to match the current descriptions with the prior map database (as mentioned in Section 3.1.4.3) to localize the vehicle position. It uses a constant motion model to identify the location if the matching succeeds, otherwise it relocalizes the position by using a wider search in the map around the location identified last time.

In addition, the algorithm needs to update the map in case the current surroundings are different from the prior map (e.g., the map is built under different weather conditions). Lastly, loop closing is executed periodically to detect the potential trajectory loop when the vehicle is moving in order to calibrate the vehicle position based on the map database.

3.2.1.4 Fusion

The fusion engine (FUSION) retrieves the coordinates of the objects being tracked by the trackers, and combines with the current vehicle location provided by the localization engine. The combined information is transformed into the same 3D coordinate space, and sent to the motion planning engine to make vehicle operation decisions.

3.2.1.5 Motion Planning

For motion planning (MOTPLAN), we use the algorithms in the Autoware, which is a recently published open-source framework [86]. It leverages a graph-search based approach to find the minimum-cost path in space lattices when the vehicle is in an large opening area like parking lot or rural area [131]. When the vehicle is in structured areas (e.g., downtown area in cities), it uses conformal lattices with spatial and temporal information to adapt the motion plan to the environment [112, 164].

3.2.1.6 Mission Planning

For mission planning (MISPLAN), we also adopt the implementation from the state-of-the-art Autoware framework [86]. The policy is a rule-based approach which takes the traffic rules and the driving area condition to determine the routing path trajectory, which aligns with what industry leader Mobileye has been using [147]. The algorithm operates the vehicle to follow the route generated by navigation systems such as Google Maps. It is only executed once unless the vehicle deviates from planned routes.

3.2.2 System Characterization

To understand the computational profile and identify the potential bottlenecks in autonomous driving systems, we characterize our system using representative KITTI dataset [45] on an Intel server compiled with Intel Math Kernel Library (MKL), and the hardware specifications are listed in Table 3.2.

Figure 3.6 shows the measured mean, 99th- and 99.99th-percentile latency for each of the algorithmic components. Mission Planning (MISPLAN) is not included because it is only invoked once at the beginning to plan the route and will not be executed again unless the vehicle deviates from the routing path. As shown in the figure, the latency contributed by each of DET, TRA, and LOC individually has

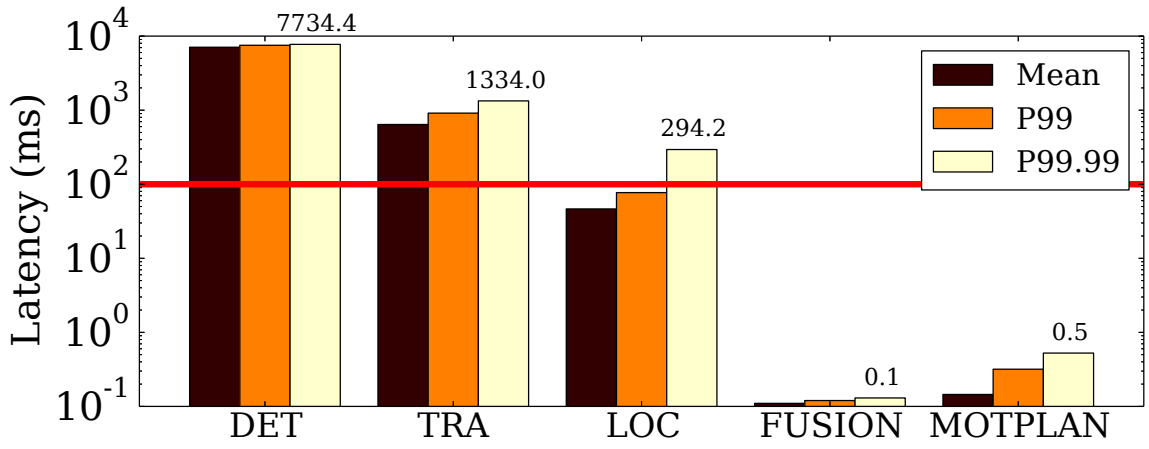


Figure 3.6: Latency of each algorithmic component on a multicore CPUs system in the end-to-end autonomous driving system. The latency contributed by each of object detection engine (DET), object tracking engine (TRA), and localization engine (LOC) has already exceeded the latency requirement of the end-to-end system. These three components dominate the end-to-end latency, and thereby are the computational bottlenecks that prevent us from meeting design constraints.

already exceeded the end-to-end system latency constraints at 100 ms, and these three components dominate the system latency. Therefore, we conclude DET, TRA and LOC as the computational bottlenecks of our autonomous driving system, and we will focus on them for the rest of the work.

We then characterize DET, TRA and LOC in details to investigate where they spend most of their computing cycles. Figure 3.7 shows the cycle breakdown of each algorithmic component. From the figure, we can easily identify that Deep Neural Networks (DNNs) is the most computational intensive portion in DET and TRA, as they consume 99.4% and 99.0% of the execution time respectively. Unlike the other two DNN-based algorithms, Feature Extraction (FE) consumes more than 85% of the execution time in LOC. These large fractions indicate the DNN portion in DET and TRA, and the FE portion in LOC are good candidates for acceleration in order to meet the strict real-time processing performance constraints for autonomous driving systems.

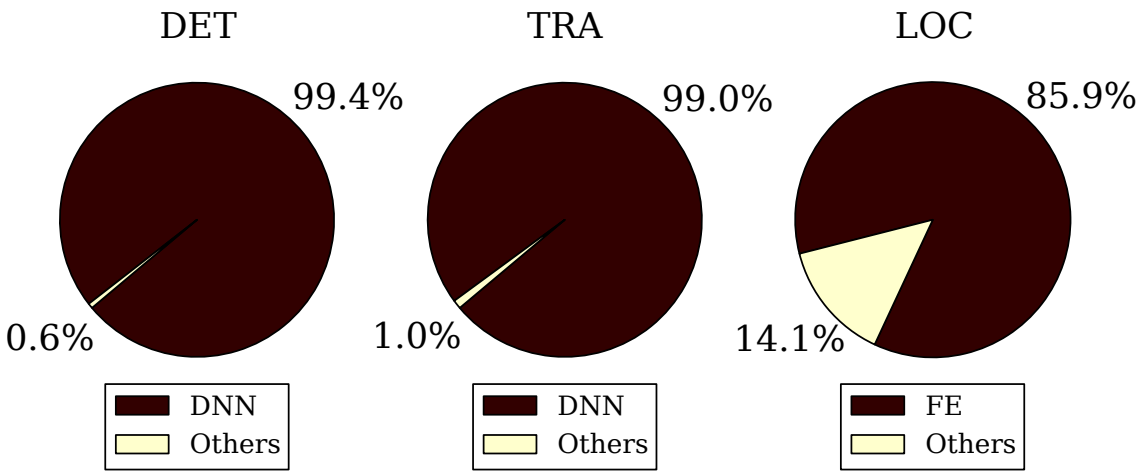


Figure 3.7: Cycle breakdown of the object detection (DET), object tracking (TRA) and localization (LOC) engines. The Deep Neural Networks (DNNs) portion in DET and TRA, and the Feature Extraction (FE) portion in LOC account for more than 94% of the execution in aggregation, which makes them ideal candidates for acceleration.

3.3 Accelerating Autonomous Driving

As demonstrated in Section 3.2.2, conventional multicore CPU systems are not suitable to meet all the design constraints, particularly the real-time processing requirement. Therefore, we port the critical algorithmic components to alternative hardware acceleration platforms, and investigate the viability of accelerator-based designs. In this section, we detail our design and implementation on these accelerator platforms, and evaluate their performance in regards to the design constraints of autonomous driving systems.

Table 3.2: Computing platform specifications. **: DSP (Digital Signal Processor)*

	CPU	GPU	FPGA	ASIC (CNN)	ASIC (FC)	ASIC (LOC)
Model	Intel Xeon E5-2630 v3	NVIDIA TitanX (Pascal)	Altera Stratix V	TSMC 65 nm	TSMC 45 nm	ARM 45 nm
Frequency	3.20 GHz	1.4 GHz	800 MHz	200 MHz	800 MHz	4 GHz
# Cores	16	3584	256*	N/A	N/A	N/A
Memory	128 GB	12 GB	2 GB	181.5 KB	N/A	N/A
Memory BW	59.0 GB/s	480.0 GB/s	6.4 GB/s	N/A	N/A	N/A

3.3.1 Accelerator Platforms

We focus on three different state-of-the-art computing platforms including GPUs, FPGAs and ASICs, and use multicore CPUs as our baseline. The detailed specifications of the hardware are listed in Table 3.2. The multicore CPU platform we use is a server-grade dual-socket machine with 8 cores on each socket, which represents a more conventional computing system design. The GPU accelerator we study is a latest Nvidia Titan X GPU with Pascal microarchitecture with 12 GB on-board memory, which offers powerful computing capability with 3584 cores. For FPGA platform, we employ an Altera Stratix V development board equipped with 256 Digital Signal Processors (DSPs) and large reconfigurable fabric. Lastly, we explore ASIC designs using prior work built on TSMC 65nm and 45nm technology [18, 55], and also build our own implementation using ARM 45nm technology.

3.3.2 Porting Methodology

3.3.2.1 GPU Implementation

To port the three bottleneck algorithmic components (i.e., DET, TRA and LOC) to GPUs, we leverage highly optimized machine learning software libraries. In particular, we implement the YOLO [135] object detection algorithm using the cuDNN library provide by Nvidia [20]. We port GOTURN [64] object tracking algorithm to GPUs using Caffe [78], which again allows us to benefit from the highly optimized cuDNN library [20]. The ORB-SLAM [119] algorithm used for localization is ported to GPUs with OpenCV library [10].

3.3.2.2 FPGA Implementation

To port the three bottleneck algorithmic components to FPGAs, we focus on the most time-consuming algorithms, namely DNN and Feature Extraction (FE), which

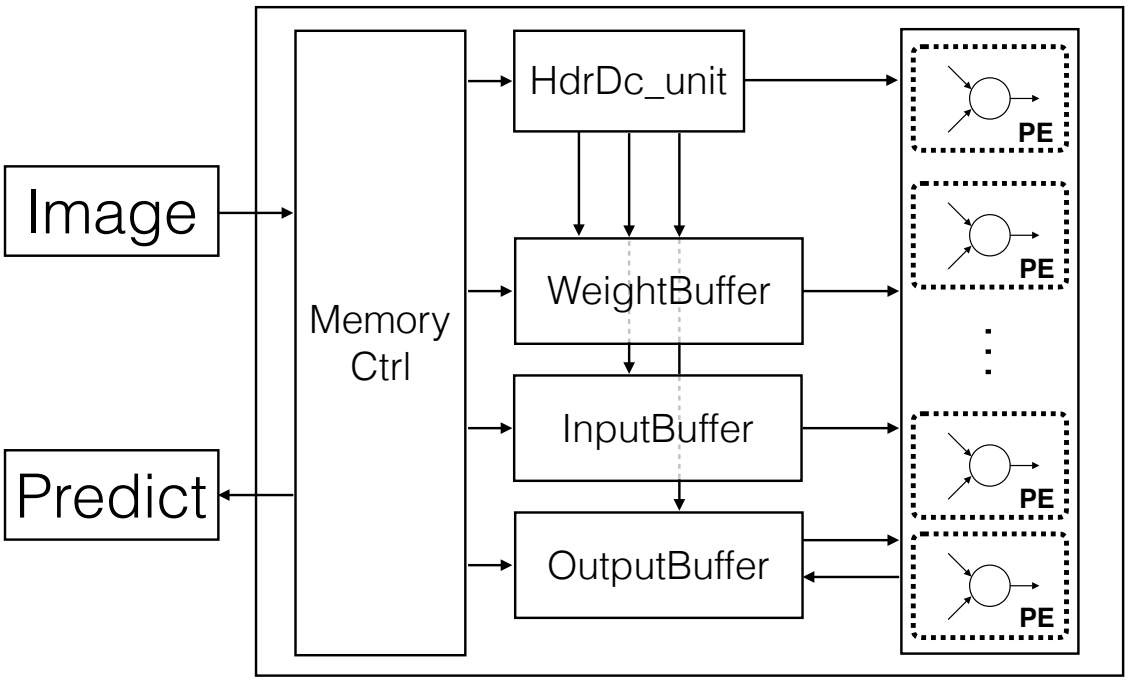


Figure 3.8: Diagram of our DNNs implementation on FPGAs. The limited on-chip memory on FPGAs is not sufficient to hold all the network architecture configurations, so the networks are executed layer by layer. For each layer, the memory controller initiates the data transfer and the layer definition is used by the header decoder unit (HdrDc_unit) to configure the layer. Processing Elements (PEs) consumes data in the WeightBuffer and InputBuffer to compute the output and store it to the OutputBuffer. To hide the data transfer latency, we implement double buffering for all buffers.

account for almost 95% of the total execution cycles. We build our own optimized implementations on an Altera Stratix V platform and detail our implementation of these two algorithms in the following sections.

DNNs on FPGAs Despite the high memory bandwidth and large external memory, the Altera Stratix V platform has limited on-chip memory which is insufficient to hold all the neural network architecture configurations (i.e., the network topology and weights of the neurons) and the intermediate results, because the advanced state-of-the-art DNNs we use have complex and deep network architectures.

An overview of our design is presented in Figure 3.8. Our implementation is capa-

ble of executing all the types of layers used in DET and TRA, including convolutional layers, pooling layers, ReLu layers and fully connected layers. The implementation is composed of four major components: memory controller, buffers, header decoder unit (HdrDc_unit) and the processing elements (PEs). The memory controller first initiates the data transfer between the host device and the FPGA accelerator, and the layer definition (i.e., layer type, weights) is fed to the header decoder unit (HdrDc_unit) to configure the layer. Each buffer stores the corresponding neural network weights, input values, and internal temporary variables until the execution of the layer has been completed. Each PE, primarily consists of multiply-accumulate (MAC) units instantiated by the digital processing processors (DSPs) on the fabric, then performs the necessary computation on the data stored in the WeightBuffer and InputBuffer and writes the output to the OutputBuffer. To hide the data transferring latency, we implement double buffering for all buffers to prefetch the needed data in advance while executing the current layer. Overall, we are able to achieve an utilization higher than 80% on the available adaptive logic modules (ALMs) and DSPs.

FE on FPGAs Feature extraction (FE) dominates the computation time in LOC, so we focus on porting FE when porting to FPGAs. There are two major components in the algorithm ORB we use: oFAST that extracts the feature points and rBRIEF that computes a descriptor for each feature point. An overview of our design is summarized in Figure 3.9.

For oFAST, we implement an image buffer (ImgBuffer) and a feature point buffer (FtrPntBuffer) using shift register, and the mask window is assigned to the corresponding register. As the input data streaming into the buffer, they are filtered by the mask window so the feature detector only receive the data it needs. Orient_unit is implemented with an `atan2` Lookup Table (LUT), to avoid the extensive use of multipliers and dividers of computing `atan2` naively.

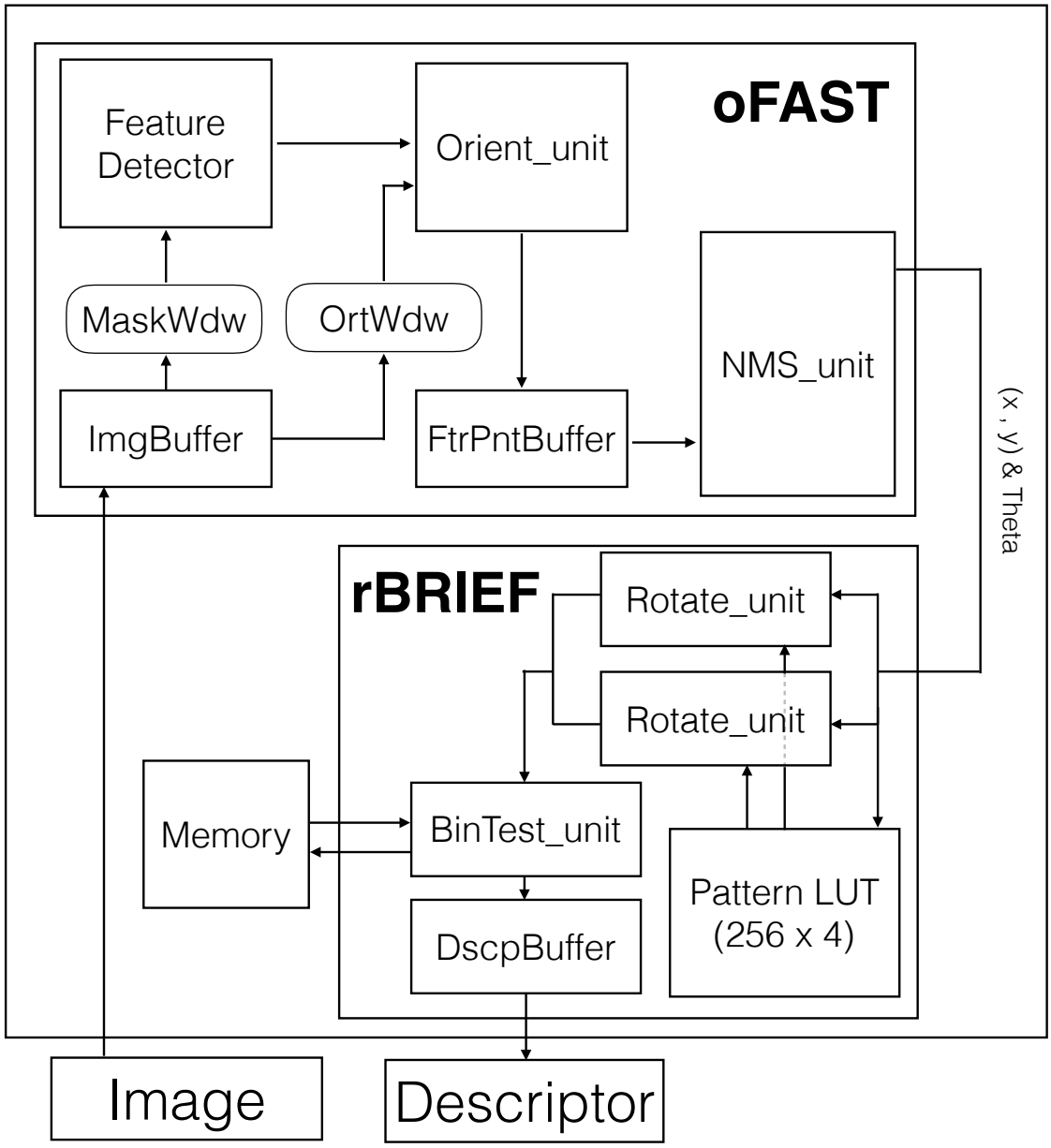


Figure 3.9: Diagram of our implementation of Feature Extraction (FE) on FPGAs. As the input images streaming into the image buffer (ImgBuffer), they are filtered by the mask window (MaskWdw). The feature detector extracts the features of interest and store them into the feature point buffer (FtrPntBuffer). It is then consumed by the rotate_unit to rotate the corresponding coordinates, and the generated feature descriptors are stored into the descriptor buffer (DscpBuffer). To optimize the performance of our design, we implement all the complex trigonometric functions with Lookup Tables (LUTs) to avoid the extensive use of multipliers and dividers, which reduces the latency by a factor of 1.5×.

Table 3.3: Feature Extraction (FE) ASIC specifications.

	Specification
Technology	ARM Artisam IBM SOI 45 nm
Area	6539.9 um^2
Clock Rate	4 GHz (0.25 ns/cycle)
Power	21.97 mW

For rBRIEF, we store the pattern information in the pattern LUT on-chip. Rotate_unit is implemented to rotate to the corresponding coordinates. Similarly to how we implement `atan2` for oFAST, we implement `sin` and `cos` functions using LUTs to avoid the extensive use of multipliers and dividers. Due to the limited on-chip memory available, we execute one binary test at a time and store the result into the descriptor buffer (DscpBuffer) iteratively. As a result, 256 iterations are required to complete one feature point description. However, because of the simplicity of this design, we can achieve high clock rate and thereby low latency. By synthesizing on real systems, we demonstrate our FE implementation can execute at a frequency of 250MHz on the Stratix V development board. By implementing complex trigonometric functions with LUTs, we improve the performance of FE by a factor of 1.5×.

3.3.2.3 ASIC Implementation

We employ previously published ASIC implementations for DNNs [18, 55]. Due to the limited published ASIC implementation of feature extraction (FE), we implement our FE ASIC design and synthesize it with modern technology.

We implement FE ASIC following similar design as our FPGA implementation described in Figure 3.9 in Verilog and synthesize it using ARM Artisam IBM SOI 45 nm library. We verify the result and evaluate the design with post-synthesis simulation. Table 3.3 shows the details of of FE ASIC implementation, where we are able to achieve a clock frequency as high as 4 GHz. This is largely due to the simplicity of our design, as well as the optimized design with re-timing pipeline. Despite the

more cycles needed for the entire pipeline, we achieve better performance comparing to more complex implementations we previously experimented with. Additionally, we are able to achieve a 4× reduction in latency by replacing complex trigonometric function computations with LUTs.

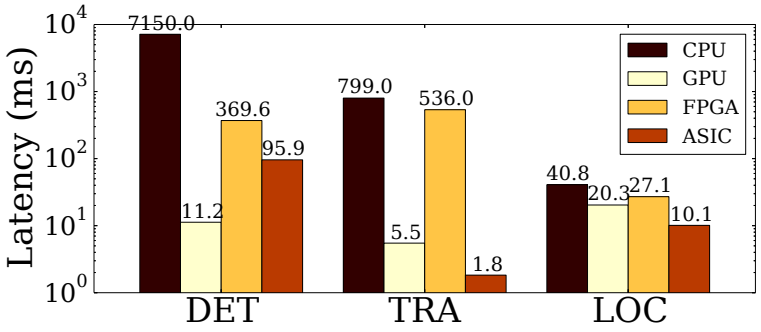
3.4 Evaluation

In this section, we conduct thorough evaluations of various acceleration platforms to explore the design landscape. We focus on the three computational bottlenecks identified in Section 3.2.2 as they constitute more than 94% of the end-to-end execution. In particular, we would like to answer the following questions in this section:

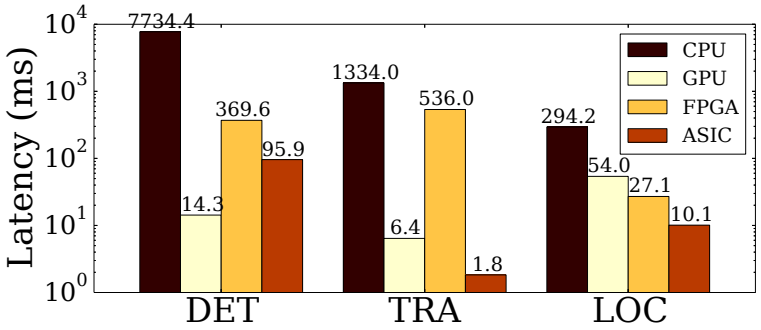
- How much speedup can these accelerators achieve for autonomous driving (Section 3.4.1)?
- Can accelerator-based autonomous driving systems meet the performance and predictability constraints (Section 3.4.2)?
- How does the power consumption of accelerator-based autonomous driving systems affect the vehicle (Section 3.4.3)?
- How scalable are such systems regarding the growing camera resolutions (Section 3.4.4)?

3.4.1 Acceleration Results

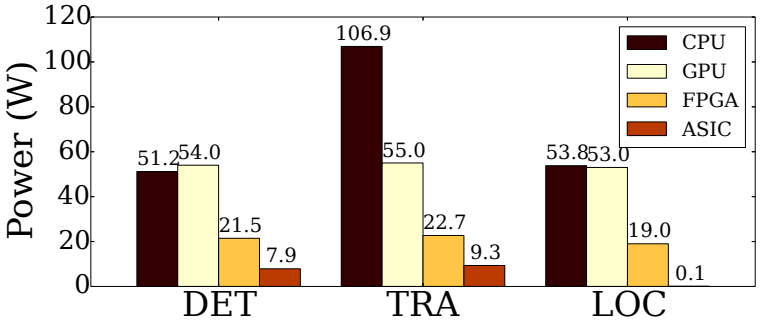
Figure 3.10 presents the acceleration results we are able to achieve across mean latency, 99.99th-percentile latency and the measured power consumption on the four computing platforms we investigate. We measure the performance and power on real systems for CPUs, GPUs, and FPGAs using Watts Up power meter [66]. For ASICs, we reference prior work [18, 55] for performance and power measurements for



(a) Mean Latency Across Platforms.



(b) 99.99th-Percentile Latency Across Platforms.



(c) Power Consumption Across Platforms

Figure 3.10: Acceleration results across various accelerator platforms. The latency of running DET or TRA alone on CPUs or FPGAs has already exceeded the end-to-end latency constraints, which suggests they are not viable candidates for running the complex DNN-based DET and TRA algorithms that demand large amount of computational resources.

the two DNN-based algorithms object detection (DET) and object tracking (TRA), and extrapolate them based on the amount of processing units needed. For our own ASIC implementation of the feature extraction (FEs) algorithm, we use the power estimated in the post-synthesis result and simulate the performance with our post-synthesis module.

3.4.1.1 Mean Latency

As shown in Figure 3.10a, it is impractical to run either DET or TRA on the multicore CPU systems, as the latency of each individual component is already significantly higher than the end-to-end system latency constraints (i.e., 100 ms). This is because both of these components are using DNN-based algorithms, which demands large amount of computing capacity that conventional multicore CPUs does not offer. On the contrary, GPUs provide significantly lower mean latency across all three workloads benefiting from the massive parallel processing power provided by the large number of processors. Although FPGAs achieve significant latency reduction comparing to CPUs, their mean latency for DET (i.e., 369.6 ms) and TRA (i.e., 536.0 ms) are still too high to meet the latency constraints at 100 ms. This is largely due to the limited number of DSPs available on the fabric. To support these complex DNN-based algorithms, large amount of DSPs on FPGAs are needed to provide significant compute power, which can be achieved by much advanced FPGAs (e.g., Xilinx VC709 FPGA board [181]). As we expected, ASICs can achieve significant latency reduction, where the mean latency for executing TRA is as low as 1.8 ms. Note the reason why DET runs slower on ASICs than GPUs is because of the limited clock frequency at 200 MHz this particular design can operate at, which does not preclude similar designs with high clock frequencies to outperform GPUs.

Finding 1. *Multicore CPUs are not viable candidates for running object detection (DET) and object tracking (TRA), which are composed of complex DNN-based algo-*

rithms that demand large amount of computational resources. The limited number of DSPs becomes the main bottleneck preventing FPGAs from meeting the performance constraints.

3.4.1.2 Tail Latency

Figure 3.10b presents the 99.99th-percentile latency across four platforms. As we can see from the figure, although multicore CPUs can execute the localization algorithm within the performance constraints on average (i.e., mean latency), they suffer from high tail latency across all three workloads. This empirically demonstrated our observation in Section 3.1.4.2 that due to its large performance variability, tail latency should be used to evaluate the performance of autonomous driving systems to meet the performance predictability constraints. The other computing platforms do not experience any significant increase from mean latency to the tail latency, which is highly preferable for such mission-critical real-time applications.

Finding 2. *Due to the large performance variability of localization algorithm, tail latency should be used to evaluate the performance of autonomous driving systems to meet the real-time constraints, whereas conventional metrics like mean latency can easily cause misleading conclusions.*

3.4.1.3 Power Consumption

We present the power consumption across 4 platforms for each algorithmic bottleneck in Figure 3.10c. Note that these measurements focus on the computing system only and do not count the power consumption of the storage engine or the impact of the thermal constraints. The measurements are taken for a single camera, where the end-to-end system consists of multiple cameras (e.g., eight for Tesla [159]) and each camera is paired with a replica of the computing engine to be able to process camera streams from different angles. We will discuss the end-to-end system power

consumption in Section 3.4.3.

As shown in the figure, specialized hardware platforms like FPGAs and ASICs offer significantly higher energy efficiency comparing to general-purpose platforms such as conventional multicore CPUs and GPUs. For instance, running DET on ASICs reduces the power consumption by almost a factor of 7 comparing to CPUs and GPUs. Although the measured power consumption of the computing engine for individual camera is relatively low, remember that the computing engines need to be replicated to handle multiple camera streams and the storage engine and thermal constraints will heavily magnify end-to-end system power as mentioned in Section 3.1.4.5. As we will demonstrate empirically in Section 3.4.3, the choice of accelerator platform has a significant impact on the vehicle driving range and fuel efficiency.

Finding 3. *Specialized hardware like FPGAs and ASICs offers significantly higher energy efficiency comparing to conventional general-purpose platforms like multiple CPUs and GPUs for autonomous driving tasks.*

3.4.2 End-to-End Performance

We then investigate the end-to-end system performance of these accelerator-based autonomous driving system designs. Figure 3.11 presents the mean latency and the 99.99th-percentile latency of the end-to-end system across different configurations. The x-axis denotes the configuration, where the color of each grid represents the computing platform each algorithmic component is running on (e.g., a red dotted box on the x-axis represents running LOC on ASIC). For example, the 2nd left-most set of bars in the figure represents the latency of running DET and TRA both on GPUs, and LOC on CPUs. Note the end-to-end latency is determined by the slowest path between LOC and DET + TRA, because they are executed in parallel.

We observe in the figure that certain configurations (e.g., LOC on CPUs, DET and TRA on GPUs) can meet the performance constraints at 100ms latency when

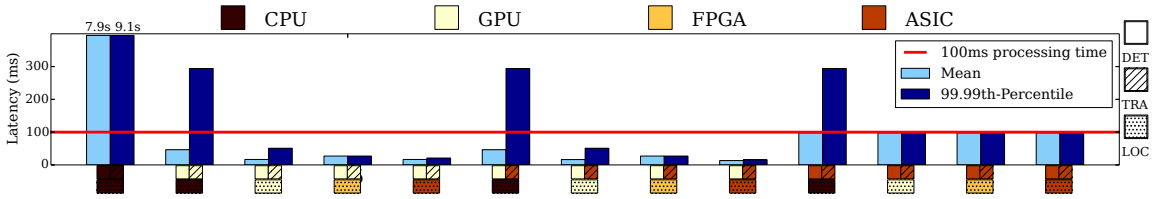


Figure 3.11: The mean and 99.99th-percentile latency of running different algorithmic components across different configurations denoted on x-axis. For each configuration, the color of each grid represents the computing platform each algorithm is running on (e.g., a red dotted grid represents running LOC on ASICs). There are several configurations that meet the performance constraints at tail latency of 100 ms, which means accelerator-based designs are viable for autonomous driving systems.

mean latency is considered, but are no longer viable when considering the tail latency. This again confirms our observation that tail latency should be used when evaluating autonomous driving systems. In addition, none of the viable designs has multicore CPUs due to the inherent non-determinism and unpredictability. With acceleration, we are able to reduce the end-to-end tail latency from 9.1s (i.e., on multicore CPUs) to 16.1ms to meet the real-time processing constraints.

Finding 4. *Accelerator-based design is a viable approach to build autonomous driving systems, and accelerator platforms with high performance predictability (e.g., ASICs) are preferable to meet the real-time processing constraints.*

3.4.3 Power Analysis

In this section, we investigate the power consumption of the end-to-end accelerator-based autonomous driving systems and quantify the corresponding impact on the driving range and fuel efficiency of the vehicle. The results are evaluated based on a Chevy Bolt [19]. As mentioned in Section 3.1.4.5, the system power consumption includes the both computing engine and the storage engine, and will then be magnified by the required cooling capacity to remove the additional heat. We assume the system needs to store the map of the United States (i.e., 110 W power for 41 TB storage space), and is equipped with 8 cameras (i.e., the same as Tesla [159]) each

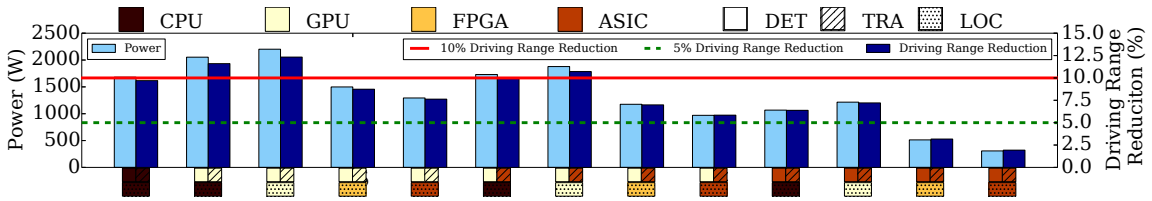


Figure 3.12: The power consumption and the corresponding driving range reduction of running different algorithmic components across different configurations. Configurations equipped with GPUs consume significant amount of power and reduce the driving range up to 12% while ASICs approaches can achieve efficiency which only reduce the driving range by 2%.

connecting to a replica of the computing system.

Figure 3.12 presents the end-to-end power consumption of the same set of accelerator-based autonomous driving system configurations as Figure 3.11, where we use the same notation on x-axis to denote the system configurations. The light blue bars and left y-axis show the total power consumption of the end-to-end system, and the dark blue bars and right y-axis show its corresponding driving range reduction. While mentioned in Section 3.4.2 that powerful accelerators like GPUs can deliver the computation at low latency, it is shown in the figure that most of the configurations equipped with GPUs draw large amount of power (i.e., more than 1,000 W), which results in significant reductions in the driving range of the vehicle. In particular, performing all computing tasks on GPUs can reduce the vehicle driving range by as much as 12%, which dilutes the benefit of using GPUs. To minimize this negative impact, specialized hardware like FPGAs and ASICs are needed to reduce the driving range reduction to within 5%.

Finding 5. *While power-hungry accelerators like GPUs can deliver the computation at low latency, the driving range of the vehicle can be significantly reduced by as much as 12%, largely due to the magnifying effect of the thermal constraints in such systems. Specialized hardware like FPGAs and ASICs are needed to restrain the impact under 5%.*

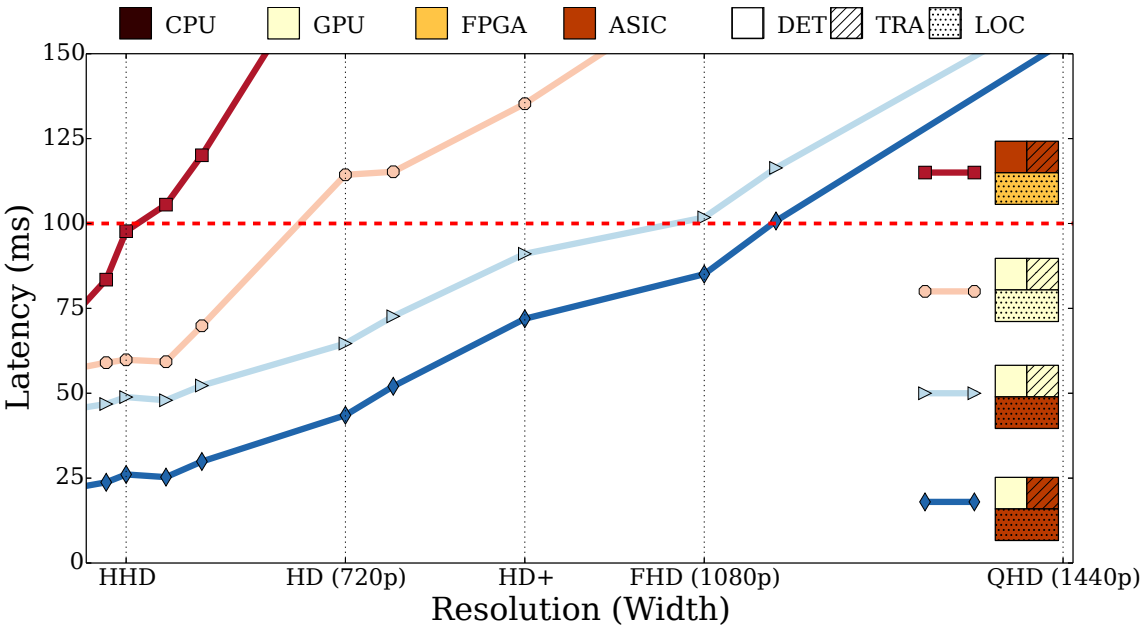


Figure 3.13: Performance scalability regarding camera resolutions of various configurations. Although some configurations can meet the design constraints at Full HD (FHD) resolution, none of them can sustain higher resolutions like Quad HD (QHD). This suggests computational capacity still remains the bottleneck that prevents us from achieving the higher accuracy enabled by higher resolution cameras.

3.4.4 Scalability Analysis

Besides the processing latency, the performance predictability of autonomous driving systems is also determined by the functional aspects – the accuracy of making the correct operational decisions. As demonstrated by prior work [6], increasing camera resolution can significantly boost the accuracy of the autonomous driving systems by sometimes as much as 10%. For example, doubling the input resolution can improve the accuracy of VGG16, a DNN-based state-of-the-art object detection algorithm, from 80.3% to 87.4%. Therefore, we investigate the system scalability of our accelerator-based autonomous driving systems in supporting future higher resolution cameras. We modify the resolution of the benchmarks to study this question, and present the end-to-end latency as a function of the input camera resolution in Figure 3.13 of various accelerator-based configurations. As we can see from the figure,

although some of the ASIC- and GPU-accelerated systems can still meet the real-time performance constraints at Full HD resolution (1080p), none of these configurations can sustain at Quad HD (QHD).

Finding 6. *Computational capability still remains the bottleneck that prevents us from benefiting from the higher system accuracy enabled by higher resolution cameras.*

3.5 Summary

In this Chapter, we first present and formalize the design constraints in performance, predictability, storage, thermal and power when building autonomous driving systems. To investigate the design of such systems, we build a representative end-to-end autonomous driving system using state-of-the-art machine learning algorithmic components. Using this system, we then identify three computational bottlenecks, namely localization, object detection and object tracking. To design a system that meets all the design constraints, we explore three different accelerator platforms to accelerate these computational bottlenecks. We show that GPU-, FPGA-, and ASIC-accelerated systems can reduce the tail latency of these algorithms by 169×, 10×, and 93× respectively. Based on these accelerated system designs, we further explore the tradeoffs among performance, power and future scalability of autonomous driving systems. We find that while power-hungry accelerators like GPUs can predictably deliver the computation at low latency, their high power consumption, further magnified by the cooling load to meet the thermal constraints, can significantly degrade the driving range and fuel efficiency of the vehicle. We also demonstrate that computational capability remains the bottleneck that prevents us from benefiting from the higher system accuracy enabled by higher resolution cameras.

CHAPTER IV

Accelerating Object Recognition for Streaming Videos

Recent advances in computer vision especially using deep learning algorithms have made it feasible to leverage automatic video understanding to facilitate meaningful offline analytic on collections of videos as well as assisting human beings like autonomous driving. However, such algorithms, such as Faster R-CNN, are quite computationally expensive, making it extremely difficult to process massive volumes of videos or perform any real-time tasks. Prior work in improving the latency of such applications often comes at the price of reduced overall accuracy, which makes them a lot less promising for mission-critical applications like autonomous driving.

In this Chapter, we propose a lossless technique on Faster R-CNN to improve the processing performance of video understanding without reduced accuracy by exploiting computation reuse opportunities. Based on our observation that the majority of the pixels do not change across adjacent frames in continuous videos, we can reduce the amount of computation required for each frame via only processing sub-regions changed and reusing existing intermediate results from previous frames for regions remained same. Unlike prior work that requires to sacrifice a certain amount of accuracy to achieve the performance improvement, which makes them less attractive for real-world applications like autonomous driving, our approach is able to reduce

redundant computation to improve the overall performance while still providing the same accuracy. In our evaluation in KITTI, ImageNet VID and Cityscapes benchmarks, we are able to achieve up to 1.27× speedup with nearly lossless accuracy (i.e., $< 1\%$).

4.1 Characterizing State-of-the-Art Object Recognition

In this section, we first review the architecture of one of the most decent object recognition algorithm, namely Faster R-CNN, then we characterize Faster R-CNN to understand the computational profile and identify the potential opportunities for acceleration.

4.1.1 Faster R-CNN Review

Faster R-CNN [136] is originally proposed to solve object detection problems, which take images as input, and identify classes of objects and corresponding positions in images. Region proposal network (RPN) is introduced in Faster R-CNN to address the computation bottlenecks encountered by prior works [46, 63]. The entire framework consists of three key components: (1) feature extraction network, (2) region proposal network, and (3) classifier network, as shown in Figure 4.1 (left). Images received from camera or memory are first sent to feature extraction network to extract a set of feature maps. Region proposal network then takes those feature maps to identify the coordinates of potential regions (i.e., region proposals) and corresponding objectiveness scores. For each region proposal, features within the region are first pooled into fixed size feature maps (i.e., ROI pooling [46]) then object class probabilities are computed and corresponding detection boundaries are regressed by the classifier network.

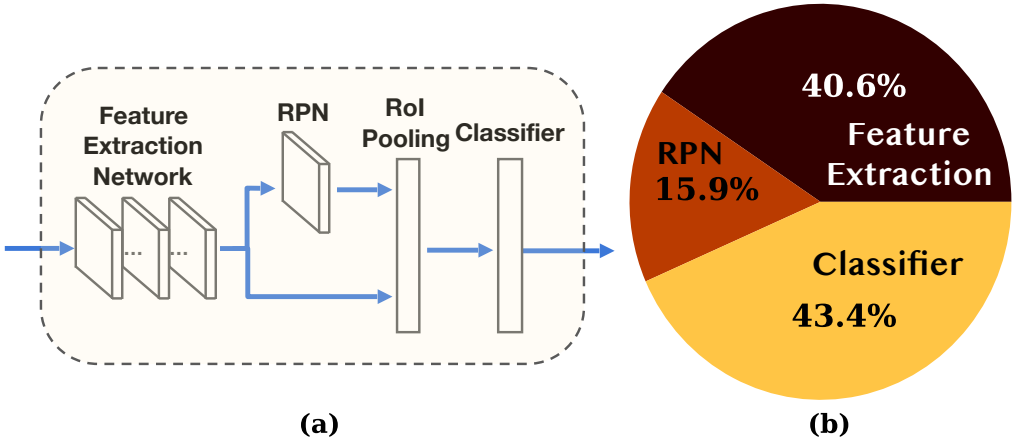


Figure 4.1: Faster R-CNN framework overview (left) and the computational profile breakdown (right). The entire architecture consists of three types of networks: (1) feature extraction network, (2) region proposal network, and (3) classifier network. The cycle breakdown of the end-to-end execution time is shown in the right figure. Feature extraction network and classifier network dominate the end-to-end latency (i.e., over 84% execution time) and thereby are the computational bottlenecks and good candidates for further acceleration.

4.1.2 Characterization

To understand the computational profile and identify the potential bottlenecks in Faster R-CNN architecture, we use Tensorflow Object Detection API [68] to characterize Faster R-CNN and evaluate the processing profile with KITTI object detection benchmark [45], which composed of almost 15,000 images and a total of 80,256 objects labeled. The characterization is conducted on a TitanX Pascal GPU and the breakdown of the measured execution time on the entire pipeline is shown in Figure 4.1 (right). The latency contributed by feature extraction network and classifier network individually dominate the end-to-end execution time (i.e., over 84% of execution time spent in these two network). Therefore we conclude feature extraction network and classifier network as the computational bottlenecks of the full pipeline and design acceleration method to improve the performance of these two components.

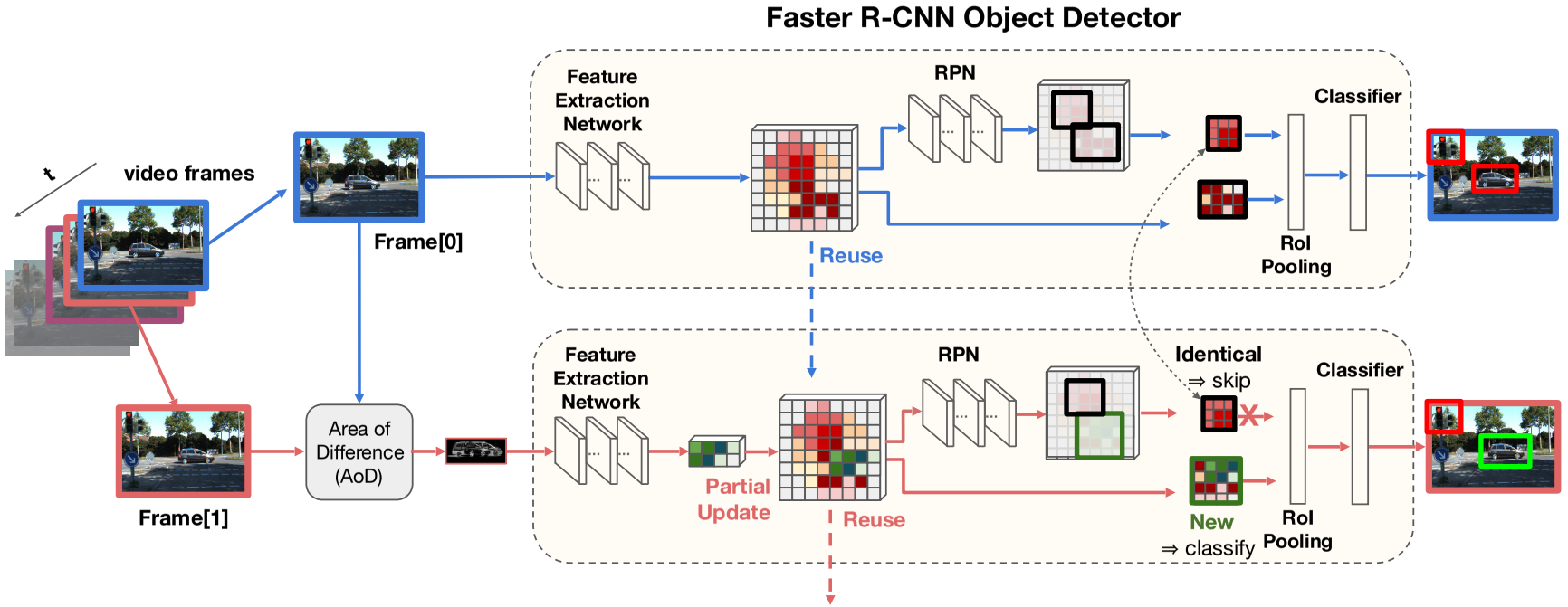


Figure 4.2: Overview of our proposed method, which is built based on Faster R-CNN framework [68, 136]. The first frame is passed to Faster R-CNN pipeline (top), which generates all the intermediate feature maps, region proposals and the final detection results. For the following frame (bottom) in the video, the Area of Difference (AoD) is first generated by performing frame difference between current frame and previous frame. AoD is then sent to feature extraction network to generate partial feature maps, which are updated to the preserved feature maps. We perform the same computation on the entire feature maps in region proposal network (RPN). In the classifier network, we only process new proposals generated from RPN and update the final detection results.

4.2 Proposed Method

Our goal is to analyze the acceleration opportunities in Faster R-CNN architecture for videos. The key challenge is to achieve processing improvement without causing accuracy degradation. Also, employing complex methods introduces significant computation overhead, and ends up with limited or even worse processing performance. We propose two computation reuse methods which enable us to achieve significant performance improvement without sacrificing accuracy, which is shown in Figure 4.2.

4.2.1 Feature Reuse

When applying Faster R-CNN on a video, which can be considered as sequences of consecutive frames, we keep processing each frame by executing the same procedure including feature extraction, region proposal and classification. Recall that in the feature extraction network stage, we extract a set of 2D features from the original image by performing sequences of convolutional computation, where a set of convolving filters are applied on the input image. However, the context between two consecutive frames might be highly similar and thus share alike spatial features, which suggests that a large portion of feature maps used for region proposal network might be identical and only small portion of them provides new information. Extracting new features from entire frames repetitively prevents us from utilizing the features obtained from previous frame and thus leads to redundant computation executed.

To address this issue, we propose to reuse features processed in previous frames and apply feature extraction only on the differences of two adjacent frames. As presented in Figure 4.2, we process the first image entirely and obtain the detection outcomes as well as its intermediate outputs. The image and all of its intermediate outputs (e.g., feature maps) will be preserved for future updating. For the following frame, we first perform frame differencing to obtain the area of difference (AoD) between frames. In the frame differencing stage, we calculate the per-pixel absolute

difference between two images and perform a binary thresholding then to generate a better quality of frame difference results and avoid noises in images. The color in each individual pixel will be set to white if it is higher than the binary threshold, and black otherwise. The contours of frame difference results are then identified and considered as the area of difference (AoD) between the current frame and the previous frame, which are used as the inputs for the feature extraction network. Note that AoD represent the region that is different between two consecutive frames and also the new spatial information that Faster R-CNN needs to understand and detect. We consider the extreme outer contour as the region that will be used and the size of AoD depends on the differencing result. We then perform feature extraction on the AoD to generate associated feature maps, which takes less processing latency since smaller inputs are fed. Finally, the corresponding region of previous feature maps is updated with the new feature maps extracted from the current frame and considered as the complete feature maps for the region proposal network. To conclude, instead of processing the entire frame, we identify the differences between two frames and only compute AoD to extract new features needed for the current frame, which results in significant performance improvement in the feature extraction network stage and we present the result in Section 4.3.2.

4.2.2 Region Reuse

In the previous section, we proposed an approach to reuse the preserved feature maps from previous frame and update it with partial feature maps extracted from area of differences (AoD) in the feature extraction network stage. We now explore the acceleration opportunity in the classifier network stage, which is the other computational bottleneck observed in our characterization. As we discussed in Section 4.1, for each region proposal generated from region proposal network, features within the region are first pooled into fixed size feature maps and then the classifier network

iteratively computes the class probabilities of objects and the associated detection boundaries are also regressed to generate the final positions. The number of region proposals is usually large (typically between 100 to 300) [68] in order to propose good quality of regional boxes and achieve decent accuracy for object detection. Computing these proposals iteratively causes a significant overhead and lead to the key bottleneck in classifier network.

To address this issue, we propose to reduce iterations of classifying proposals by identifying identical ones and reuse the results preserved in the previous execution in the classifier network stage. As presented in Figure 4.2, we use the region proposal network to process feature maps entirely to obtain a complete list of region proposal boxes, which is then preserved for the following comparison and reuse. With the high spatial similarity observed between two consecutive frames in videos, a portion of region proposals generated by region proposal network are usually similar or even identical. After generating the proposals from the following frame, we examine them with the preserved region proposals from the previous frame and pinpoint those differences. We then send only those different proposals (i.e., detection boundaries different from previous preserved region proposals, and the values inside each detection boundary differ from previous proposals) to classifier network to perform classification and regression. All identical (i.e., the size and the position of detection boundaries as well as the values inside detection boundaries are exactly the same) proposals between following frame and previous frame won't be processed again and we directly use the previous classification results for these proposals in the current frame. Last, those region proposals and classification results are updated for the next coming frame. Note that in the region proposal network stage, the entire feature maps are processed entirely to ensure the quality of region proposals and we reduce the amount of iterations in the classifier network stage instead to achieve faster processing performance and avoid repeating similar computation.

4.3 Experiments

In this section, we evaluate the efficacy of our methods on improving processing performance. We first validate the recognition accuracy achieved against baseline, which is detailed in Section 4.3.1. We then examine the processing performance on each of the key components: feature reuse, region reuse with respect to varying input ratios as well as an ablation study in Section 4.3.2. The complete end-to-end performance analysis is finally evaluated in Section 4.3.3.

Dataset - We evaluate our methods on accuracy and processing performance by using the KITTI dataset [45]. KITTI dataset is a representative real-world benchmark dedicated to autonomous driving applications and consists of 21 training sequences and 29 test sequences. We mainly evaluate on the object tracking benchmark since it presents continuous image sequences for us to apply our methods, while the object detection benchmark is composed of over 15,000 independent images. Two classes including car and pedestrian are well-annotated in the benchmark and evaluated in our results. Besides KITTI, we also report our results on ImageNet VID [140], and Cityscapes [26] at the end of the section. ImageNet VID is an extensive video dataset for object detection, which contains 3862, 555 and 937 fully-annotated video snippets in training, validation and test sets respectively. The frame rate is 25-30 fps for most of snippets. We evaluate the end-to-end performance of our method on the test set. Cityscapes is recent released large-scale dataset collected for urban scene understanding and autonomous driving, which contains snippets of street scenes collected from 50 different cities, at a frame rate of 17 fps. The evaluation is performed on the test videos provided for qualitative and quantitative analysis.

Implementation Details - Our implementation is based on Tensorflow Object Detection API [68]. We use the pretrained Faster R-CNN with ResNet-50 model, which is trained on COCO dataset [103], as the baseline and implement feature reuse and region reuse methods. First, we adjust the image resizer in the pre-processing step

in order to process varying size of frames instead of fixed size of frames. We then utilize Tensorflow Graph Editor library [1] to manipulate intermediate feature maps and ROIs in the computation graph created by the pretrained model. We add updating steps after feature maps and ROIs generated in feature extraction network and region proposal network respectively. To update existing feature maps, we implement a mapping process to identify the corresponding positions and update the feature maps for following the region proposal network. We then pinpoint new ROI outcomes comparing to ones generated from previous feature maps by using a hash table to store the outcomes and make the comparison. We keep the top 100 proposals generated from RPN for the object classification. The same Non-Maximum Suppression (NMS) in the baseline is applied to the final detection output for each class separately. Note that we use multicore Intel Xeon E5-2630 v3 CPUs to perform the pre-processing (i.e., identifying AoD) and post-processing steps (i.e., NMS) and TitanX Pascal GPU is employed for feature extraction network, region proposal network and classifier network.

Evaluation Metrics - We conduct the accuracy analysis by using mean Average Precision (mAP), where the public code is provided by KITTI [45]. Three varying difficulties are evaluated including easy, moderate and hard, which are defined based on the size of bounding boxes, occlusion level and truncation level (e.g., moderate level is defined as (1) the height of bounding box: 25 pixels, (2) occlusion level: partly occluded and (3) truncation level: 30%). Since ground truth annotations for the testing set are not released, we evaluate mAP by using the training set with labels provided as our validation set. Note that we use the pretrained Faster R-CNN model provided by Tensorflow Object Detection API [68], which is trained on COCO dataset [103]. We use the same pretrained model for the baseline and our method running on the validation set for the accuracy analysis. For the speedup analysis, we evaluate across three datasets (i.e., KITTI, ImageNet VID and Cityscapes) and

Class	**Car**			**Pedestrian**		
Difficulty	Easy	Moderate	Hard	Easy	Moderate	Hard
Baseline	77.27%	69.47%	61.55%	68.62%	65.08%	54.58%
Ours	76.51%	68.64%	60.92%	68.29%	64.75%	54.40%
Error	0.76%	0.83%	0.63%	0.33%	0.33%	0.18%

Table 4.1: Accuracy (i.e., mAP) analysis results on KITTI benchmark. We use the pretrained Faster R-CNN model [68] trained on COCO dataset [103] as our baseline and apply our methods to validate the recognition accuracy results. We consider car and pedestrian as the target classes followed by [45]. We observe subtle errors (i.e., less than 1%) across easy, moderate, and hard level of difficulties between baseline and our method, which demonstrate that we are able to achieve comparable recognition accuracy while improving the execution time during inferences. See Section 4.3.1 for details.

measure the mean processing latency for inferences on the test set and calculate the geometric mean of the processing latency across videos in each dataset. We then report the performance improvement (i.e., speedup) comparing against baseline. We execute the baseline and our method on each video for 5 iterations then calculate the average processing latency to exclude deviations during runtime.

4.3.1 Accuracy Analysis

In this section, we investigate the accuracy implication when applying our method on the baseline framework across varying classes and difficulties. To evaluate the accuracy, we conduct the detection experiment defined in KITTI benchmark across two types of classes: car and pedestrian and report accuracy results and the associated error. Note that we consider 'car', 'bus' and 'truck' as 'car'-type object in the detection results since they are considered as the same class in KITTI benchmark (i.e., car). The results are presented in Table 4.1 and subtle accuracy difference (i.e., $<$ 1%) observed in the result is mainly caused during the pre-processing stage, in which we set a threshold to filter out insignificant frame differences and refine the quality of AoD for following computation. This pre-determined threshold could be varied de-

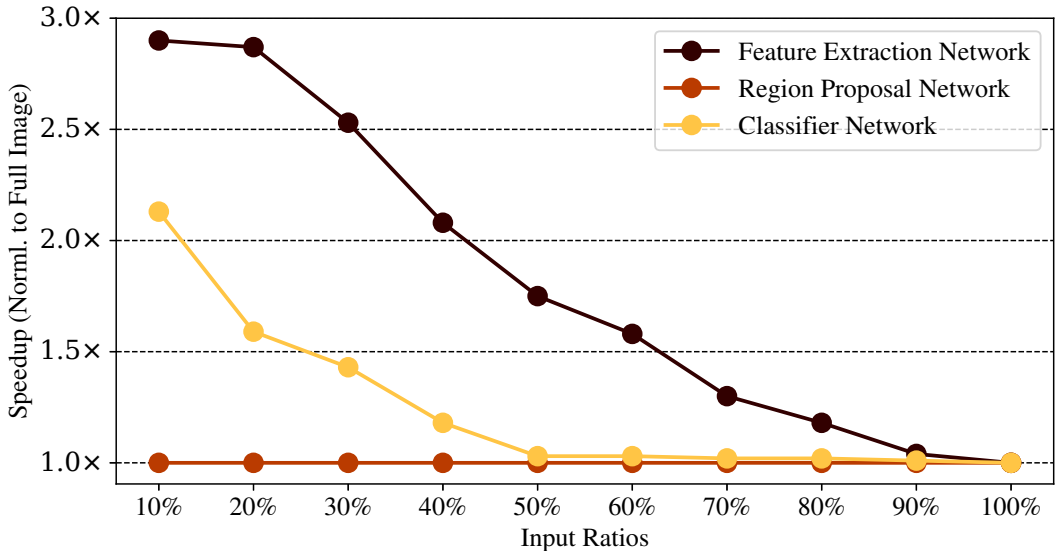


Figure 4.3: Speedup achieved in feature extraction network and classifier network with respect to different ratio of inputs (i.e., the size of AoDs and the number of region proposals respectively), showing that processing smaller inputs improves the execution time effectively. Note that the region proposal network is not accelerated since it is not the key computational bottleneck in Faster R-CNN.

pending on the content of the video and here we present a fixed threshold value and its corresponding accuracy results. Overall, the results demonstrate near identical accuracy can be achieved across varying difficulties when applying our methods on KITTI benchmark.

4.3.2 Acceleration Results

We then investigate the accelerated performance after we demonstrate that nearly no loss in the overall recognition accuracy can be achieved with our methods. The actual speedup that can be achieved by our methods is primarily determined by the size of input ratios. As demonstrated in prior works [102], decreasing the size of inputs processed can significantly accelerate the processing performance since the processing latency is highly proportional to the amount of computation performed (i.e., the size of images). Therefore, we first evaluate the actual speedup achieved

by feature reuse and region reuse with respect to different input ratios, an ablation study under varying AoD ratios is then conducted.

Speedup Achieved by Our Methods - We alter the size of input images for feature extraction network and adjust the number of region proposals for classifier network to evaluate the impact of different input ratios on the speedup. Note that we feed the same size of feature maps (i.e., entire feature maps) for regional proposal network. The speedup is calculated by measuring the mean latency on the real hardware platform (i.e., TitanX Pascal GPU). As shown in Figure 4.3, the input ratios are denoted in x-axis and the associated speedup comparing to the original size is presented in y-axis. The computation latency can be accelerated by up to 2.9× in feature extraction network and over 2.2× speedup can be achieved in classifier network, which suggests that applying our methods indeed improves the execution performance in Faster R-CNN effectively.

Ablation Study - Figure 4.4 presents the end-to-end speedup across different AoD ratios, where we use the same notation on both x-axis and y-axis in Figure 4.3 to denote the speedup and AoD ratios. The black bars (left of three bars) show the speedup when applying feature reuse only, the red bars (middle of three bars) show the speedup when applying both feature reuse and region reuse, and the light yellow bars (right of three bars) represent the speedup when applying both methods and take data communication overhead (i.e., transferring images between CPU and GPU) into account. While we are able to achieve up to 2.35× when processing 10% size of frame as shown in the figure, it presents cutoff points where it becomes worse when processing larger ratio of AoD. For example, processing over 80% size of frame leads to worse performance due to the pre-processing overhead. Note that this cutoff point varies and changes depending on the pre-processing methods. In this case, we observe that re-processing the entire frame is a better approach when the ratio of AoD is regarded over 80%. In the following experiment, this tradeoff point would be

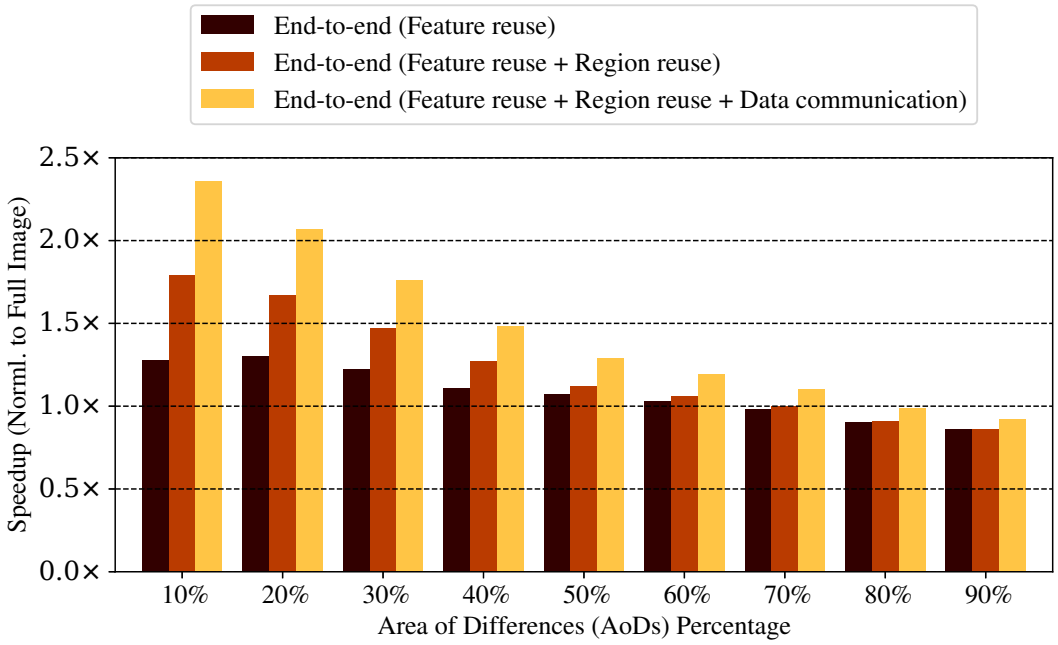


Figure 4.4: Speedup of end-to-end processing latency achieved when applying feature reuse and region reuse, as well as considering data communication cost with respect to different ratio of AoDs. The speedup performance becomes worse in larger ratio of AoD (e.g., over 80% when applying both methods and considering the communication cost) due to the pre-processing cost and results in re-processing the entire frame a better approach.

considered and we compute either AoD or the entire image depending on the ratio to optimize the overall end-to-end performance (i.e., process AoD when the ratio is below 80%, re-process the entire frame otherwise).

4.3.3 End-to-End Performance Analysis

In this section, we investigate the end-to-end performance against baseline. We start by evaluating the impact of ratio of AoD on the end-to-end performance. Figure 4.5 shows the AoD ratios (on top) in every frame and the associated end-to-end processing latency running with baseline (black dash line) and our method (yellow line) during a 30 seconds video example. The processing latency of baseline is mostly static over time, which is expected since DNN execution is deterministic given the

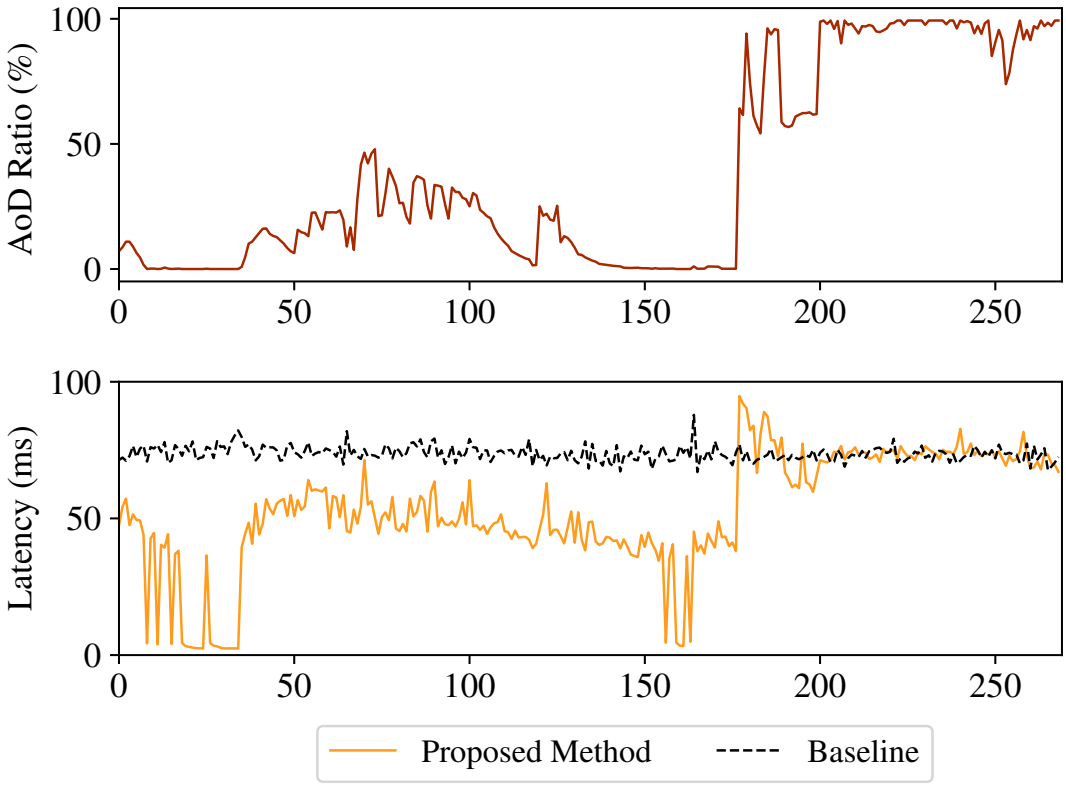


Figure 4.5: AoD ratio (on top) and the associated end-to-end processing latency (on bottom) recorded with baseline (Faster R-CNN alone) and our methods applied running in KITTI benchmark. Our methods are able to achieve significant speedup when small AoD detected, and dynamically switch to process the original frame instead when large portion of AoD detected to meet comparable processing latency as baseline.

same size of images. On the other hand, the processing latency of our method might differ since it depends on the size of images. The AoD ratios vary depending on the content of videos and as shown in the figure, AoD ratios remain in low percentage (e.g., between frame 0 to frame 160), which translate to faster performance as depicted in the yellow line. In addition, our method is able to identify increasing ratio of AoD (e.g., between frame 160 to frame 250) and maintains comparable performance against baseline. We further evaluate the end-to-end performance results in Figure 4.6, in which we present the distribution of AoD ratios, the latency of baseline

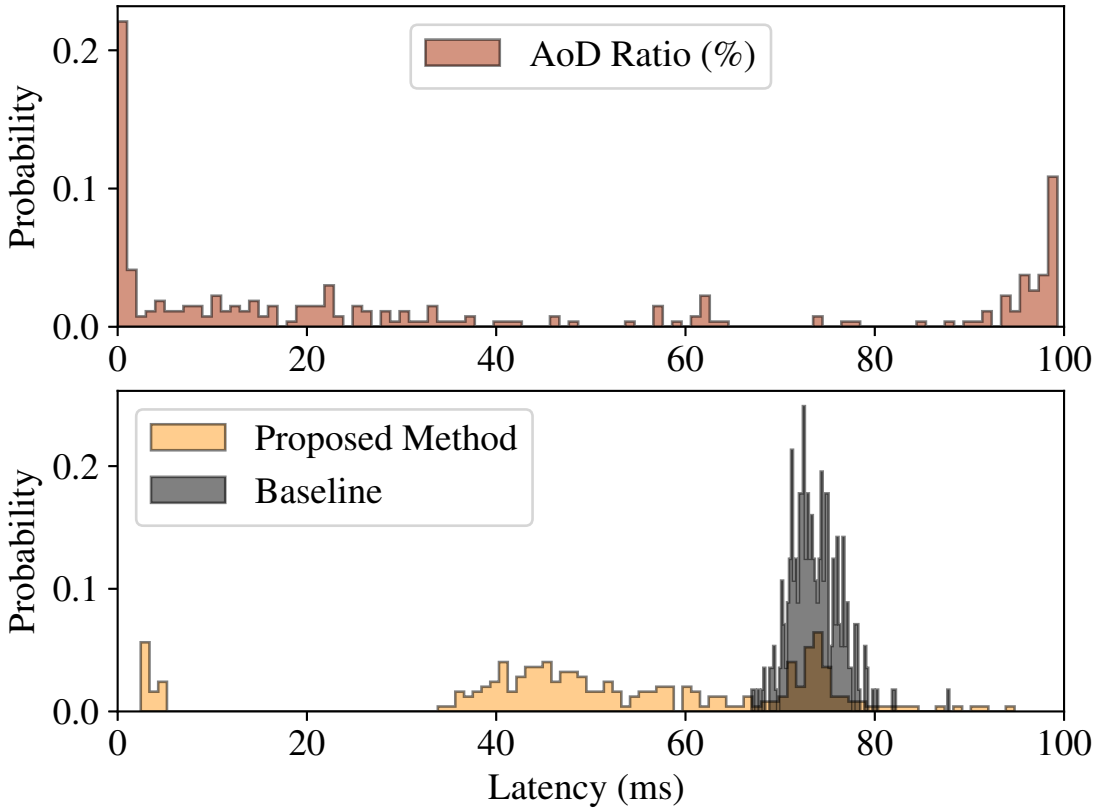


Figure 4.6: AoD ratio (top) and the associated end-to-end processing latency (bottom) with baseline (Faster R-CNN alone) and our methods applied running on KITTI benchmark presented in histogram. The processing latency is significantly improved after applying our methods and even faster performance can be achieved with subtle changes between frames (e.g., <3% AoD ratio).

and our method. We observe that the end-to-end performance has been significantly improved. Specifically, we are able to detect only small portion of images changes in the video and achieve even faster speedup.

Table 4.2 shows the overall performance improvement achieved by our method across three datatsets (i.e., KITTI, ImageNet VID and Cityscapes). We evaluate the test sets in each dataset with both the baseline (i.e., Original Faster R-CNN) and our method. We calculate the geometric mean speedup across all the videos in each dataset (e.g., we calculate the geometric processing latency for both the baseline and our method across 937 videos in ImageNet VID dataset then calculate

Dataset	Performance improvement
KITTI [45]	1.22×
ImageNet VID [140]	1.27×
Cityscapes [26]	1.24×

Table 4.2: End-to-end performance speedup across three datasets (i.e., KITTI, ImageNet VID and Cityscapes). We observe up to 1.27 × processing improvement could be achieved which demonstrates that our method is effective across varying datasets.

the speedup accordingly). Overall, we observe up to 1.27 × speedup across three large-scale datasets. Besides, since the processing performance of our method is highly proportional to the ratio of AoD, we find that under certain scenarios (e.g., left or right turn of a car) the ratio of AoD increases significantly and therefore leads to the amount of computation increases. However, we are able to maintain competitive performance against baseline even though the ratio of AoD is relatively large (e.g., over 80%) since we identify the cutoff point and prevent us from having worse performance against baseline as mentioned in Section 4.3.2.

4.4 Summary

We identify key computational bottlenecks in Faster R-CNN architecture and propose two new computation reuse approaches: (1) feature reuse, and (2) region reuse for Faster R-CNN in videos, which enables us to reuse all the intermediate computed outputs (e.g., feature maps, region proposals and detection results) between consecutive frames and avoid redundant computation. Our proposed methods effectively leverage previous computed outputs and execute DNNs on partial inputs (e.g., AoD and region proposals) to achieve real-time video understanding. Our experiments verified that our methods can substantially improve the processing performance by as much as 1.27× on three large-scale datasets with nearly no loss in the overall recognition accuracy.

CHAPTER V

Adasa: A Conversational In-Vehicle Digital Assistant for Autonomous Driving

Advanced Driver Assistance Systems (ADAS) come equipped on most modern vehicles and are intended to assist the driver and enhance the driving experience through features such as lane keeping system and adaptive cruise control. However, recent studies show that few people utilize these features for several reasons. First, ADAS features were not common until recently. Second, most users are unfamiliar with these features and do not know what to expect. Finally, the interface for operating these features is not intuitive.

To help drivers understand ADAS features, we present a conversational in-vehicle digital assistant that responds to drivers' questions and commands in natural language. With the system prototyped herein, drivers can ask questions or command using unconstrained natural language in the vehicle, and the assistant trained by using advanced machine learning techniques, coupled with access to vehicle signals, responds in real-time based on conversational context. Results of our system prototyped on a production vehicle are presented, demonstrating its effectiveness in improving driver understanding and usability of ADAS.

5.1 Understanding Drivers' Information Needs

To understand what information drivers need and what the critical problems are that prevents them from utilizing ADAS features, we first conducted an in-depth analysis on customer verbatims describing the real problems that the owners of vehicles equipped with ADAS features encounter. These verbatims collected by Ford Customer Service Division (FCSD) via customer call center services include over 9,000 cases provided for analysis consisting of owners of Ford Fusion, Ford Explorer, Lincoln MKS and Lincoln MKT equipped with ADAS features from 2013 to 2017. From these verbatims, we identify three primary question categories:

- **Division of driving responsibility between the driver and ADAS** – More than 4,400 out of the 9,000 (i.e., 48.9%) verbatims ask about the expected division of driving responsibility between the driver and the ADAS feature. For example, drivers complain LKS does not keep the vehicle in its lane after it has been switched on, which is because LKS only engages when the vehicle speed is higher than its operational threshold (i.e., 40 miles per hour) and the lane marking lines are visible.

- **Interface to activate ADAS features** – Over 2,000 cases (i.e., 23.1%) ask how to turn on certain ADAS features (e.g., LKS). This is primarily because drivers are not familiar with these features and they often have a hard time recognizing the buttons to turn them on and off. This demonstrates that there is a gulf of execution that prevents most drivers from utilizing ADAS features.

- **Meaning of instrument cluster iconography** – 1,300 out of 9,000 (i.e., 14.5%) verbatims ask about the symbols on the dashboard because quite a few ADAS features present different symbols dynamically depending on the real-time surrounding environment (e.g., a typical ACC shows a vehicle symbol on

the dashboard only when there is a vehicle ahead) which also shows that most drivers encounter issues of understanding.

One could argue that auto manufacturers provide comprehensive information about these questions in the vehicle owner's manual. However, many drivers are still found to be confused about ADAS features for two reasons, thereby not being able to utilize them in their vehicles. First, it is tedious to read the printed user manuals due to their lengthiness (i.e., often hundreds of pages). Second, these questions should be addressed in real time, as opposed to conventional approaches where drivers could not and should not read user manuals while driving. This is because only 3.16% users actually read the printed user manual when they encounter issues like this, as reported by prior work [124]. Feedback from the Ford user experience design team further confirms these observations. Expecting all users to carefully read the owner's manual may be overly optimistic, and these 9,000 verbatims we analyzed have demonstrated that to be the case. Therefore, we conclude an improved and more natural user interface is required to improve the usability of ADAS features.

5.2 System Design

To enhance the current user interface for ADAS features and improve its usability, we present **Adasa**, an in-vehicle digital assistant based on the insights we gained from our analysis of over 9,000 real user verbatims. Adasa allows the driver to ask any question or command and control in natural human language while driving, and the assistant analyzes the human speech in real-time and combines it with the contextual information about vehicle status (e.g., what mode ACC is currently operating in) to provide responses accordingly in human speech.

In this section, we first formalize the design objectives for Adasa. We then present an overview of the components in Adasa, and walk through the life of an example

query to illustrate the workflow of the system. Lastly, we present the details of the key hardware apparatus and software components.

5.2.1 Design Objectives

To improve the system through real-world testing and subjective evaluation beyond the insights we gained from analyzing the call center verbatims, we formalize the following items as the key design objectives of an in-vehicle digital assistant:

1. **Intelligent understanding** – The digital assistant is able to understand incomplete questions and identify out-of-scope queries properly.

2. **Real-time processing** – The digital assistant has the ability to process queries and respond with the requested information to drivers in real-time in a dynamic driving environment.

3. **Accurate responses** – The digital assistant can provide useful information regarding driver's questions and control the vehicle correctly.

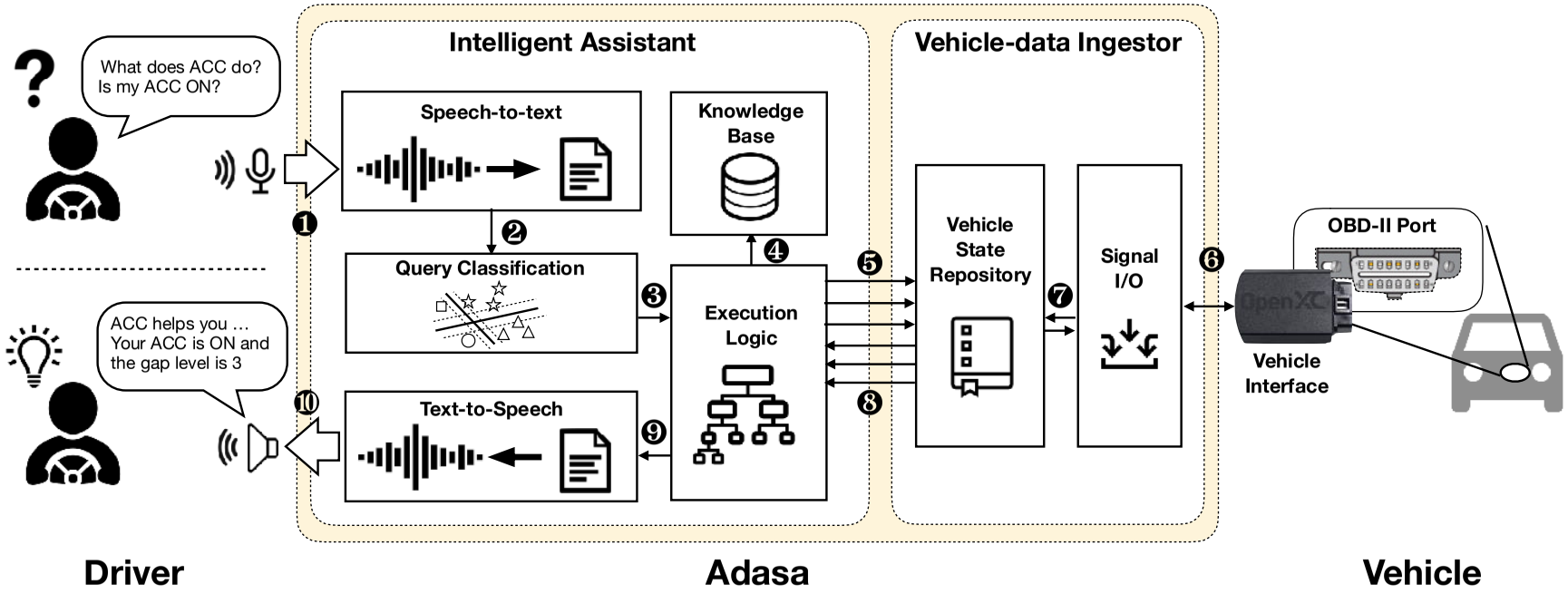


Figure 5.1: Overview of Adasa system, which is designed based on recent publications [61] and integrated with a commercially available vehicle [25]. Driver's voice captured by the microphone is translated into its text equivalent (❶) and is passed to Query Classifier to identify the type of query (❷). Execution Logic operates the query (❸) and retrieves the information needed from either Knowledge Base (❹) or the vehicle (❺) depends on the type of query. Vehicle-data Ingestor receives the CAN data through the vehicle interface and records the status of each signal in Vehicle State Repository (❻-❼). Once the result is passed back to Execution Logic (❽), the response is converted to a WAV file and output to the speaker to answer driver's questions (❾-❿).

5.2.2 System Overview: The Life of a Query

Adasa provides a voice-enabled user interface for drivers to ask questions about and command and control ADAS features using natural human language, without requiring any complex operations that may distract them from driving. The system leverages contextual information of vehicle state and feature status (e.g., what mode ACC is currently operating in) by accessing the signals from the standard OBD-II port. This allows drivers to ask questions like "*what does the green symbol on my dashboard mean?*" without knowing it is related to ACC. In addition, the system is able to send control signals via OBD-II port to enable or disable features to allow drivers to make commands such as "*can you help me turn on adaptive cruise control?*" instead of knowing or asking about and then pressing the activation buttons on the steering wheel. Leveraging the state-of-the-art framework for building intelligent assistant, Adasa is trained to answer a wide range of questions covering the most frequently asked ones we identified in our analysis on real user verbatims from Ford, and is capable of delivering responsive feedback at real-time.

Figure 5.1 presents a high-level diagram of the system components and how user queries are handled in Adasa. Adasa is composed of hardware apparatus and software components. On the hardware side, Adasa consists of a compute device and a vehicle interface device. On the software side, Adasa consists of an Intelligent Assistant runtime, and a Vehicle-data Ingestor runtime.

To illustrate how these components are integrated, we walk through the life of a query step-by-step in Figure 5.1. The life of a query begins with a user's voice input. After the driver activates the system to begin listening by pressing the voice activation button on the steering wheel, the voice input will be sent to the Speech-to-Text engine of Adasa (step 1) deployed on the computing device. This voice input is then transcribed into its text equivalent (step 2). The text of the query is classified by the Query Classifier, where different intents will be given a different label (step 3). For

example, questions like "*what does adaptive cruise control do?*" and "*what is adaptive cruise control?*" will be labeled as the same class because they are both querying about the functionality of ACC, but questions such as "*will lane keeping system steer the vehicle for me?*" will be labeled as a different class. The label generated by the Query Classifier will then be sent to the Execution Logic for further analysis. Depending on the intent label of the query, the Execution Logic needs to either fetch the necessary information from a vehicle-specific knowledge base (step 4), or sends a request to the Vehicle-data Ingestor to obtain/modify the values of certain vehicle signals(step 5). The Vehicle-data Ingestor runs an asynchronous Signal I/O callback which continuously intakes message streams from the vehicle via a Vehicle Interface plugged into the OBD-II port in the vehicle (step 6), and maintains a lookup table containing the updated values of all the vehicle signals internally in its Vehicle State Repository (step 7). Upon receiving a request from the Execution Logic, Vehicle-data Ingestor searches the current values of the requested signals, updates the signals if the request is a command, and otherwise sends the corresponding signals back to the Execution Logic (step 8). Combining all this information, Execution Logic composes a response accordingly in text format, and sends the response to the Text-to-Speech engine (step 9) and produces a WAV file. The resulting WAV file of the response is played back to the driver through the in-vehicle speakers (step 10), informing drivers with the information they requested.

In the following subsections, we describe each of these components in detail.

5.2.3 Hardware Apparatus

5.2.3.1 Vehicle

As shown in Figure 5.2-(B), we employ a 2017 Lincoln Continental Reserve Sedan which is equipped with the Technology Package including ACC. As part of its driver assistance suite, the vehicle is also equipped with LKS, blind spot information system

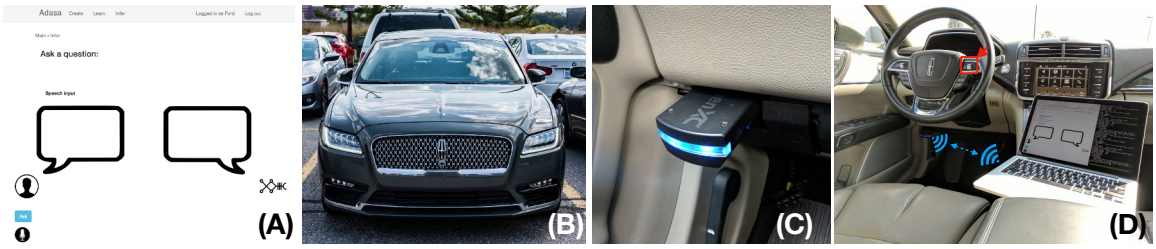


Figure 5.2: (A) The proposed digital assistant, Adasa, identifies user's questions or commands regarding ADAS features in human language and responds with answers or actions accordingly. (B) Adasa is integrated into a commercially available vehicle for evaluation and a real-world driving user study. (C) Adasa is able to access vehicle CAN signals via Bluetooth to conduct system diagnosis or system control. (D) Adasa setup, driver enables Adasa by pressing the button on the wheel to start the conversation.

with cross traffic alert, radars and 360 degree camera technology for pre-collision assist and pedestrian detection [25].

5.2.3.2 OBD-II Port and Vehicle Interface

Our vehicle provides an on-vehicle diagnostic system called On-Board Diagnostic (OBD) system, which gives us access to the internal signals of the vehicle through the Controller Area Network (CAN). OBD-II is an industry standard of such a system implemented in all cars manufactured in the United States from 1996 onward. This standard specifies the pin mapping, the protocol and the message format of the in-vehicle diagnostic connector, and provides engine control, and monitors parts of the body, accessory devices, as well as the diagnostic control network of the car. The interface of OBD-II system (referred as the *OBD-II port* in this work) is usually located within the realm of the driver's seat, underneath the instrument panel or near the footwell. To access the necessary information (e.g., the status of the targeted ADAS features) from the OBD-II port, we plug in a vehicle interface programmed with customized firmware. Vehicle interface is a piece of hardware device that bridges the vehicle and the host device via the OBD-II port. It decodes CAN messages into the software-recognizable format, and sends the decoded results over a common

interface, such as USB and Bluetooth, to the host devices. In this work, we use the widely deployed Ford Reference Vehicle Interface (VI), which is a standard open source hardware implementation of the vehicle interface, as shown in Figure 5.2-(C).

Since the publicly available standard firmware (Type-3 firmware) for a 2017 Lincoln Continental using a Ford Reference VI could not directly access ADAS data, we first reprogrammed and customized the firmware for proprietary access to both LKS and ACC information. An Intrepid Control Systems neoVI is then used to perform additional processing on and writing of CAN signals to allow Adasa to overwrite the vehicle's internal control of ADAS features as part of our prototype. Instead of connecting the OBD VI to our system through a long USB cable in the driver's footwell, which may pose a safety hazard while driving, we enabled a wireless connection between the VI and Adasa via Bluetooth. As shown in the Figure 5.2-(C), Ford Reference VI plugged into the in-vehicle OBD-II port and two blue LEDs show that VI is successfully recognized by and registered to the vehicle as well as connected to Adasa via Bluetooth.

5.2.4 Vehicle-data Ingestor

We built the Vehicle-data Ingestor by using an open-source OpenXC library to communicate using CAN data between the VI and Adasa with the read and write function enabled. To sustain the consistency of the information between the vehicle and our system, we implemented our Vehicle-data Ingestor in a callback fashion (i.e., where the Vehicle-data Ingestor is able to refresh its own data periodically as the VI receives updates from the vehicle). The stream of CAN data is then analyzed by a sub-procedure in the Vehicle-data Ingestor named Vehicle-State Repository. Vehicle-state repository records the most recent history of the data, extracts updates to each signal from the data, and maintains a lookup table that contains the latest status of each signal. In addition, as Adasa is requested to control and change the status of

ADAS features, Vehicle-data Ingestor sends the signal to VI and updates both the lookup table in Vehicle-State Repository, as well as the CAN data inside the vehicle, to ensure the consistency in both sides.

5.2.5 Intelligent Assistant

Intelligent Assistant is designed to understand drivers' questions, process the query coupled with the status of ADAS obtained from Vehicle-data Ingestor, and respond to drivers accordingly. We employ a state-of-the-art speech-based framework, namely Lucida [61], to structure our implementation, including automatic speech recognition and query classification. We describe the details of each component in the following sections.

5.2.5.1 Speech-to-Text Engine

The first major component in the Intelligent Assistant is a speech-to-text interface. This interface allows drivers to ask the questions in natural language, providing drivers an uninterrupted way to interact with Adasa and access the ADAS features. Adasa builds on the open-source Lucida framework [61], which by default uses Google Speech API [154] for automatic speech recognition (ASR). Despite the noisy on-road environment in our user study, we observed very low error rate, which is similar to prior work reported [168]. It first processes and extracts feature vectors representing the voice segments, and submits the feature vectors to a speech recognition kernel to transcribe drivers' utterances. The transcribed texts then serve as the input to the next engine.

5.2.5.2 Query Classifier

To access the necessary vehicle information for each query precisely, Adasa needs to first understand the intent of the query. For this we leverage the findings explored

in our pilot study, and conclude three types of input queries:

1. **Inquiry (FAQ) -** Queries regarding the explanation of the ADAS features are considered as Inquiry (FAQ). For example, "*How does the lane keeping system work?*", "*What is the gray speedometer on my dashboard?*".

2. **Symptom Diagnosis -** Queries regarding the symptoms or status of the ADAS features are considered as Symptom Diagnosis. For instance, "*Is my lane keeping system active?*", "*I just turned on adaptive cruise control, but why is it not working?*".

3. **Command and Control -** Queries regarding enabling, disabling and changing the status of ADAS features are considered as Command and Control. For example, "*Can you turn on lane keeping system for me?*", "*Please increase the gap distance.*".

One straightforward way to classify queries is to hand-assign a class label for every possible utterance, and use a dictionary-like structure to store the mapping between queries and labels. During runtime, class label of an input query is determined by applying string matching between the query and the keys in the key space of the dictionary. However, this approach is impractical in real-world driving scenarios since drivers might not have prior knowledge about ADAS and are unfamiliar with the exact terminology. In addition, natural language speech is imprecise and hard to predict. Especially, utterances can significantly deviate from correct grammar during driving since drivers need to focus on the traffic conditions. These observations show that Adasa should be able to handle incomplete and ambiguous sentences which render the hand-coding approach infeasible.

We address these challenges by employing a machine learning based classifier to automate the process of query classification. To encapsulate a large scope of questions for our query classifier, we collect a large amount of training data by using crowd

Figure 5.3: An Amazon Mechanical Turk (MTurk) task assignment example. We asked the MTurk workers to rephrase the statement, "*what does it mean if I see a red line and a grey line on the dashboard?*" with the picture of the entire dashboard and a red bounding box around the area of inquiry in order to help the workers understand the questions and task at hand.

sourcing on *Amazon Mechanical Turk (MTurk)* [14] via the following steps: First, we analyze the customer verbatims and the feedback described in Section 5.1 to identify the scope of query classes that we have to cover. Second, for each of these classes, we create a task assignment on MTurk, in which we provide a textual description of a driving scenario and a query example to ask under that scenario. We then ask MTurk workers to provide five rephrases of that query. To better contextualize the MTurk workers, we also include a picture of the dashboard of the targeted scenario in the assignment for the workers to gain a better understanding about what they would experience if they were in the vehicle. For example, as shown in the Figure 5.3, we have assignments in which there are questions such as *"what does it mean if I see a red line and a grey line on the dashboard?"* With the picture of the instrument cluster provided, the workers understand the questions precisely and easily. Third, we collect the completed assignment, manually remove the redundant rephrases, and evaluate the quality of the rest of the rephrases. We include only the qualified rephrases in our final training data. Finally, we train our classifier with this training dataset by utilizing support vector machine (SVM) and deploy the trained classifier in Adasa. We use the unigram and the bigram representations of the input query as features,

which is commonly used in text classification. For instance, an input query of "*what is cruise control*" will be transformed into {'what': 1, 'is': 1, 'cruise': 1, 'control': 1, 'what is': 1, 'is cruise': 1, 'cruise control': 1}, where the keys are unigrams and bigrams in the query and the values are their occurrences. We use both unigram and bigram to capture the spatial ordering of words. Overall, over 4500 training data is collected and 73 classes are implemented including 49 Inquiry classes, 11 Symptom Diagnosis classes and 13 Command and Control classes.

Query Classifier outputs a probability distribution for each query, which represents the probability of the query falling into each of the topics. A high probability on one class means the classifier is confident that the query belongs to the corresponding topic. When drivers ask a query that is not covered by the scope the system is trained to understand (e.g., "*how is the weather in San Diego?*"), the output probability distribution will not have any class with a high enough probability (i.e., greater than 0.5). The system identifies such queries as out-of-scope, and respond with "*I am not trained to handle this topic yet. Please ask me about adaptive cruise control and lane keeping systems.*"

5.2.5.3 Execution Logic

Once the input query is analyzed and classified, Execution Logic then accesses the necessary information and answers the query. Depending on the intent label of the query, Execution Logic fetches information from different sources. To compose a response for *Inquiry (FAQ)* queries, we build a vehicle-specific knowledge base that contains the static information about the vehicle and ADAS features. Adasa accesses the target entry and retrieves the corresponding answer.

To answer *Symptom Diagnosis* queries, on the other hand, Adasa needs to access the states of different features of the vehicle. Adasa, based on the targeted ADAS feature and the intent of the diagnostic query, composes a request and sends it to

Vehicle-data Ingestor which decodes the request and seeks for the current status of or a recent update to the targeted signals. Once the requested value is obtained, Vehicle-State Repository returns the result to Execution Logic. We enable non-blocking multi-threaded access from Execution Logic to Vehicle-State Repository, which ensures that Execution Logic can access vehicle states in a timely manner. Finally, Execution Logic applies the returned value and composes a textual response correspondingly.

To complete *Command and Control* queries, Adasa requires to send control signals into the vehicle via VI to alter the ADAS functions. When Execution Logic receives the intent of query from Query Classifier, control signals mapped to the corresponding CAN data messages are generated and transmitted to the VI via Bluetooth and modify the value on the CAN bus inside the vehicle. Also, Execution Logic updates the states of the target signal in Vehicle-State Repository and composes a textual response to indicate that the command is completed.

5.2.5.4 Text-to-Speech Engine

The output textual response is then translated to speech by the text-to-speech module, where Google Chrome Speech Synthesis library is employed [154]. Several attributes such as the voice of gender, speech rate, pitch and volume can be customized for different users. We use standard Google Female English and set rate, pitch and volume as default. The output speech then is played by the speaker.

5.3 Evaluation

To evaluate our system in improving the perceived usability of ADAS features, we conducted a user study in a real-world driving environment. In this study, each participant interacted with Adasa while driving a 2017 Lincoln Continental Reserve equipped with the Technology Package, which includes ACC and LKS. During the driving study, we focused on ACC and LKS, and encouraged participants to ask

Adasa any question, including questions that were beyond the scope of these two ADAS features, so as to obtain complete feedback based on their overall experiences. The study setup is detailed in the following sections.

5.3.1 Participants

We recruited 15 participants: 11 male, 4 female, ages 24—35 (M = 27.1, SD = 3.2). Each participant had a valid US driver's license and was covered by an auto insurance policy sponsored by the university. 80% (12 out of 15) participants did not have prior experience with ADAS, but had sufficient driving experience (M = 8.7, SD = 5.1 driving years). Note that sufficient driving experience is required since participants would be asked to interact with Adasa during the driving study. Participants were recruited through email announcements at the authors' university.

5.3.2 Adasa

As Figure 5.2-(D) shows, we built Adasa and deployed it in the Lincoln Continental described above to conduct the study. Adasa ran on a laptop placed next to the driver seat and connected through Bluetooth to Ford Reference VI to transmit CAN data via OBD-II port. The left blue light on the VI shows CAN data could be accessed successfully and the right blue light shows the data from the VI was received by Adasa. We prototyped a voice interface with a front-end web application, where participants could ask questions via this interface. To enable Adasa and start the conversation, participants were instructed to press the button on the steering wheel to enable voice recording and send the query to the internal modules of Adasa. The laptop would output the answer using human voice through the speaker in the laptop once Adasa retrieves the results.

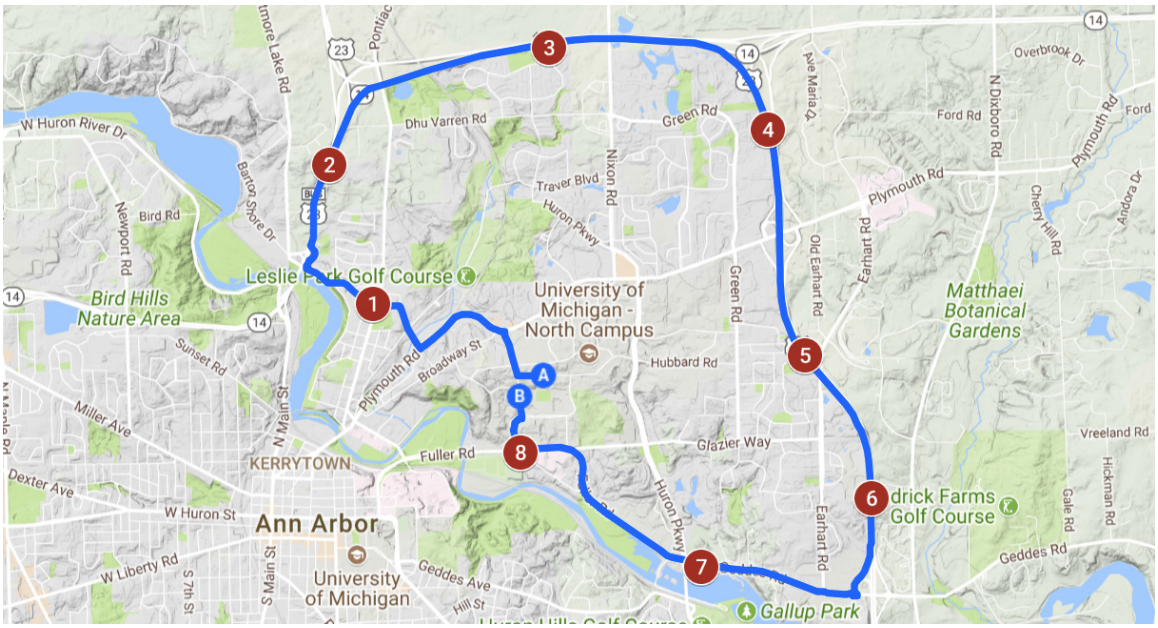


Figure 5.4: The map of the 11.7 miles route. It includes 7 miles highway and 4.7 miles suburban road to accommodate the use of both ACC and LKS.

5.3.3 The Route

Participants drove on a predefined 11.7 mile (18.8 km) route, which consisted of 9 segments, as Figure 5.4 shows. The carefully planned route consists of 7 miles (11.3 km) highway and 4.7 miles (7.5 km) suburban road, allowing participants to have enough time and diversity in driving scenarios to accommodate the use of both the LKS and ACC systems. Participants took an average of 20 minutes to complete this route. Participants were asked to complete a different task in every segment, as the next subsection describes.

5.3.4 Tasks

The detailed tasks are shown in the Table 5.1. The first segment (segment 0) is designed for the participants to become familiar with operating the vehicle and asking Adasa questions. Segment 1 consists of another portion of the local road including several stop signs where the participants were asked to turn on LKS. In this segment, participants continued driving while LKS is enabled on the local road.

Table 5.1: Summary of tasks assigned during the driving study.

Segment #	Task
0	**Start driving and be familiar with the vehicle**
1	**Turn on the lane keeping system**
2	**Get on highway and stay in right-most lane**
3	**Turn the adaptive cruise control to standby**
4	**Turn the adaptive cruise control to active**
5	**Drift out of lane slightly to drive on the lane line**
6	**Turn off the adaptive cruise control**
7	**Drift out of lane slightly to drive on the lane line**
8	**Turn off the lane keeping system**

The participants are then asked to get on the highway after finishing segment 1 and stay in the rightmost lane while keeping the vehicle speed at 60 miles per hour for safety. After entering the highway, the participants were asked to turn ACC to standby mode and active mode in segment 3 and segment 4 respectively. These two segments were designed for the participants to experience the ACC feature and ask Adasa questions if they chose to do so. In the segment 5, the participants were asked to drift out of lane slightly at the right-most lane to experience the LKS feature while ensuring that the vehicle is under control and the speed limit is followed. In segment 6, the participants were asked to get off the highway, turn off the ACC, and drive back to the starting location. Segment 7 was designed to let the participants experience LKS in the local area. This segment is a straight urban road with only one traffic light. During this segment, the participants were asked to test LKS again by drifting out slightly to step on the lane line. Finally, participants were asked to turn off LKS and drive to the end location. We expected drivers would encounter most of the driving scenarios when executing tasks during the drive, although this is not always possible. For example, even when ACC is set to active, it can still be hard to expect the driver to experience the Stop-and-Go function (i.e., detect when the vehicle ahead has stopped, and resume after the vehicle ahead moves) since the traffic conditions on road may vary.

5.3.5 Procedure

Upon the participant's arrival at the starting location, the participant was greeted and told that his/her help was needed to investigate the driving experience on a vehicle equipped with ADAS features and that he/she would interact with our digital assistant, Adasa. Also, each participant was required to sign an insurance form to obtain the university's approval for participating in the study. After asking for the participant's informed consent and checking his/her valid driver license, the participant was introduced to the vehicle and asked to "please have a seat in the drivers' seat". After both the researcher and participants were seated, the researcher showed the participant the pre-defined route. The participant was then informed that varying tasks were needed to be performed during each segment and was encouraged to ask any questions. These questions were not limited to questions about the two ADAS features, as long as the participants felt they were helpful for resolving their confusion. Since most of the participants were unfamiliar with ADAS features, which may share vehicle control with the driver, the researcher instructed the participant to prioritize their safety and allowed them to interrupt the study at any time if they felt nervous or were in a dangerous situation. The researcher stayed in the vehicle to guide the participant along the route as well as the tasks needed to be executed along each segment. As soon as the participant understood all the instructions, he/she was allowed to start the study and was instructed to utilize the first segment (segment 0) to get familiar with driving the vehicle.

Upon finishing the route and the arrival at the end location, the participant was asked to fill out a post-questionnaire regarding his/her driving experience during the drive. Afterwards, the participant was interviewed by the experimenter for more detailed information, and participants were asked about any further questions regarding the vehicle, Adasa, or ADAS features. At the point where the participant expressed he/she had no further questions, the study was deemed formally over.

Table 5.2: Targeted assessments of questions in questionnaire.

Question #	Description
Q1	**I think it is acceptable to have a voice assistant in the vehicle.**
Q2	**I think this car equipped with the voice assistant is intelligent.**
Q3	**The voice assistant helps me understand the features of advanced driving assistant system.**
Q4	**I think the answer that provided by the voice assistant is useful.**
Q5	**I think the voice assistant can understand my question and provide accurate diagnosis during driving.**
Q6	**The voice assistant makes using these advanced driving assistance features more pleasant.**
Q7	**Driving the vehicle with the voice assistant makes me nervous.**
Q8	**The responsiveness and reliability of the voice assistant meet my expectation.**

5.3.6 Questionnaire

Participants filled out a questionnaire adapted from DeLone and McLean information-system (IS) success model [31, 32], with questions that evaluate three main aspects of the system: system quality, information quality and user satisfaction. All questions in the questionnaire are specifically asked regarding the assistant itself without considering the quality of the ADAS features. Therefore, the evaluation is scoped to evaluate the effectiveness of Adasa in improving drivers' understanding of ADAS features rather than the usability of ADAS features on the vehicle. The participants answer each question in the categories with a 10-point Likert scale with anchors 1 = "strongly disagree" and 10 = "strongly agree". For system quality, the participants are asked to evaluate system acceptability, system intelligence and system helpfulness. For example, "*The digital assistant helps me understand the features of advanced driver assistance system.*" For information quality, the information usefulness and the accuracy of the diagnosis are evaluated. One example question is: "*I think the digital assistant can understand my question and provide accurate diagnosis during driving.*" For the user satisfaction, the participants are asked regarding their nervousness, pleasantness and if their expectation had been met. The targeted assessments of each question in the questionnaire are shown in Table 5.2. The participants were also welcome to provide comments at the end of the questionnaire for us to evaluate the system.

5.4 Results

With the user study conducted in a real-world environment, we are able to investigate drivers' behaviors and satisfaction when interacting with Adasa. In this section, we demonstrate the effectiveness of our system via quantitative and qualitative analyses of our user study.

5.4.1 Quantitative System Analysis

We perform a quantitative system analysis by evaluating three key metrics: (1) query understanding, (2) response correctness, and (3) processing latency. Particularly, query understanding (i.e., understanding the query correctly) and response correctness (i.e., responding with the correct answer) have been commonly used to evaluate conversational assistants such as Apple Siri and Amazon Echo [137, 138].

5.4.1.1 Query Understanding

We first evaluate query understanding by quantifying the percentage of the queries that are categorized into the corresponding intent class correctly, which aligns with how query understanding has commonly been evaluated in prior studies [137, 138] on several digital assistants including Amazon Echo, Google Home and Apple Siri. The experiment including 800 general queries was conducted in April 2017 for Apple Siri and August 2017 for Amazon Echo and Google Home respectively. To facilitate a fair comparison, we divide the data we collected via Amazon MTurk into two completely disjoint sets, the training set and the testing set, and report the accuracy of our model on the testing set to provide an unbiased evaluation. As shown in prior studies [137, 138], state-of-the-art digital assistants (e.g., Siri) are able to identify and understand over 90% of the queries asked. In our evaluation, we find Adasa is also able to categorize queries at an accuracy of 92.5%, which suggests that our system can achieve state-of-the-art query understanding.

5.4.1.2 Response Correctness

We then evaluate the response correctness, which can be quantified as the percentage of the queries answered correctly by the digital assistant. For Adasa, we measure the response correctness during the user study, where the participants were asked to inform the experimenter whether Adasa provided the correct answers or not

during the test drive. The results in our user study show that each participant asked 12.88 questions on average (i.e., 1–3 questions per segment) during the study. We found that Adasa achieves overall 77% response correctness, which aligns with the subjective user feedback that we present later this section. Overall, we found the participants highly satisfied with responses provided by the system. Comparing to the state-of-the-art digital assistants available on the market, our system achieves a comparable, if not better (i.e., 75.4% correctness on Apple Siri, 65.3% on Google Home, and 53.6% on Amazon Echo as shown in prior studies [137, 138]), level of response correctness.

5.4.1.3 Processing Latency

Besides the response correctness, the performance of the system can also be determined by the processing latency. A well-designed digital assistant should be able to process queries and respond in real-time, which is also the second key design objective mentioned in Section 5.2.1. We measure the processing latency with respect to query types (i.e., inquiry, symptom diagnosis and command and control) in the real vehicle deployment. The result demonstrates that it merely takes 1.50 seconds (M = 1.50, SD = 0.38) from driver pressing the button to Adasa responding on average. Specifically, the processing latency across three types of query is 1.17 seconds for FAQ (M = 1.17, SD = 0.19), 1.57 seconds for diagnosis (M = 1.57, SD = 0.21) and 1.76 seconds for command (M = 1.76, SD = 0.45).

5.4.2 Subjective User Feedback

Figure 5.5 presents the average scores of all the eight questions, where the x-axis represents different questions regarding Adasa usage with the average scores of these questions shown on the y-axis. A score of 7 out of 10 or greater was seen in all cases except 'nervousness', which will be discussed later, suggesting that Adasa is

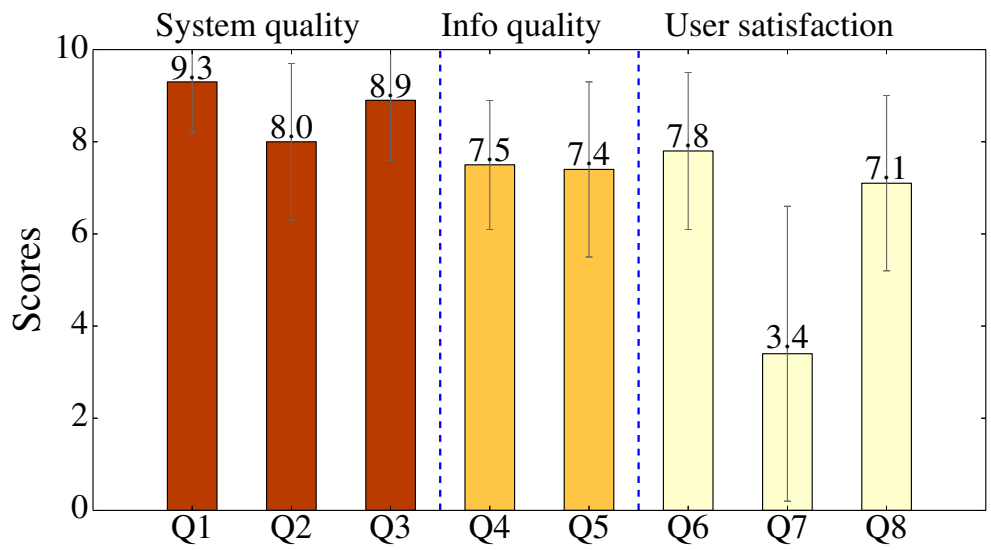


Figure 5.5: The average scores of the participants' feedback across different questions in the questionnaire. We obtain over 7 out of 10 in all cases and the participants are less nervous when using Adasa (Q7 score = 3.4). It suggests that Adasa is considered as a helpful and useful system for them to understand and utilize ADAS features in the vehicle.

considered to be a helpful and useful system for them to understand and utilize ADAS features in the vehicle, which aligns with the third design objective. An interesting observation is that 80% participants (12 out of 15) in our study have not used ADAS features previously since their own vehicles are not equipped with ADAS features, or they are unfamiliar with them. Adasa improves the understanding of ADAS as participants provide highly positive feedback (Q3 score = 8.9), showing Adasa is helpful to understand these features.

5.4.2.1 System Quality

System quality was evaluated by asking participants questions about the following three aspects: system acceptability, system intelligence, and system helpfulness. As shown in Figure 5.5, participants considered that an in-vehicle digital assistant like Adasa "highly acceptable" for in-vehicle use (Q1 score = 9.3) and it could help them understand the features of ADAS (Q3 score = 8.9). Participants mentioned: "*The*

voice assistant is convenient and makes me drive more safely as I could focus on the road all the time when driving and get responses I needed", "*It is useful to have while driving since I can keep my eyes on driving instead of seeing the dashboard.*", "*The voice assistant makes it much easier to access the ADAS features that are complex in the manual. Using voice is a much more natural way to interact with intelligent features in the car.*"

5.4.2.2 Information Quality

When evaluating the information quality Adasa provided, the expectation regarding the usefulness and the understandability of the information provided by Adasa are considered. Based on the feedback, most of the responses are precise and easy-to-understand (Q4 score = 7.5; Q5 score = 7.4). However, we observe that participants frequently asked questions regarding events that happened a short time ago. For example, one question was asked: " *What happened 3 minutes ago that my adaptive cruise control did not work at all?*" While Adasa currently is able to only respond to the queries based on the current status, this significant finding would help us improve our prototype in the future.

5.4.2.3 User Satisfaction

We evaluate the user satisfaction by considering participants' pleasantness, nervousness, and if Adasa could meet their expectations during driving. The results show that participants feel pleasant (Q6 score = 7.8) and Adasa could meet their expectations (Q8 score = 7.1), as shown in Figure 5.5. For user nervousness, participants indicated a score between 1 (not feeling nervous at all) and 10 (very nervous). Based on the results, while the average level of nervousness is low among all the participants (Q7 score = 3.4), we find two participants provided a score of 8 and a score of 9 respectively, showing they were highly nervous during driving. They men-

tioned that they remained conservative about using these features even though they considered Adasa helpful for them to understand these features. Apart from these two participants, other participants gave mostly 1's and 2's, showing they did not feel very nervous using Adasa while driving.

5.5 Discussion

While our results indicate that an in-vehicle digital assistant comparable to Adasa can indeed improve ADAS feature usability and the overall perceived driving experience, we also identified numerous practical insights and challenges. Particularly, there are two common themes arising from our observations: (1) response length; and (2) query completeness.

5.5.1 Response Length

Participants are able to interact with Adasa in several scenarios (e.g., driving as in on the highway or statically as in a parking lot). We find that participants react differently to similar responses in the different environment. For example, participants often anticipate brief responses while driving in a dynamic environment (e.g., highway, road intersection) since they are unable to digest the information provided by Adasa and focus on the road environments simultaneously. In contrast, more thorough explanations are expected as participants interact with Adasa in a static environment (e.g., parking lot) to understand how to use these ADAS features. This observation aligns with our feedback from the questionnaire that 60% (i.e., 9 out of 15) of the participants commented that they are satisfied with the comprehensive information provided and are able to readily understand and more quickly familiarize themselves with those particular ADAS features. Consequently, an Adasa-like system should be able to identify the current driving status and provide proper length of responses accordingly since different response lengths are expected depending on driver's current

status.

Finding - *Different length of responses are expected in varying driving environments as drivers might be distracted while driving. A well-designed digital assistant is capable of identifying the environment and providing proper length of responses.*

5.5.2 Query Completeness

We find that questions asked by drivers are usually incomplete and unstructured because of the following reasons: (1) drivers are unfamiliar with ADAS features so they are often unable to describe their questions precisely; (2) drivers often pay attention to the traffic conditions while interacting with Adasa-like system, which makes it difficult for them to structure complete sentences. However, Adasa is trained with the training data collected by MTurk, which is mostly comprised of complete sentences since those workers were unable to experience the system while driving and respond as such. As a participant mentioned: *"Most of the questions that I had the system could answer, but I had to repeat myself multiple times."* This demonstrates that drivers focus mostly on the road and their utterances might significantly deviate from correct grammar or complete sentences. Although our system can achieve up to 77% response correctness, this observation shows that training data for the classifier should include more diverse queries to build a much robust classifier for an Adasa-like system.

Finding - *Incomplete queries are asked frequently since drivers need to pay attention to traffic conditions and might not have prior knowledge to describe the questions precisely.*

5.6 Summary

In this Chapter, we present Adasa, a conversational in-vehicle digital assistant that intakes and answers driver's questions in natural spoken language. Drivers are able to

ask questions about or command and control both ACC and LKS using unconstrained natural language in the vehicle. The digital assistant trained using advanced machine learning techniques coupled with access to the vehicle signals responded in real-time based on conversational and environmental context. Results of the system deployed onto a production vehicle were presented demonstrating its effectiveness in improving driver understanding and usability of the ADAS.

We envision a few directions for future work. We believe that an Adasa-like system could educate drivers about ADAS features and broaden their knowledges about ADAS. Future studies could also evaluate longer-term learning gains by utilizing Adasa and the impacts to the driving experience. In addition, we anticipate gathering more variation in training data (e.g., incomplete queries) to further build a robust in-vehicle digital assistant based on our feedback observation. The ADAS features presented in this work only constitutes LKS and ACC. Extensions of this work would consider other features such as forward collision warning (FCW) and automatic parallel parking for users to understand and utilize these features and to further improve the driving experience.

Last, we expect that Adasa built in an autonomous vehicle is able to help humans from two different perspectives: intra- and inter-communication. For intra-communication, an Adasa-like system could help people with disabilities (e.g., blind people) to utilize the vehicle. For example, blind people could speak to the car regarding their destination and Adasa would be able to understand the content of the conversation and drive people toward the destination. This application provides the service for people with disabilities and improve both the usability of the car as well as their user experience. For inter-communication, vehicles require the ability to communicate with other road users (i.e., pedestrians) and understand their intentions so as to react accordingly. Prior works demonstrate the importance of identifying pedestrians' behavior [133, 109, 132, 67] for autonomous vehicles. Future studies could also

evaluate the impact of building Adasa-like system in autonomous vehicles that affects cars and pedestrians interaction.

We believe that in the future more ADAS features will continue to be introduced to improve the driving experience and drivers will be able to utilize an Adasa-like system to interact with the vehicle. We hope Adasa will encourage discussion and excite more in-vehicle interface design within the HCI community.

CHAPTER VI

Conclusion

A widespread interest in autonomous vehicles has grown significantly in recent years, including from companies such as Google, Tesla, Uber and Ford. Autonomous vehicles are able to understand their surroundings and make deicisions to take drivers to the ultimate destination with little human help. However, vehicles equipped with autonomous driving capability are still largely under experiments and several critical roadblocks observed prevent them from being available despite the remarkable improvements in our community. Specifically, key challenges are observed across the stack of autonomous driving systems from hardware, software to human-vehicle interaction and must be addressed to bridge the gap between such application and our community. This dissertation investigates cross-layer autonomous driving systems from hardware, software to human-vehicle interaction and proposes solutions across the stack to design future autonomous driving systems.

I first present and formalize the design constraints in performance, predictability, storage, thermal and power when building autonomous driving systems. To investigate the design of such systems, I build a representative end-to-end autonomous driving system using state-of-the-art machine learning algorithmic components. Using this system, I then identify three computational bottlenecks and explore three different accelerator platforms to accelerate these computational bottlenecks Based on

these accelerated system designs, I further explore the tradeoffs among performance, power and future scalability of autonomous driving systems. Secondly, I characterize the compute profile of object detectors, which is one of the critical bottlenecks in autonomous driving systems and extract insights regarding how intermediate information could be re-utilized. With the insights in mind, I propose two new computation reuse approaches, which enable us to leverage previous computed understandings and extract new features in the following frame to improve the processing performance of autonomous vehicles. Lastly, I conduct an in-depth pilot study and identify huge gaps between drivers and advanced in-vehicle systems. I design a conversational in-vehicle digital assistant, Adasa, that intakes and answers driver's questions in natural human language to improve the usability and experience. I then integrate Adasa onto a production vehicle and conduct a real-world driving study. Results of Adasa deployed onto a production vehicle were presented demonstrating its effectiveness in improving driver understanding and usability of advanced in-vehicle systems.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: a system for large-scale machine learning.

[2] J. Albericio, P. Judd, T. Hetherington, T. Aamodt, N. E. Jerger, and A. Moshovos. Cnvlutin: Ineffectual-neuron-free deep neural network computing. In *ACM SIGARCH Computer Architecture News*, volume 44, pages 1–13. IEEE Press, 2016.

[3] J. Albericio, P. Judd, T. Hetherington, T. Aamodt, N. E. Jerger, and A. Moshovos. Cnvlutin: Ineffectual-neuron-free deep neural network computing. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 1–13, June 2016.

[4] N. H. Amer, H. Zamzuri, K. Hudha, and Z. A. Kadir. Modelling and control strategies in path tracking control for autonomous ground vehicles: A review of state of the art and challenges. *Journal of Intelligent & Robotic Systems*, pages 1–30, 2016.

[5] Apple. iOS-CarPlay-Apple, 2017.

[6] K. Ashraf, B. Wu, F. N. Iandola, M. W. Moskewicz, and K. Keutzer. Shallow networks for high-accuracy road object-detection. *arXiv preprint arXiv:1606.01561*, 2016.

[7] Audi USA. 2017 Audi A4 ultra offers highest EPA-estimated fuel economy in competitive segment, 2017.

[8] A. Barón and P. Green. Safety and usability of speech interfaces for in-vehicle tasks while driving: A brief literature review. Technical report.

[9] C. Berger and B. Rumpe. Autonomous driving-5 years after the urban challenge: The anticipatory vehicle as a cyber-physical system. *arXiv preprint arXiv:1409.0413*, 2014.

[10] G. Bradski. *Dr. Dobb's Journal of Software Tools*, 2000.

[11] M. Braun, N. Broy, B. Pfleging, and F. Alt. A design space for conversational in-vehicle information systems. In *Proceedings of the 19th International Conference*

on Human-Computer Interaction with Mobile Devices and Services, page 79. ACM, 2017.

[12] A. Broad, M. Jones, and T.-Y. Lee. Recurrent multi-frame single shot detector for video object detection.

[13] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *European conference on computer vision*, pages 25–36. Springer, 2004.

[14] M. Buhrmester, T. Kwang, and S. D. Gosling. Amazon's mechanical turk: A new source of inexpensive, yet high-quality, data? *Perspectives on psychological science*, 6(1):3–5, 2011.

[15] S. Changpinyo, M. Sandler, and A. Zhmoginov. The power of sparsity in convolutional neural networks. *arXiv preprint arXiv:1702.06257*, 2017.

[16] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam. Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '14, pages 269–284, New York, NY, USA, 2014. ACM.

[17] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun, and O. Temam. Dadiannao: A machine-learning supercomputer. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-47, pages 609–622, Washington, DC, USA, 2014. IEEE Computer Society.

[18] Y.-H. Chen, J. Emer, and V. Sze. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. In *ACM SIGARCH Computer Architecture News*, volume 44, pages 367–379. IEEE Press, 2016.

[19] Chervolet. Chevrolet Bolt EV. `http://www.chevrolet.com/bolt-ev-electric-vehicle`, 2017.

[20] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*, 2014.

[21] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie. Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory. In *Proceedings of the 43rd International Symposium on Computer Architecture*, ISCA '16, pages 27–39, Piscataway, NJ, USA, 2016. IEEE Press.

[22] S. Choi, F. Thalmayr, D. Wee, and F. Weig. Advanced Driver-Assistance Systems: Challenges and Opportunities Ahead, 2016. McKinsey&Company.

[23] F. M. Company. Ford Sync, 2017.

[24] G. M. Company. Chevrolet MyLink: Take Control Of Your Vehicle's Technology, 2017.

[25] L. M. Company. 2017 LINCOLN CONTINENTAL, 2017.

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

[27] T. M. Corporation. Toyota-Entune, 2017.

[28] M. Courbariaux, Y. Bengio, and J.-P. David. Training deep neural networks with low precision multiplications. *arXiv preprint arXiv:1412.7024*, 2014.

[29] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[30] B. S. Craig Smoth. 160 Amazing YouTube Stats and Facts, 2018.

[31] W. H. DeLone and E. R. McLean. Information systems success: The quest for the dependent variable. *Information systems research*, 3(1):60–95, 1992.

[32] W. H. Delone and E. R. McLean. The delone and mclean model of information systems success: a ten-year update. *Journal of management information systems*, 19(4):9–30, 2003.

[33] Z. Du, D. D. Ben-Dayan Rubin, Y. Chen, L. He, T. Chen, L. Zhang, C. Wu, and O. Temam. Neuromorphic accelerators: A comparison between neuroscience and machine-learning approaches. In *Proceedings of the 48th International Symposium on Microarchitecture*, MICRO-48, pages 494–507, New York, NY, USA, 2015. ACM.

[34] Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen, and O. Temam. Shidiannao: Shifting vision processing closer to the sensor. In *Proceedings of the 42Nd Annual International Symposium on Computer Architecture*, ISCA '15, pages 92–104, New York, NY, USA, 2015. ACM.

[35] A. Eckert, A. Hohm, and S. Lueke. An integrated adas solution for pedestrian collision avoidance. In *Proceedings of the 23rd International Conference on the Enhanced Safety of Vehicles, Seoul, Republic of Korea*, pages 13–0298, 2013.

[36] Electrek. Elon Musk clarifies Tesla's plan for level 5 fully autonomous driving: 2 years away from sleeping in the car, 2017.

[37] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. `http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html`, 2012.

[38] R. Farrington and J. Rugh. Impact of vehicle air-conditioning on fuel economy, tailpipe emissions, and electric vehicle range. In *Earth technologies forum*, pages 1–6, 2000.

[39] M. A. Fayazbakhsh and M. Bahrami. Comprehensive modeling of vehicle air conditioning loads using heat balance method. Technical report, SAE Technical Paper, 2013.

[40] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. Van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. *arXiv preprint arXiv:1504.06852*, 2015.

[41] M. C. for Future Mobility. Autonomous Driving, 2017.

[42] Y. Forster, F. Naujoks, and A. Neukum. Increasing anthropomorphism and trust in automated driving functions by adding speech output. In *Intelligent Vehicles Symposium (IV), 2017 IEEE*, pages 365–372. IEEE, 2017.

[43] M. Gao, C. Delimitrou, D. Niu, K. T. Malladi, H. Zheng, B. Brennan, and C. Kozyrakis. Draf: A low-power dram-based reconfigurable acceleration fabric. In *Proceedings of the 43rd International Symposium on Computer Architecture*, ISCA '16, pages 506–518, Piscataway, NJ, USA, 2016. IEEE Press.

[44] M. Gärtner, A. Meschtscherjakov, B. Maurer, D. Wilfinger, and M. Tscheligi. Dad, stop crashing my car!: Making use of probing to inspire the design of future in-car interfaces. In *Proceedings of the 6th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, pages 1–8. ACM, 2014.

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

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

[47] Global Management Consulting. Autonomous Vehicle Adoption Study, 2016.

[48] Google. Are We There Yet? Silicon in Self-Driving Cars., 2016.

[49] Google. Android Auto, 2017.

[50] B. Graham. Sparse 3d convolutional neural networks. *arXiv preprint arXiv:1505.02890*, 2015.

[51] B. Graham and L. van der Maaten. Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307*, 2017.

[52] R. Graham and C. Carter. Comparison of speech input and manual control of in-car devices while on the move. *Personal Technologies*, 4(2):155–164, 2000.

[53] B. C. Group. Nearly 10,000 Deaths Could Be Prevented and More Than $250 Billion Saved with Greater Use of Driver Assistance Technologies, 2015.

[54] Y. Guo, A. Yao, and Y. Chen. Dynamic network surgery for efficient dnns. In *Advances In Neural Information Processing Systems*, pages 1379–1387, 2016.

[55] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally. Eie: efficient inference engine on compressed deep neural network. In *Computer Architecture (ISCA), 2016 ACM/IEEE 43rd Annual International Symposium on*, pages 243–254. IEEE, 2016.

[56] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[57] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[58] J. H. Hansen, P. Angkititrakul, J. P. Plucienkowski, S. Gallant, U. H. Yapanel, B. L. Pellom, W. H. Ward, and R. A. Cole. " cu-move": analysis & corpus development for interactive in-vehicle speech systems. In *INTERSPEECH*, pages 2023–2026, 2001.

[59] B. Hassibi and D. G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in neural information processing systems*, pages 164–171, 1993.

[60] J. Hauswald, Y. Kang, M. A. Laurenzano, Q. Chen, C. Li, T. Mudge, R. G. Dreslinski, J. Mars, and L. Tang. Djinn and tonic: Dnn as a service and its implications for future warehouse scale computers. In *Proceedings of the 42Nd Annual International Symposium on Computer Architecture*, ISCA '15, pages 27–40, New York, NY, USA, 2015. ACM.

[61] J. Hauswald, M. A. Laurenzano, Y. Zhang, C. Li, A. Rovinski, A. Khurana, R. G. Dreslinski, T. Mudge, V. Petrucci, L. Tang, et al. Sirius: An open end-to-end voice and vision personal assistant and its implications for future warehouse scale computers. In *ACM SIGPLAN Notices*, volume 50, pages 223–238. ACM, 2015.

[62] J. Hauswald, M. A. Laurenzano, Y. Zhang, C. Li, A. Rovinski, A. Khurana, R. G. Dreslinski, T. Mudge, V. Petrucci, L. Tang, and J. Mars. Sirius: An

open end-to-end voice and vision personal assistant and its implications for future warehouse scale computers. In *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '15, pages 223–238, New York, NY, USA, 2015. ACM.

[63] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European conference on computer vision*, pages 346–361. Springer, 2014.

[64] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference on Computer Vision*, pages 749–765. Springer, 2016.

[65] P. Hill, A. Jain, M. Hill, B. Zamirai, C.-H. Hsu, M. A. Laurenzano, S. Mahlke, L. Tang, and J. Mars. Deftnn: Addressing bottlenecks for dnn execution on gpus via synapse vector elimination and near-compute data fission. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 786–799. ACM, 2017.

[66] J. M. Hirst, J. R. Miller, B. A. Kaplan, and D. D. Reed. Watts up? pro ac power meter for automated energy recording: A product review. *Behavior Analysis in Practice*, 6(1):82, 2013.

[67] L. Hobert, A. Festag, I. Llatser, L. Altomare, F. Visintainer, and A. Kovacs. Enhancements of v2x communication in support of cooperative autonomous driving. *IEEE communications magazine*, 53(12):64–70, 2015.

[68] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. 2017.

[69] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks. In *Advances in neural information processing systems*, pages 4107–4115, 2016.

[70] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898, 2017.

[71] J. Hur and S. Roth. Joint optical flow and temporally consistent semantic segmentation. In *European Conference on Computer Vision*, pages 163–177. Springer, 2016.

[72] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[73] Intel. Intel Core i7-4790K Processor, 2017.

[74] A. Jain, P. Hill, S.-C. Lin, M. Khan, M. E. Haque, M. A. Laurenzano, S. Mahlke, L. Tang, and J. Mars. Concise loads and stores: The case for an asymmetric compute-memory architecture for approximation. In *Microarchitecture (MICRO), 2016 49th Annual IEEE/ACM International Symposium on*, pages 1–13. IEEE, 2016.

[75] A. Jain, A. Phanishayee, J. Mars, L. Tang, and G. Pekhimenko. Gist: Efficient data encoding for deep neural network training. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pages 776–789. IEEE, 2018.

[76] S. Jain and J. E. Gonzalez. Fast semantic segmentation on video using motion vector-based feature interpolation. *arXiv preprint arXiv:1803.07742*, 2018.

[77] Y. Ji, Y. Zhang, S. Li, P. Chi, C. Jiang, P. Qu, Y. Xie, and W. Chen. Neutrams: Neural network transformation and co-design under neuromorphic hardware constraints. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–13, Oct 2016.

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

[79] G. Johansson and K. Rumar. Drivers' brake reaction times. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 13(1):23–27, 1971.

[80] R. W. Johnson, J. L. Evans, P. Jacobsen, J. R. Thompson, and M. Christopher. The changing automotive environment: high-temperature electronics. *IEEE Transactions on Electronics Packaging Manufacturing*, 27(3):164–176, 2004.

[81] K. A. Joudi, A. S. K. Mohammed, and M. K. Aljanabi. Experimental and computer performance study of an automotive air conditioning system with alternative refrigerants. *Energy conversion and Management*, 44(18):2959–2976, 2003.

[82] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, et al. In-datacenter performance analysis of a tensor processing unit. *arXiv preprint arXiv:1704.04760*, 2017.

[83] P. Judd, J. Albericio, T. Hetherington, T. M. Aamodt, N. E. Jerger, and A. Moshovos. Proteus: Exploiting numerical precision variability in deep neural networks. In *Proceedings of the 2016 International Conference on Supercomputing*, page 23. ACM, 2016.

[84] P. Judd, J. Albericio, T. Hetherington, T. M. Aamodt, and A. Moshovos. Stripes: Bit-serial deep neural network computing. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–12, Oct 2016.

[85] M. F. Jung, D. Sirkin, T. M. Gür, and M. Steinert. Displayed uncertainty improves driving experience and behavior: The case of range anxiety in an electric car. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 2201–2210. ACM, 2015.

[86] S. Kato, E. Takeuchi, Y. Ishiguro, Y. Ninomiya, K. Takeda, and T. Hamada. An open approach to autonomous vehicles. *IEEE Micro*, 35(6):60–68, 2015.

[87] D. Kern and A. Schmidt. Design space for driver-based automotive user interfaces. In *Proceedings of the 1st International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, pages 3–10. ACM, 2009.

[88] D. Kim, J. Kung, S. Chai, S. Yalamanchili, and S. Mukhopadhyay. Neurocube: A programmable digital neuromorphic architecture with high-density 3d memory. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 380–392, June 2016.

[89] P. Koopman and M. Wagner. Challenges in autonomous vehicle testing and validation. *SAE International Journal of Transportation Safety*, 4(1):15–24, 2016.

[90] P. Kortum. *HCI beyond the GUI: Design for haptic, speech, olfactory, and other nontraditional interfaces*. Morgan Kaufmann, 2008.

[91] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, L. Čehovin, G. Nebehay, T. Vojíř, G. Fernández, A. Lukežič, A. Dimitriev, A. Petrosino, A. Saffari, B. Li, B. Han, C. Heng, C. Garcia, D. Pangeršič, G. Häger, F. S. Khan, F. Oven, H. Possegger, H. Bischof, H. Nam, J. Zhu, J. Li, J. Y. Choi, J.-W. Choi, J. F. Henriques, J. van de Weijer, J. Batista, K. Lebeda, K. Öfjäll, K. M. Yi, L. Qin, L. Wen, M. E. Maresca, M. Danelljan, M. Felsberg, M.-M. Cheng, P. Torr, Q. Huang, R. Bowden, S. Hare, S. Y. Lim, S. Hong, S. Liao, S. Hadfield, S. Z. Li, S. Duffner, S. Golodetz, T. Mauthner, V. Vineet, W. Lin, Y. Li, Y. Qi, Z. Lei, and Z. H. Niu. *The Visual Object Tracking VOT2014 Challenge Results*, pages 191–217. Springer International Publishing, Cham, 2015.

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

[93] D. M. Krum, J. Faenger, B. Lathrop, J. A. Sison, and A. Lien. All roads lead to chi: interaction in the automobile. In *CHI'08 Extended Abstracts on Human Factors in Computing Systems*, pages 2387–2390. ACM, 2008.

[94] A. R. Kumar, B. Ravindran, and A. Raghunathan. Pack and detect: Fast object detection in videos using region-of-interest packing. In *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*, pages 150–156. ACM, 2019.

[95] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th international conference on Machine learning*, pages 473–480. ACM, 2007.

[96] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[97] J. D. Lee, B. Caven, S. Haake, and T. L. Brown. Speech-based interaction with in-vehicle computers: The effect of speech-based e-mail on drivers' attention to the roadway. *Human factors*, 43(4):631–640, 2001.

[98] K. J. Lee, Y. K. Joo, and C. Nass. Partially intelligent automobiles and driving experience at the moment of system transition. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14.

[99] J. Levinson, M. Montemerlo, and S. Thrun. Map-based precision vehicle localization in urban environments. In *Robotics : Science and Systems (RSS), 2007*, 2007.

[100] G. Li, Y. Xie, T. Wei, K. Wang, and L. Lin. Flow guided recurrent neural encoder for video salient object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3243–3252, 2018.

[101] Z. Li, B. Ni, W. Zhang, X. Yang, and W. Gao. Performance guaranteed network acceleration via high-order residual quantization.

[102] S.-C. Lin, Y. Zhang, C.-H. Hsu, M. Skach, M. Haque, L. Tang, and J. Mars. The architectural implications of autonomous driving: Constraints and acceleration. In *Proceedings of the 23th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, ASPLOS-23, Williamsburg, VA, USA, 2018. ACM.

[103] 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*, pages 740–755. Springer, 2014.

[104] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky. Sparse convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 806–814, 2015.

[105] D. Liu, T. Chen, S. Liu, J. Zhou, S. Zhou, O. Teman, X. Feng, X. Zhou, and Y. Chen. Pudiannao: A polyvalent machine learning accelerator. In *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '15, pages 369–381, New York, NY, USA, 2015. ACM.

[106] V. E.-W. Lo and P. A. Green. Development and evaluation of automotive speech interfaces: useful information from the human factors and the related literature. *International Journal of Vehicular Technology*, 2013, 2013.

[107] J.-H. Luo, J. Wu, and W. Lin. Thinet: A filter level pruning method for deep neural network compression. *arXiv preprint arXiv:1707.06342*, 2017.

[108] D. Mahajan, J. Park, E. Amaro, H. Sharma, A. Yazdanbakhsh, J. K. Kim, and H. Esmaeilzadeh. Tabla: A unified template-based framework for accelerating statistical machine learning. In *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 14–26, March 2016.

[109] M. Matthews, G. Chowdhary, and E. Kieson. Intent communication between autonomous vehicles and pedestrians. *arXiv preprint arXiv:1708.07123*, 2017.

[110] D. V. McGehee, E. N. Mazzae, and G. S. Baldwin. Driver reaction time in crash avoidance research: Validation of a driving simulator study on a test track. In *Proceedings of the human factors and ergonomics society annual meeting*, volume 44, pages 3–320. SAGE Publications, 2000.

[111] C. McManus, W. Churchill, A. Napier, B. Davis, and P. Newman. Distraction suppression for vision-based pose estimation at city scales. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3762–3769. IEEE, 2013.

[112] M. McNaughton, C. Urmson, J. M. Dolan, and J.-W. Lee. Motion planning for autonomous driving with a conformal spatiotemporal lattice. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4889–4895. IEEE, 2011.

[113] J. Miller, A. Emadi, A. Rajarathnam, and M. Ehsani. Current status and future trends in more electric car power systems. In *Vehicular Technology Conference, 1999 IEEE 49th*, volume 2, pages 1380–1384. IEEE, 1999.

[114] Mobileye. Autonomous Driving. `https://www.mobileye.com/`, 2017.

[115] Mobileye. Enabling Autonomous. `http://www.mobileye.com/future-of-mobility/mobileye-enabling-autonomous/`, 2017.

[116] Mobileye. Mobileye C2-270 Essentials, 2017.

[117] Mobileye. Mobileye CES 2017 Press Conference, 2017.

[118] M. Mody. ADAS Front Camera: Demystifying Resolution and Frame-Rate. EETimes, 2016.

[119] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[120] A. Napier and P. Newman. Generation and exploitation of local orthographic imagery for road vehicle localisation. In *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pages 590–596. IEEE, 2012.

[121] C. Nass, I.-M. Jonsson, H. Harris, B. Reaves, J. Endo, S. Brave, and L. Takayama. Improving automotive safety by pairing driver emotion and car voice emotion. In *CHI'05 Extended Abstracts on Human Factors in Computing Systems*, pages 1973–1976. ACM, 2005.

[122] A. Newell and S. K. Card. The prospects for psychological science in human-computer interaction. *Human-computer interaction*, 1(3):209–242, 1985.

[123] D. Norman. *The design of everyday things: Revised and expanded edition*. Basic Books (AZ), 2013.

[124] D. G. Novick and K. Ward. Why don't people read the manual? In *Proceedings of the 24th annual ACM international conference on Design of communication*, pages 11–18. ACM, 2006.

[125] E. Ohn-Bar and M. M. Trivedi. Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations. *IEEE transactions on intelligent transportation systems*, 15(6):2368–2377, 2014.

[126] B. Pan, W. Lin, X. Fang, C. Huang, B. Zhou, and C. Lu. Recurrent residual module for fast inference in videos.

[127] A. Parashar, M. Rhu, A. Mukkara, A. Puglielli, R. Venkatesan, B. Khailany, J. Emer, S. W. Keckler, and W. J. Dally. Scnn: An accelerator for compressed-sparse convolutional neural networks. In *ACM SIGARCH Computer Architecture News*, volume 45, pages 27–40. ACM, 2017.

[128] V. Patraucean, A. Handa, and R. Cipolla. Spatio-temporal video autoencoder with differentiable memory. *arXiv preprint arXiv:1511.06309*, 2015.

[129] D. J. Perreault and V. Caliskan. Automotive power generation and control. *IEEE Transactions on Power Electronics*, 19(3):618–630, 2004.

[130] T. Pfister, J. Charles, and A. Zisserman. Flowing convnets for human pose estimation in videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1913–1921, 2015.

[131] M. Pivtoraiko, R. A. Knepper, and A. Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26(3):308–333, 2009.

[132] A. Rasouli, I. Kotseruba, and J. K. Tsotsos. Are they going to cross? a benchmark dataset and baseline for pedestrian crosswalk behavior. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 206–213, 2017.

[133] A. Rasouli and J. K. Tsotsos. Autonomous vehicles that interact with pedestrians: A survey of theory and practice. *arXiv preprint arXiv:1805.11773*, 2018.

[134] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.

[135] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016.

[136] 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*, pages 91–99, 2015.

[137] L. V. Research. Faceoff: Amazon echo show VS Google home part II, 2017.

[138] L. V. Research. Siri semester exam grade improves to C from D+, 2018.

[139] M. Rhu, N. Gimelshein, J. Clemons, A. Zulfiqar, and S. W. Keckler. vdnn: Virtualized deep neural networks for scalable, memory-efficient neural network design. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–13, Oct 2016.

[140] 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*, 115(3):211–252, 2015.

[141] SAE International. AUTOMATED DRIVING, Levels of driving automation are defined in new SAE International standard J3016. `http://www.sae.org/misc/pdfs/automated_driving.pdf`, 2014.

[142] E. M. Schau, M. Traverso, and M. Finkbeiner. Life cycle approach to sustainability assessment: a case study of remanufactured alternators. *Journal of Remanufacturing*, 2(1):1–14, 2012.

[143] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[144] A. Schmidt, A. K. Dey, A. L. Kun, and W. Spiessl. Automotive user interfaces: human computer interaction in the car. In *CHI'10 Extended Abstracts on Human Factors in Computing Systems*, pages 3177–3180. ACM, 2010.

[145] Seagate Technology LLC. Seagate Desktop HDD Specification. `http://www.seagate.com/consumer/upgrade/desktop-hdd/\#specs`, 2017.

[146] L. Sevilla-Lara, D. Sun, V. Jampani, and M. J. Black. Optical flow with semantic segmentation and localized layers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3889–3898, 2016.

[147] S. Shalev-Shwartz, S. Shammah, and A. Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *NIPS Workshop on Learning, Inference and Control of Multi-Agent Systems*, 2016.

[148] H. Sharma, J. Park, D. Mahajan, E. Amaro, J. K. Kim, C. Shao, A. Mishra, and H. Esmaeilzadeh. From high-level deep neural models to fpgas. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–12, Oct 2016.

[149] S. Shi and X. Chu. Speeding up convolutional neural networks by exploiting the sparsity of rectifier units. *arXiv preprint arXiv:1704.07724*, 2017.

[150] D. Sirkin, N. Martelaro, M. Johns, and W. Ju. Toward measurement of situation awareness in autonomous vehicles. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 405–415. ACM, 2017.

[151] A. D. Stewart and P. Newman. Laps-localisation using appearance of prior structure: 6-dof monocular camera localisation using prior pointclouds. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2625–2632. IEEE, 2012.

[152] D. L. Strayer, J. M. Cooper, J. Turrill, J. R. Coleman, and R. J. Hopman. The smartphone and the drivers cognitive workload: A comparison of apple, google, and microsofts intelligent personal assistants. *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale*, 71(2):93, 2017.

[153] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 573–580. IEEE, 2012.

[154] O. Täckström, D. Das, S. Petrov, R. McDonald, and J. Nivre. Token and type constraints for cross-lingual part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, 1:1–12, 2013.

[155] TechCrunch. Intel buys mobileye in $15.3b deal, moves its automotive unit to israel, 2017.

[156] TechCrunch. Nvidia is powering the world's first Level 3 self-driving production car, 2017.

[157] TechCrunch. Waymo reveals completely homegrown sensor suite for Pacifica autonomous test car, 2017.

[158] Tesla. Full Self-Driving Hardware on All Cars. `https://www.tesla.com/autopilot`, 2017.

[159] Tesla Inc. Tesla Autopilot: Full Self-Driving Hardware on All Cars. `https://www.tesla.com/autopilot`, 2017.

[160] S. Thorpe, D. Fize, C. Marlot, et al. Speed of processing in the human visual system. *nature*, 381(6582):520–522, 1996.

[161] S. Truschin, M. Schermann, S. Goswami, and H. Krcmar. Designing interfaces for multiple-goal environments: Experimental insights from in-vehicle speech interfaces. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 21(1):7, 2014.

[162] Udacity. An Open Source Self-Driving Car. `https://www.udacity.com/self-driving-car`, 2017.

[163] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers. Finn: A framework for fast, scalable binarized neural network inference. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 65–74. ACM, 2017.

[164] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

[165] U.S. Department of Transportation – Federal Highway Administration. Highway Statistics 2015. `https://www.fhwa.dot.gov/policyinformation/statistics.cfm`, 2015.

[166] U.S. Department of Transportation – National Highway Traffic Safety Administration. Federal Automated Vehicles Policy: Accelerating the Next Revolution in Roadway Safety. `https://www.transportation.gov/AV`, 2017.

[167] Velodyne. Velodyne HDL-64E LiDAR . `http://velodynelidar.com/hdl-64e.html`, 2017.

[168] Venturebeat. Google's speech recognition technology now has a 4.9% word error rate, 2017.

[169] Waymo. Introducing Waymo's suite of custom-built, self-driving hardware, 2017.

[170] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2074–2082, 2016.

[171] F. Weng, P. Angkititrakul, E. E. Shriberg, L. Heck, S. Peters, and J. H. Hansen. Conversational in-vehicle dialog systems: The past, present, and future. *IEEE Signal Processing Magazine*, 33(6):49–60, 2016.

[172] F. Weng, S. Varges, B. Raghunathan, F. Ratiu, H. Pon-Barry, B. Lathrop, Q. Zhang, H. Bratt, T. Scheideck, K. Xu, et al. Chat: A conversational helper for automotive tasks. In *Ninth International Conference on Spoken Language Processing*, 2006.

[173] R. W. Wolcott and R. M. Eustice. Visual localization within lidar maps for automated urban driving. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 176–183. IEEE, 2014.

[174] R. W. Wolcott and R. M. Eustice. Fast lidar localization using multiresolution gaussian mixture maps. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 2814–2821. IEEE, 2015.

[175] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng. Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4820–4828, 2016.

[176] F. Xiao and Y. Jae Lee. Video object detection with an aligned spatial-temporal memory. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 485–501, 2018.

[177] T.-J. Yang, Y.-H. Chen, and V. Sze. Designing energy-efficient convolutional neural networks using energy-aware pruning. *arXiv preprint arXiv:1611.05128*, 2016.

[178] R. Yazdani, A. Segura, J. M. Arnau, and A. Gonzalez. An ultra low-power hardware accelerator for automatic speech recognition. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–12, Oct 2016.

[179] Y. C. Yeh. Triple-triple redundant 777 primary flight computer. In *Aerospace Applications Conference, 1996. Proceedings., 1996 IEEE*, volume 1, pages 293–307. IEEE, 1996.

[180] X. Zabulis, H. Baltzakis, and A. A. Argyros. Vision-based hand gesture recognition for human-computer interaction. *The universal access handbook*, 34:30, 2009.

[181] C. Zhang, Z. Fang, P. Zhou, P. Pan, and J. Cong. Caffeine: Towards uniformed representation and acceleration for deep convolutional neural networks. In *Computer-Aided Design (ICCAD), 2016 IEEE/ACM International Conference on*, pages 1–8. IEEE, 2016.

[182] J. Zhang and S. Singh. Visual-lidar odometry and mapping: Low-drift, robust, and fast. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 2174–2181. IEEE, 2015.

[183] W. Zhang, P. Srinivasan, and J. Shi. Discriminative image warping with attribute flow. In *CVPR 2011*, pages 2393–2400. IEEE, 2011.

[184] X. Zhu, J. Dai, L. Yuan, and Y. Wei. Towards high performance video object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7210–7218, 2018.

[185] X. Zhu, Y. Wang, J. Dai, L. Yuan, and Y. Wei. Flow-guided feature aggregation for video object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 408–417, 2017.

[186] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei. Deep feature flow for video recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2349–2358, 2017.