Edge-Assisted Over-the-Air Software Updates

Arpan Bhattacharjee*, Hamza Mahmood*, Sidi Lu†, Nejib Ammar‡, Akila Ganlath‡, Weisong Shi*
University of Delaware, DE, USA***, William & Mary†, Toyota InfoTech Lab‡‡
{arpan,mhamza,weisong}@udel.edu, sidi@wm.edu, {nejib.ammar,akila.ganlath}@toyota.com

2023 IEEE 9th International Conference on Collaboration and Internet Computing (CIC) | 979-8-3503-3912-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/CIC58953.2023.00013

***Abstract*—The imperative transformation of future connected vehicles into intelligent computing platforms necessitates the implementation of Software-defined Vehicles (SDVs). SDVs enable the progressive addition and upgrading of automotive applications throughout the vehicle's lifecycle, facilitated by software Over-the-Air (OTA) update technology. However, the exploration of OTA updates for automotive applications is currently very limited. To the best of our knowledge, our work is pioneering in implementing an edge-assisted framework for automotive OTA updates. It meticulously considers the heterogeneity of vehicular software models and vehicle computing units, varied communication distances, and diverse sizes of vehicle clusters. We offer crucial insights based on pertinent evaluation metrics, such as update latency, transmission bandwidth, and successful rate, alongside an in-depth scalability analysis. Our study employs three distinct sizes of fundamental vehicle models: ResNet-18 (46.8 MB), ResNet-50 (102.5 MB), and Faster R-CNN (175.2 MB). These models are employed to evaluate the update performance across eight distance groups ranging from 0 to 21 meters with a 3-meter gap. We also exploit heterogeneous computing platforms to appraise the success rate and execute comprehensive scalability analysis. This novel approach significantly advances the current understanding and implementation of OTA updates in the automotive community.**

***Index Terms*—Software-Defined Vehicle (SDV), Edge Computing, Software Over-the-Air (OTA) update, Autonomous Vehicles and Vehicle Computing.**

I. Introduction

A. Software Defined Vehicle (SDV)

SDV is an automobile with predominantly software-driven capabilities and functionalities. It represents the ongoing evolution of automobiles, shifting from being predominantly hardware-oriented products to becoming electronic devices on wheels with a strong emphasis on software integration [1]. An extensive volume of software code has become an integral part of modern luxury vehicles. Specific high-end models now incorporate an astonishing 150 million lines of code [2]. These lines are intricately woven throughout many electronic control units (ECUs) and advanced sensors, cameras, radar systems, light detection, and lidar devices [3]. The convergence of three key trends—electrification, automation, and connectivity—is revolutionizing customer expectations, compelling manufacturers to increasingly rely on software solutions to meet the evolving demands of the market [3]. Consumers are increasingly drawn to software-centric features encompassing advanced driver assistance functionalities, cutting-edge infotainment systems, and intelligent connectivity solutions [4].

B. Software Over-the-Air (OTA) Updates

Software OTA updates refer to remotely updating and upgrading software on devices, such as smartphones, IoT devices, and more recently connected vehicles.

1) Traditional OTA Updates: Traditionally, OTA updates provide a convenient and efficient way to deliver bug fixes, security patches, feature enhancements, and other software improvements to many devices without physical access [7]. By wirelessly transmitting the updated software to the target devices, OTA updates enable seamless and on-demand software maintenance and ensure that devices run the latest and most secure software versions. This mechanism is crucial in enhancing software functionality, security, and user experience while minimizing downtime and the need for manual intervention in the update process. Table. I shows a comparative analysis of three distinct approaches to OTA updates in the context of Apple [5], Linux [6] and Tesla [7].

2) Automotive OTA Updates: The significance of OTA updates has grown significantly in connected and autonomous vehicles. These updates are crucial in delivering new software features to vehicles without physical visits to service centers. By leveraging OTA updates, vehicles can seamlessly receive and install the latest software improvements, ensuring enhanced functionality and reliability while eliminating the logistical challenges associated with traditional maintenance procedures [8]. Furthermore, OTA updates offer manufacturers flexibility and agility in swiftly introducing new vehicle features and functionalities, effectively addressing customer needs and market demands with more significant efficiency [9]. These updates also enhance performance by optimizing vehicle systems, improving fuel efficiency, and fine-tuning autonomous driving capabilities. In addition, they also bolster security measures by promptly addressing vulnerabilities and safeguarding vehicles against cyber threats while enhancing the consumer experience by ensuring vehicles remain up to date with the latest software features and improvements. This, in turn, elevates customer satisfaction and extends the longevity of vehicles.

As cloud, edge, and vehicle computing technologies get popular, they will likely play a pivotal role in enabling efficient OTA updates from edge RSUs to vehicles. By bringing computational resources closer to vehicles, cloud/edge computing reduces latency and network congestion, ensuring faster and more reliable updates [10]. RSUs with edge computing infrastructure serve as local repositories for update packages, minimizing the need for individual

979-8-3503-3912-3/23/$31.00 ©2023 IEEE
DOI 10.1109/CIC58953.2023.00013

TABLE I
COMPARATIVE ANALYSIS OF OTA UPDATES BETWEEN APPLE, LINUX, AND TESLA.

Criteria	Apple	Linux	Tesla
Update Delivery Mechanism	Distributed through Apple Software delivery mechanism.	Utilize package management systems like APT or DNF to deliver updates.	Tesla vehicles receive OTA updates remotely initiated by the company.
Scope of Updates	Updates include new features, bug fixes, performance improvements, and security patches	Primarily focus on software packages and can include bug fixes, security patches, and new package versions	Updates encompass various components such as firmware, Autopilot system, and infotainment features, enabling continuous feature enhancements and bug fixes
Emphasis on Security	Updates are digitally signed and verified to prevent tampering or unauthorized modifications	Depends on the distribution and package sources, with maintainers ensuring the integrity of software packages	Digitally signed by the company, ensuring the authenticity and integrity of the software delivered to vehicles
User Intervention	Requires minimal user intervention, with automatic prompts and an intuitive installation process	Requires user intervention, such as running package managers or approving updates, and may involve system restarts	Initiated remotely by the company, reducing the need for user intervention beyond confirming installation
Industry Impact	Significant impact on the consumer electronics industry, providing regular updates and improving device performance and security	Popularized the concept of OTA updates in the open-source community, ensuring software packages stay up to date with bug fixes and security patches	Revolutionized the automotive industry, allowing vehicles to receive new features, enhancements, and bug fixes without requiring physical visits to service centers

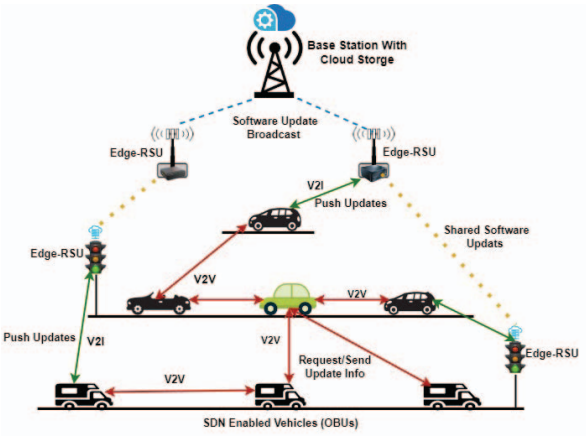


Fig. 1. OTA Updates from Edge-RSU to SDN enabled Vehicles (OBU).

vehicle connections to remote servers. This localized storage and processing optimize bandwidth utilization. Additionally, edge computing enables the intelligent distribution of updates based on location and criticality, improving efficiency. Furthermore, edge computing enhances OTA update security through authentication and verification within the trusted edge environment [11]. Overall, edge computing empowers RSUs to efficiently deliver OTA updates, improving reliability, bandwidth utilization, and security. Fig. 1 showcases an edge-vehicle OTA architecture, wherein the SDN controller delegates control plane management to local controllers or Edge-RSUs and end vehicle nodes. In this setup, the SDN controller base station's role is limited to forwarding new software updates and policies to the Edge-RSUs, rather than sending the entire updates and instructions to the vehicles for individual update decisions. This decentralized approach empowers Edge-RSUs and vehicle nodes to autonomously handle their current software updates, enabling efficient and streamlined OTA management within the network.

3) Motivation of Adopting OTA: Our primary motivation to adopt OTA is as follows:

- Minimization of vehicle recalls and cost savings: OTA updates significantly reduce the need for vehicle recalls, resulting in substantial cost reductions for automakers.
- Timely updates beyond traditional locations: OTA updates allow vehicles to be updated at various locations, such as the owner's home or workplace, eliminating the need to visit dealerships or maintenance garages.
- Centralized software distribution: OTA updates enable direct distribution of software to target vehicles without the involvement of dealers and maintenance garages.
- Faster time to market: New software can be distributed efficiently as needed without relying on customer vehicle returns or scheduled maintenance.
- Mandatory updates for improved safety: Critical updates, particularly safety-related ones, can be pushed to vehicles without waiting for customer approval.
- Proven technology from the telecommunications industry: OTA updates have been widely adopted in the telecommunications industry, providing users with updated software through successful OTA mechanisms.

4) Challenges of Adopting OTA: The challenges of adopting OTA updates is as follows:

- Network Connectivity and Bandwidth: OTA updates require stable and reliable network connectivity, particularly in areas with limited internet access. Bandwidth limitations can lead to slow or failed updates.
- Compatibility and Fragmentation: Ensuring software compatibility across different vehicle models, hardware configurations, and software versions is challenging.
- Security Risks: OTA updates introduce potential security vulnerabilities, as they involve transmitting and installing software over the air. Safeguarding against unauthorized access, tampering and ensuring data integrity during the update process is crucial.

- Data Management and Storage: Efficiently managing and storing the data generated by OTA updates, including update histories, is essential.
- User Acceptance and Trust: Gaining user trust and addressing concerns about privacy, security, and potential disruptions are essential for OTA update adoption.

C. Contribution of This Work

The core contribution of this work is in implementing software OTA updates on the physical testbed and providing actionable insights on the trade-off between updating different sizes of software, diverse distances, successful rates, and transmission bandwidth. We also conduct a comprehensive scalability analysis on a group of heterogeneous computing units and figure out their influences on the success of the OTA update, latency, and bandwidth. Specific contributions are listed as follows.

- To the best of our knowledge, this is the first work to implement an edge-assisted framework for automotive software OTA updates. It has the potential to advance the current understanding and implementation of software OTA updates in the automotive community.
- We offer crucial and unique insights based on pertinent evaluation metrics, such as update latency, transmission bandwidth, and successful rate. Evaluate the trade-off between these metrics with varied communication distances (up to 21 meters) and diverse sizes of vehicle clusters with heterogeneous computing units for the in-depth scalability analysis.
- We also explored and discussed the state-of-the-art research efforts and works for automotive software OTA updates in both academia and industry.

The rest of this paper is organized as follows: Sec. II reviews the background and related work on OTA. The building blocks of software OTA updates are presented in Sec. III. Extensive experimental results are shown in Sec. IV and finally we conclude the entire paper discussion in Sec. V.

II. Background and Related Work

A. Theoretical Exploration

The motivation behind adopting OTA updates from edge server-based Roadside Units (RSUs) to SDN-enabled vehicles (On-Board Units or OBUs) stems from several key factors. However, we will look into the advantages and disadvantages of the cloud and fog/edge-based OTA before diving deep into that.

1) Cloud based OTA Updates: The software updates are recently pushed to the network cloud/edge/fog nodes for further dissemination to the software-defined vehicles. Automakers deploy software updates to the cloud, where all connected vehicles can retrieve the latest software versions. These OTA updates can occur when the vehicles are stationary or in motion. However, certain drawbacks are associated with downloading updates from a remote cloud data center [1]. Due to the distance between the vehicle and the cloud, response times for downloading time-critical updates may need to be faster. Moreover, transmitting critical updates through globally connected channels exposes them to security vulnerabilities [12].

2) Edge-based OTA Updates: In contrast, edge Computing brings cloud functionality closer to data generation sources. It involves deploying edge nodes directly connected to cyber-physical devices like sensors and actuators. These edge nodes near the data-generating devices can process data locally, reducing response latency. Additionally, the edge nodes are distributed and decentralized, ensuring a more dependable and resilient system without a single point of failure [13]. Regarding OTA updates, edge computing outshines cloud computing in several aspects. The primary advantage of edge computing is it brings computing resources near to the vehicles, enabling faster and more efficient delivery of updates. By processing updates locally at the edge, edge computing reduces latency, ensuring timely updates even in time-critical scenarios [14]. Additionally, edge computing optimizes bandwidth usage by distributing updates locally, minimizing network congestion. It enhances reliability by allowing updates to be processed even in intermittent or unreliable network connectivity [15].

B. The Gap in Previous Work

OTA update technology plays an essential role in the entire lifecycle of SDVs, spanning design, development, manufacturing, and continuous usage over the vehicle's lifespan. Compared to conventional wired updates, the benefits of OTA updates for automotive software are outlined in [16], along with a high-level architecture for these updates. However, this research needs more specific information regarding the wireless medium, essential security mechanisms, and other technical aspects. A study conducted by multiple authors in [9], [17] has predominantly concentrated on remote software updates for vehicles and related security considerations. However, these studies would greatly benefit from exploring scenarios involving local updates performed within service centers and integrating advanced update mechanisms such as parallel updates [18].

Idrees et al. [17] presented a system that facilitates OTA updates by utilizing a Hardware Security Module (HSM) for tasks such as data encryption, key management, and ensuring data integrity across the wireless interface and all Electronic Control Units (ECUs) in the vehicle. However, it is essential to note that this implementation necessitates the installation of an HSM on each ECU, leading to a considerable escalation in costs associated with the system. Therefore, careful consideration should be given to the cost implications before implementing this approach. Nilsson et al. [19] propose a system where automotive OTA updates are facilitated by connecting the vehicle to a server via an internet link. The authors highlight key security aspects concerning data integrity and confidentiality in OTA updates but do not describe the wireless network being used within the network. In [20], a linear programming software update

scheme is proposed. The objective is to minimize handovers when pushing OTA updates from fog nodes. In this work, the fog node directly engaged all the vehicles available in its communication range. Thus, this could overburden the fog nodes and impact the performance of other delay-sensitive applications deployed at the fog.

III. Proposed Framework and Methodology

A. Framework Description

We propose a two-phase system model for OTA update dissemination in the SDV network. Let $E = e1, ..., en$ represent a set of n Edge-RSU units, and $V = v1, ..., vi$ represent a set of i vehicles, where $n < i$. In the first phase, vehicles in the lower tier, which require periodic updates to their application software, are involved. In the second phase, strategically positioned Edge-RSU units broadcast software updates to nearby vehicles based on their demands. Each vehicle has OBUs and storage capabilities, allowing it to install updates and relay them to neighboring vehicles. The timely dissemination of newer versions of software is critical to ensure software security and stability and delivering a positive user experience across the vehicular network.

1) Network Model: The SDN network operates within a routable networking environment. This network configuration enables wireless connectivity among mobile nodes, forming a self-configured and self-healing network without a fixed infrastructure. As the network topology undergoes frequent changes, the mobile nodes have the freedom to move randomly. Each node functions as a router, responsible for forwarding traffic to other designated nodes within the network. In this context, we assume that the spectrum assigned to vehicles is orthogonal, ensuring no collisions between the connected nodes and edge units.

2) Adversarial Model: The adversary model considers the system's Edge-RSU units (E) and vehicles (V). The adversary, represented by functions $Adv_E(S, A_E)$ and $Adv_V(S, A_V)$, can monitor and manipulate software updates. The Edge-RSU adversary $Adv_E(S, A_E)$ aims to compromise the updates' integrity, security, or functionality. In contrast, the vehicle adversary $Adv_V(S, A_V)$ aims to evade or bypass the security measures implemented by the Edge-RSU units. They may intercept, modify, or distribute unauthorized updates, threatening the system's security. Countermeasures, represented by the function $C(S, A_E, A_V)$, can mitigate these risks by integrating cryptographic mechanisms, secure protocols, verification techniques, and intrusion detection systems. In this adversarial setting, the objective functions $O_E(S, A_E)$ and $O_V(S, A_V)$ capture the adversaries' goals, seeking to maximize their advantage.

B. Quantitative Requirements of Automotive OTA Updates

1) Parameters: C_v is the computational capacity of vehicle V. S_v is the storage capacity and R_v represents communication range of vehicle V. U_f represents the size of update file f. Additionally network topology and connectivity information as well as security requirements are required.

2) Variables:

- Routing decisions: This defines the paths for update dissemination from Edge-RSU units to vehicles. Let X_ve be a binary variable indicating if vehicle V receives an update from E (1 if true, 0 otherwise).
- Scheduling decisions: Scheduling helps to determine the timing and order of updates for each vehicle. Let T_v the time at which vehicle V receives the update.
- Resource allocation: Assigns available bandwidth B and storage capacity S to vehicles for receiving and processing updates. Let, B_v the bandwidth allocated to vehicle V for receiving updates and S_vu be the storage space allocated to V for storing updates.
- Security measures: Specify the cryptographic algorithms, authentication protocols, and integrity verification techniques used during the update process.

3) Constraints:

- Bandwidth limitations: We have to ensure that the sum of the updated file sizes transmitted to vehicles does not exceed the available bandwidth. Bandwidth limitation is shown here as: $\sum U\_f{}^*C\_ve \leq B\_v$ for all vehicles v where U_f is the file size, C_v is the capacity and B_v is bandwidth.
- Communication Range: Vehicles must be within the communication range of an Edge-RSU unit to receive updates. $X\_ve \leq R\_v$ for all vehicles V and Edge-RSU E and X_ve be a binary variable indicating if V receives an update from E.
- Computational and storage capacities: Updates must fit within the computational and storage capacities of the vehicles. $\sum U\_f$ * $X_$v e $\leq C\_v$ for all vehicles V and $\sum U\_f$ * $X_ve \leq S_$vu for all vehicles V.
- Security constraints: Apply authentication and authorization during initial connection set up between the Edge E and vehicle V to prevent unauthorized updates.
- Real-Time Constraints: Time-sensitive updates must be delivered within specified deadlines. $T\_v \leq$ deadline for time-sensitive updates.
- Energy consumption: Limit the energy consumption of vehicles during the update process.

4) Objective Function:

- Minimize the overall update time, which includes update dissemination and installation time across all V.
- Minimize $\sum(T\_v - t\_e)$ for all vehicles v, where t_e is the time at which the update is available at E.

C. Proposed OTA Update Mechanism

The OTA update mechanism facilitates the distribution of software updates from Edge-RSU to vehicles in a wireless network. In this scenario, the updates are packaged as Docker images, which provide a lightweight and portable containerization format. When the edge-RSU in our OTA framework receives an update request, it is divided into multiple modules for parallel processing and reliable computation. The vehicle in our framework shown in Fig.

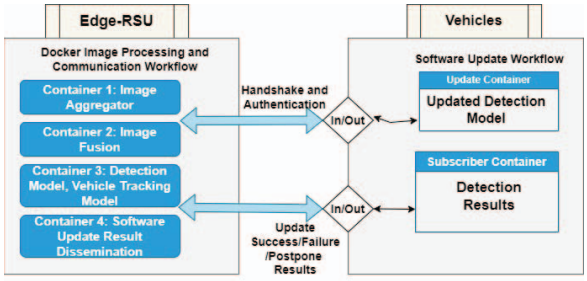


Fig. 2. Edge-RSU-Vehicle based Software Update Workflow

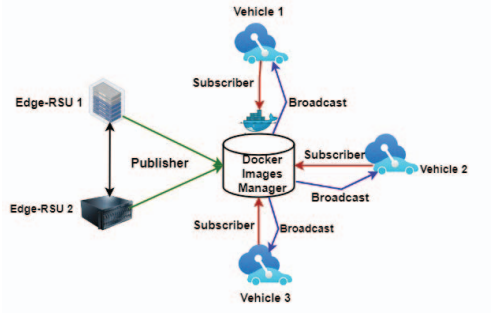


Fig. 3. Software Update Dissemmination using a Publish Subscriber Paradigm.

2 is equipped with separate lightweight components (as containers) for OTA update exposure and reception, ensuring isolation and controlled sharing. Upon receiving the update request, the edge node directs it to the relevant workflow components (as containers) for processing. This includes aggregating sensor data from multiple vehicles or traffic infrastructure, ensuring data consistency, and applying fusion algorithms to combine the data effectively. The fused data is then passed to an object detection module, which identifies the locations and types of objects based on the nature of the data and the current load on the edge node. Furthermore, the generation and consumption of updates between vehicles and edge nodes are decoupled using a publish-subscribe subsystem shown in Fig. 3. This subsystem employs a broker that maintains a list of topics, allowing vehicles to publish their sensor data through the broker. By leveraging this subsystem, edge nodes can efficiently handle multiple workflows by retrieving the necessary data from the appropriate containers, reducing the overall workload. Both vehicles and edge nodes subscribe to specific topics managed by the broker. When a new update is received, the broker notifies the subscribers using notification method tailored to the characteristics of the update. The broker can be distributed across multiple edge nodes for improved scalability and reliability.

1) Prepare the Update Server: Create an update server as a centralized repository for hosting the latest software updates. This server ensures the secure and reliable storage and distribution of updates to the connected systems. The updates are packaged as Docker images, encompassing the essential components and configurations required for the updated software. The server setup process includes configuring the network settings and deploying the server.

2) Update Trigger Mechanism: The update trigger mechanism enables the OTA update server to initiate software updates on vehicles in a wireless network. Let V be the set of vehicles in the network, U be the set of software updates available on the OTA update server, and t be the update trigger function that sends a notification to the vehicles for initiating the update process. The update trigger mechanism can be represented as t: $U \rightarrow V$. This function t maps each software update $u \in U$ to the set of V that need to receive the update. When a new update u is available on the update server, the update trigger function t is invoked to send a notification to the vehicles in V, indicating that an update is available and should be retrieved and installed.

3) Update Broadcasting and Deployment: The update broadcasting and deployment mechanism involves the Edge-RSU units broadcasting software updates to nearby (V). These (E) units act as the distribution points for the updates and are responsible for delivering the updates to nearby vehicles (V). Each Edge-RSU unit e in E performs the broadcasting function $B(e, v)$, where v is a specific vehicle in V, ensuring that the updates reach their intended recipients. Additionally, the deployment function $DF(e, V)$ captures the process of an E deploying updates to a set of V. By utilizing this mechanism, the E units efficiently broadcast and deploy OTA updates, enabling connected vehicles to receive and install the latest software updates.

4) Rollback and Recovery Mechanism: Implement a mechanism for rollback and recovery in case of update failures or system instability after updates. This mechanism should allow for reverting to the previous software version and recovering from potential issues introduced by the updates. The rollback and recovery mechanism involves monitoring the OTA update process and detecting failures using the failure detection function $F(t)$. When a failure is detected (F(t) = true), the rollback function R(t) is invoked to revert the software to the previous version $U-previous$.

5) Logging and Monitoring: Logging and monitoring mechanism is crucial in overseeing the OTA update process. It systematically records events and actions in a log file, denoted by L. The log file captures essential information about the software update process, including the sequence of events and any notable actions taken. With each new event E, the log file is updated using the logging function $L(t)$ = $L(t-1) \cup E(t)$, where E(t) represents the event occurring at time t. This ensures a comprehensive record of the OTA update process, facilitating analysis and troubleshooting if any issues arise. In addition to logging, the OTA system incorporates a monitoring function, denoted by M, which continuously evaluates the progress and performance of the update process. The monitoring function, represented by $M(t) = f(L(t))$, leverages the log file at time t to provide insights and assessments. By analyzing the recorded information, the monitoring function identifies potential issues, tracks the update's progress, and enables timely intervention

Fig. 4. Two categories of hardware. Subfigure (a)-(b) shows the Intel Fognodes, and Apple Macbooks, respectively.

if necessary. This proactive approach ensures that the OTA update process remains robust, reliable, and capable of detecting anomalies for a swift resolution.

D. Hardware Setup

Our OTA framework utilizes three types of hardware, namely Intel Fognode, Apple Macbook Air M1 and MacBook M2 Pro, as depicted in Fig. 4. The Edge-RSU is equipped with an Intel Fognode, which offers programmability and configuration options for diverse use cases, ensuring consistent throughput for different workloads [21]. The MacBook workstation, equipped with high-quality components including 8 & 32 core GPU and 16 &32 GB memory delivers cluster-level performance for demanding applications. Further details about these devices are provided in Table. II.

TABLE II
CONFIGURATION INFORMATION OF TWO HARDWARE DEVICES.

	Intel Fognode	**Macbook Air**	**Macbook Pro**
CPU	Intel Xeon E3-1275 v5	M1	M2 Max
GPU	NONE	8 Core	32 Core
Frequency	3.6 GHz	3.2 GHz	3.6 GHz
Memory	32 GB	16 GB	32 GB
OS	Ubuntu 16.04.6 LTS	MacOS Ventura 13.6	MacOS Ventura

E. Model Description

We conducted our OTA experiment using three machine learning models: Faster R-CNN, ResNet-18, and ResNet-50. Following is their description:

1) Faster R-CNN: Faster R-CNN is a two-stage object detection model that consists of a Region Proposal Network (RPN) and a Fast R-CNN network. The RPN generates potential object proposals by sliding a small network over the convolutional feature map, predicting regions likely to contain objects [22]. These proposals are fed into the Fast R-CNN network, which performs classification and bounding box regression. The model uses shared convolutional features to extract features from the entire image, enabling efficient computation.

2) ResNet-18: ResNet-18 [23] is a deep convolutional neural network architecture comprising 18 layers, including convolutional pooling, and fully connected layers. It introduces the concept of residual connections, allowing the network to learn residual mappings instead of directly learning the desired output. This helps alleviate the vanishing gradient problem and enables the training of very deep networks. ResNet-18 uses residual blocks with two convolutional layers and shortcut connections that skip one or more layers, allowing information to bypass the layers and propagate through the network more effectively [24].

3) ResNet-50: Similar to ResNet-18, ResNet-50 [25] is also a deep convolutional neural network architecture comprising 50 layers, including convolutional, pooling, and fully connected layers. It is an extension of the ResNet-18 model with more layers and increased complexity. Like ResNet-18, ResNet-50 also incorporates residual connections to address the vanishing gradient problem and facilitate the training of deeper networks.

F. Container-based OTA Update

Container-based OTA software updates can be a potential solution for facilitating software updates in the automotive industry. Containers are built from layered images representing specific data, software, hardware, and network configuration parameters [26]. Each container image consists of multiple layers, encompassing all necessary software libraries, binaries, and configuration settings. In this paper to remove the latency constraints we adopt Delta file based OTA update where instead of updating the whole software binary files we are updating the delta files.

1) Delta file based OTA Update: Delta file flashing involves comparing the base file with the new version file to generate a smaller delta file, significantly reducing the update size [27]. This method offers faster transmission, saving up to 90% of the transmission time compared to complete binary updates. It requires less storage and utilizes a patching algorithm to overwrite old and new data.

IV. EXPERIMENTAL DESIGN AND RESULTS ANALYSIS

A. Experimental Setup

1) Intel Fognode as Edge-RSU: The Intel Fognode is a central component of the experimental setup, functioning as the Edge-RSU responsible for hosting the software updates. It provides the necessary computational resources and creates a Wi-Fi hotspot for vehicle connectivity. For our experiment we used a 100Mbps WiFi connection.

2) Device Connection and Logging: A logging mechanism is implemented to track device connections and disconnections to the RSU's Wi-Fi hotspot. Whenever a device connects or disconnects, the log records the corresponding MAC address of the device. The entire log output is stored in a designated log file for analysis.

3) MAC Address Extraction: A script is employed to monitor changes in the log file size. Upon detecting a change, the script retrieves the last line of the log file, which typically corresponds to the connection status of a new device. This line contains the device's MAC address.

4) IP Address Retrieval: With the MAC address obtained, the experimental setup employs the "arp" (address resolution protocol) command to determine the associated IP address of the connected device. This information is crucial for subsequent communication and software update delivery.

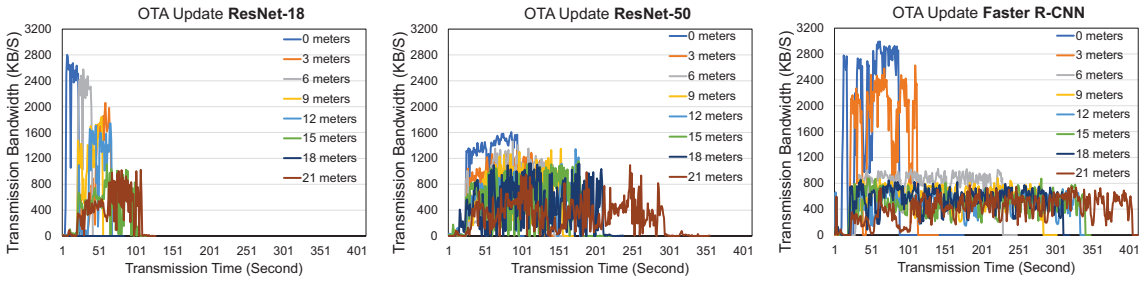
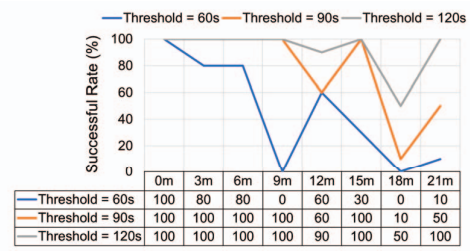


Fig. 5. Transmission bandwidth of OTA update for ResNet-18, ResNet-50, and Faster R-CNN.

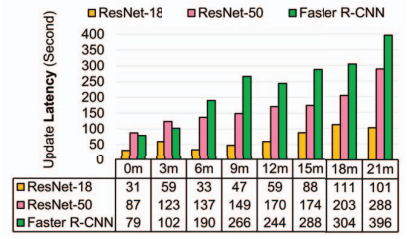


	0m	3m	6m	9m	12m	15m	18m	21m
Threshold = 60s	100	80	80	0	60	30	0	10
Threshold = 90s	100	100	100	100	60	100	10	50
Threshold = 120s	100	100	100	100	90	100	50	100

Fig. 6. The change curves of successful rate (%) when update ResNet-18.

5) Device Readiness Verification: A ping test ensures the connected device is ready to receive the software update. The setup verifies the device's responsiveness by pinging, and a successful ping response confirms the device's operational state and readiness for the update.

6) Software Update Files: The software update package consists of several components, including a Dockerfile for building a new Docker image, a Python script for executing a new model, the model weights, and a Bash script to initiate the Docker build and run the updated image. For file transfer between the Edge-RSU (Intel Fognode) and connected vehicle (Macbook), the "nc" (netcat) command is utilized. Here, "nc "operates as a versatile networking utility, functioning as both a client and a server for bidirectional communication over the network. In this setup, "nc" operates in server mode (nc-l) and listens on a specific port (port 1234). When a device connects to the Wi-Fi hotspot hosted by the Intel Fognode, the "nc" command establishes a connection, allowing the Edge-RSU (Intel Fognode) to transmit the software update files to the connected vehicle (Macbook).



	0m	3m	6m	9m	12m	15m	18m	21m
ResNet-18	31	59	33	47	59	88	111	101
ResNet-50	87	123	137	149	170	174	203	288
Faster R-CNN	79	102	190	266	244	288	304	396

Fig. 7. OTA update latency of ResNet-18, ResNet-50, and Faster R-CNN under different distances from 0 to 21 meters.

The received data is processed through a pipeline of orders, starting with compression using the "gzip" command to reduce file size for efficient transmission. The compressed data is then extracted using the "tar" command to restore the original files from the compressed archive.

7) Definition of OTA Update Failure: Given the complexity of OTA updates, a universally accepted definition of OTA update failure does not exist. Various factors, such as power outages, faulty batteries, connectivity loss, and user interference, can contribute to the failure of OTA updates and are commonly recognized as frequent causes of failures [28]. During our development of the OTA mechanism, it became apparent that these failures can hinder the successful reception, installation, or execution of updated vehicle software, resulting in potential functional inconsistencies, malfunctions, or even system instability. *In our study, we consider an OTA update to fail when network connectivity issues arise due to bandwidth limitations*. The vehicular network operates in a dynamic environment where vehicles constantly move. As the distance increases between the Edge-RSU station and the target vehicles, when triggers the updates, the time required for deploying updates significantly extends. This could potentially lead to update failures. Furthermore, simultaneously executing multiple OTA updates on different vehicles increases the risk of failures.

8) Definition of OTA Update Latency: OTA update latency refers to the time it takes to complete an update process. It is influenced by factors such as the size of the update package and the distance between the source (Edge-RSU) and the target (vehicle). Generally, the OTA update latency increases as the model size or distance increases. This latency is an important metric to consider when assessing the efficiency and effectiveness of OTA update mechanisms, as longer latencies may lead to extended periods of potential system operational disruptions.

B. Experiment Groups

The experimental procedure involved conducting a series of software OTA update experiments from the Edge-RSU (Intel Fognode) to a vehicle (MacBook) at varying distances. Distances of 0, 3, 6, 9, 12, 15, 18, and 21 meters were selected to examine the impact of distance on the OTA update process. We also used three distinct sizes of fundamental vehicle models, *Faster R-CNN (175.2 MB), ResNet-*

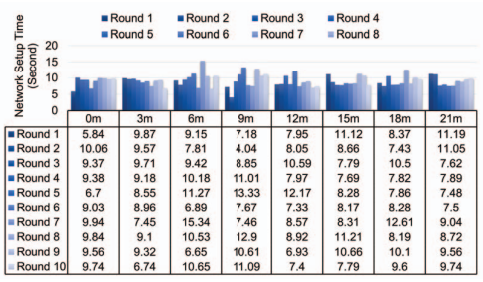
18 (46.8 MB), and ResNet-50 (102.5 MB), to assess the update performance across the eight distances. Throughout each experiment, the bandwidth was logged on the vehicle's log file, and the time taken for the update.

1) Correlation among OTA Update Time, Model Size, Distance and Transmission Bandwidth: To understand the correlation between OTA update time vs. the update model size and the varying distances between Edge-RSU and vehicles, we set the placement of Edge-RSU and the vehicle at different distances varying from 0, 3, 6, 9, 12, 15, 18, and 21 meters are analyzed, and their impact on the transmission bandwidth speed during OTA update is shown in Fig. 5.

The experiment confirms that the OTA update time increases as the distance between the Edge-RSU and the vehicle increases. The influence of model size on the update time was also observed, with larger models, such as Faster R-CNN, requiring more time than smaller models, like ResNet-18, to update. For example, at a distance of 21 meters, ResNet-18 took approximately 140 seconds to complete the update, while Faster R-CNN took around 410 seconds, almost four times longer. Similarly, ResNet-50, a larger model than ResNet-18 but smaller than Faster R-CNN, took approximately 350 seconds to complete the update at the same distance. Besides we also monitored the bandwidth throughout the experiment to identify variations or fluctuations during the OTA update. The analysis revealed correlations between distance, model size, update time, and bandwidth speed. Fig. 5 demonstrates that at lower distances, such as 0 or 3 meters, all three models exhibited higher transmission bandwidth, with ResNet-18 and Faster R-CNN peaking at around 2800 KB/s and ResNet-50 at 1600 KB/s. In contrast, at higher distances (12, 15, 18, and 21 meters), the transmission bandwidth remained consistent between 400 and 800 KB/s.

Successful Rate Exploration: As shown in Fig. 6, during our OTA update experiment, we set threshold times (60s, 90s, and 120s) to determine update success or failure. If the update process exceeded the threshold time, it was classified as a failure. We used the ResNet-18 model for this experiment, as shown in Fig. 6, and at the 60s threshold, we observed failures at 9m and 18m distances, indicating the process needed longer to complete the update. The successful rate increased as the threshold increased to the 90s and 120s. However, successful rates are varied. For example, it is around 10% at 18m and 50% at 21m for the 90s, while for the 120s threshold, it is 50% at 18m. These failures were attributed to network connectivity and software compatibility issues. Addressing these challenges is crucial for ensuring successful OTA updates. To be short, this experiment provided valuable insights into the impact of distance and model size on OTA update time. As distance and size increase, OTA update takes longer, and the transmission bandwidth speed gets slower.

2) Correlation between OTA Update Latency Vs. Model Size and Distance: We also investigated the OTA update



	0m	3m	6m	9m	12m	15m	18m	21m
Round 1	5.84	9.87	9.15	7.18	7.95	11.12	8.37	11.19
Round 2	10.06	9.57	7.81	4.04	8.05	8.66	7.43	11.05
Round 3	9.37	9.71	9.42	8.85	10.59	7.79	10.5	7.62
Round 4	9.38	9.18	10.18	11.01	7.97	7.69	7.82	7.89
Round 5	6.7	8.55	11.27	13.33	12.17	8.28	7.86	7.48
Round 6	9.03	8.96	6.89	7.67	7.33	8.17	8.28	7.5
Round 7	9.94	7.45	15.34	7.46	8.57	8.31	12.61	9.04
Round 8	9.84	9.1	10.53	12.9	8.92	11.21	8.19	8.72
Round 9	9.56	9.32	6.65	10.61	6.93	10.66	10.1	9.56
Round 10	9.74	6.74	10.65	11.09	7.4	7.79	9.6	9.74

Fig. 8. Network setup time.

latency in relation to varying model sizes and distances, as depicted in Fig. 7. The analysis revealed a clear correlation between update latency vs. model size, and distance. As the model size and distance increased, the latency also increased. For example, as shown in the Fig. 7, the latency for completing the OTA update of ResNet-18 increased from 31 seconds at 0 meters to approximately 101 seconds at 21 meters. Additionally, the latency increased significantly with larger model sizes. At a distance of 9 meters, the latency for ResNet-18 was 47 seconds, while for ResNet-50, it was 149 seconds, and for Faster R-CNN, it was around 266 seconds. These findings support the intuition that larger model sizes and longer distances result in longer OTA update latencies.

3) Network Setup Time: Network setup time refers to the duration required to establish a network connection between the update server and the target vehicle while initiating an OTA update. It involves configuring network parameters, establishing communication channels, and verifying connectivity. The setup time can vary based on network infrastructure, signal strength, and configuration complexity. Our experiment measured the network setup time while updating ResNet-18 across different distances (0m to 21m) as shown in Fig. 8. We conducted ten runs at each distance and observed a consistent network setup time of around 10 seconds, with a maximum of approximately 15 seconds at a distance of 6m. A shorter network setup time ensures a quicker start to the update process and facilitates efficient communication between the update server and the vehicle.

4) OTA Update based on Heterogeneous Computing Units: We analyzed the downlink bandwidth, which measures the data transfer speed from the Intel Fognode acting as the Edge-RSU to the Macbooks serving as vehicles. Three groups were formed with varying configurations, as depicted

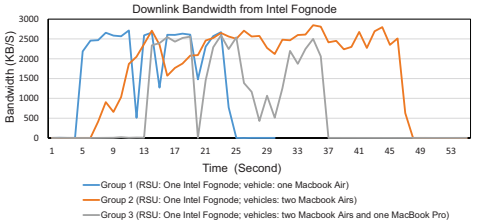


Fig. 9. Three group's downlink bandwidth from RSU.

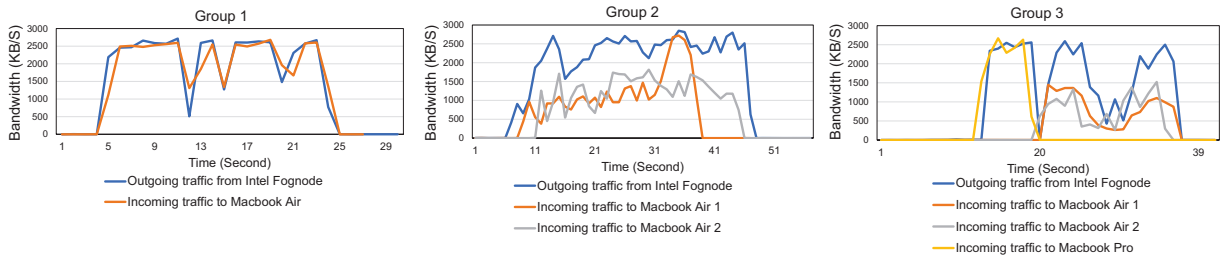


Fig. 10. Outgoing and incoming traffic for ResNet-18 OTA update.

in Fig. 9: Group 1 consisted of one Intel Fognode and one Macbook Air, Group 2 had one Intel Fognode and two Macbook Air, and Group 3 had one Intel Fognode, two Macbook Air, and one Macbook Pro. The OTA update for Group 1 was completed in 32 seconds, while for Group 2, Macbook Air 1 took 49 seconds, and Macbook Air 2 took 47 seconds. However, Group 3 exhibited a different trend, with the OTA update completing in 29 seconds for Macbook Air 1, and 27 seconds for Macbook Air 2, but failing to complete on the Macbook Pro around 9 seconds into the update. The peak bandwidth speed observed in all three groups was approximately 2800 KB/S, with the lowest observed speed being around 0 KB/S. This experiment provided valuable insights into the impact of factors such as vehicle performance and the number of vehicles on downlink bandwidth performance.

To comprehensively evaluate the importance of OTA updates across heterogeneous computing units, we conducted experiments involving the same three groups of varying computing units using the same ResNet-18 model. The OTA update was initiated again from the Edge Fognode, and the distribution of bandwidth traffic among the connected vehicles was observed, as depicted in Fig.10. In Group 1, where a single vehicle (Macbook Air 1) received the update, both the incoming bandwidth on the vehicle and the outgoing bandwidth from the Intel Fognode followed the same pattern. The peak outgoing bandwidth speed and incoming bandwidth traffic were measured at approximately 2800 KB/S, and the update was completed in around 32 seconds. However, group 2, where we used two vehicles with identical computing units, showed that both vehicles' incoming bandwidth traffic equally shares the bandwidth traffic 1300 KB/S but vehicle 1 experienced a sudden surge in bandwidth towards the end (2800 KB/S), resulting in an earlier update completion time of around 41 seconds compared to the other vehicle, which took approximately 50 seconds. In Group 3, an additional vehicle (Macbook Pro) was included in the experiment. Due to its superior hardware performance, the transfer of files was completed in a significantly shorter time of 20 seconds with a peak bandwidth speed of around 2700 KB/S. Although all three experimental groups finished the updates, vehicle one and vehicle two took longer to complete their respective updates than vehicle three.

The experiment demonstrated that vehicles with better hardware performance can receive a larger share of the network traffic. While the resources were shared equally among the two Macbook Airs in the experiment, the Macbook Pro obtained a more significant portion of the bandwidth due to its superior capabilities. These findings highlight the significance of considering the heterogeneity of computing units in OTA updates and its potential impact on network traffic distribution.

C. Observations and Insights

We presented OTA update for three models at varying distances to observe the impact of distance on OTA update time and how it also impacts the bandwidth traffic. Some interesting observations are listed, including supporting evidence and reasons to explain the observed trends and practical implications for researchers and domain experts.

Observation 1: During the analysis of the collected data, it was observed that the bandwidth experienced an irregular pattern in relation to the distance. Typically, as the distance increased, the bandwidth decreased, indicating a decrease in data transmission speed. However, there were certain cases where this pattern did not hold, and deviations were observed.

Observation 2: An additional phenomenon contradicted the anticipated pattern of decreasing bandwidth with increasing distance. In certain instances, it was observed that the bandwidth exhibited unexpected behavior. Specifically, the bandwidth was lower at shorter distances, while it appeared to be faster at longer distances. This anomaly introduced complexity to the relationship between distance and bandwidth during the software update process, deviating from the conventional understanding that greater distance corresponds to reduced bandwidth.

Observation 3: The bandwidth measurements occasionally exhibited substantial fluctuations, with intermittent instances where the bandwidth dropped to zero during the update process. These occurrences were classified as failed updates, suggesting the presence of potential obstacles or interruptions in the wireless connection.

Observation 4: OTA updates using delta files offer significant advantages over complete binary updates. By only transmitting the differences between the old and new versions, delta updates have smaller file sizes, resulting in

faster transmission times and reduced bandwidth usage. In our experiment, we observed a substantial improvement in update efficiency. While updating the binary file of Yolo [29] model took approximately 2 hours and 30 seconds, the three model we used for updating using only delta file was completed around 7 minutes. This highlights the efficiency and effectiveness of delta updates in OTA deployments.

The existence of diverse patterns and intermittent bandwidth irregularities impacts the software OTA update process. Various elements, including environmental conditions, signal interference, and other variables, may contribute to the observed variations and deviations from the anticipated distance-based pattern.

V. Concluding Remarks

In conclusion, our study addresses the imperative need for SDVs and implmement software OTA updates in the automotive industry. We have made significant contributions to the field by pioneering the implementation of an edge-assisted framework for automotive OTA updates. By carefully considering factors such as vehicular software models, vehicle computing units, communication distances, and vehicle cluster sizes, we have provided valuable insights and evaluation metrics for OTA updates. It may significantly contribute to the future development and implementation of SDVs and OTA updates in the automotive industry.

References

[1] A. Mahmood, W. E. Zhang, and Q. Z. Sheng, "Software-defined heterogeneous vehicular networking: The architectural design and open challenges," *Future Internet*, vol. 11, no. 3, p. 70, 2019.

[2] Y. Ma, Z. Wang, H. Yang, and L. Yang, "Artificial intelligence applications in the development of autonomous vehicles: A survey," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 2, pp. 315–329, 2020.

[3] S. R. Pokhrel, "Software defined internet of vehicles for automation and orchestration," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3890–3899, 2021.

[4] C. Jiacheng, Z. Haibo, Z. Ning, Y. Peng, G. Lin, and S. X. Sherman, "Software defined internet of vehicles: architecture, challenges and solutions," *Journal of communications and information networks*, vol. 1, no. 1, pp. 14–26, 2016.

[5] P. Nikbakht Bideh and C. Gehrmann, "Rosym: Robust symmetric key based iot software upgrade over-the-air," in *Proceedings of the 4th Workshop on CPS & IoT Security and Privacy*, 2022, pp. 35–46.

[6] E. Cebel, N. Donum, and H. Karacali, "Platform independent embedded linux ota method," *The European Journal of Research and Development*, vol. 2, no. 4, pp. 243–252, 2022.

[7] B. Shen, "Competitive strategies for ota services: Adapting the strategic clock for tesla," *Highlights in Business, Economics and Management*, vol. 11, pp. 26–32, 2023.

[8] S. Halder, A. Ghosal, and M. Conti, "Secure over-the-air software updates in connected vehicles: A survey," *Computer Networks*, vol. 178, p. 107343, 2020.

[9] M. Khurram, H. Kumar, A. Chandak, V. Sarwade, N. Arora, and T. Quach, "Enhancing connected car adoption: Security and over the air update framework," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*. IEEE, 2016, pp. 194–198.

[10] C. Sun, R. Xing, Y. Wu, G. Zhou, F. Zheng, and D. Hu, "Design of over-the-air firmware update and management for iot device with cloud-based restful web services," in *2021 China Automation Congress (CAC)*. IEEE, 2021, pp. 5081–5085.

[11] S. Al Blooshi and K. Han, "A study on employing uptane for secure software update ota in drone environments," in *2022 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*. IEEE, 2022, pp. 1–6.

[12] A. Ghosal, S. Halder, and M. Conti, "Stride: Scalable and secure over-the-air software update scheme for autonomous vehicles," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[13] O. Rana, K. Fizza, L. Bittencourt, and N. Auluck, "Pashe: privacy aware scheduling in a heterogeneous fog environment," 2019.

[14] A. W. Malik, A. U. Rahman, A. Ahmad, and M. M. D. Santos, "Over-the-air software-defined vehicle updates using federated fog environment," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 5078–5089, 2022.

[15] M. Al Maruf, A. Singh, A. Azim, and N. Auluck, "Faster fog computing based over-the-air vehicular updates: A transfer learning approach," *IEEE Transactions on Services Computing*, 2021.

[16] R. v. Stokar, "Updating car ecus over-the-air (fota), 2013."

[17] M. S. Idrees, H. Schweppe, Y. Roudier, M. Wolf, D. Scheuermann, and O. Henniger, "Secure automotive on-board protocols: A case of over-the-air firmware updates," in *Communication Technologies for Vehicles: Third International Workshop, Nets4Cars/Nets4Trains 2011, Oberpfaffenhofen, Germany, March 23-24, 2011. Proceedings 3*. Springer, 2011, pp. 224–238.

[18] I. Hossain and S. M. Mahmud, "Analysis of a secure software upload technique in advanced vehicles using wireless links," in *2007 IEEE Intelligent Transportation Systems Conference*. IEEE, 2007, pp. 1010–1015.

[19] D. K. Nilsson, P. H. Phung, and U. E. Larson, "Vehicle ecu classification based on safety-security characteristics," in *IET Road Transport Information and Control-RTIC 2008 and ITS United Kingdom Members' Conference*. IET, 2008, pp. 1–7.

[20] K. Fizza, N. Auluck, A. Azim, M. A. Maruf, and A. Singh, "Faster OTA updates in smart vehicles using fog computing," in *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion*, 2019, pp. 59–64.

[21] S. Biookaghazadeh, M. Zhao, and F. Ren, "Are {FPGAs} suitable for edge computing?" in *USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18)*, 2018.

[22] M. Maity, S. Banerjee, and S. S. Chaudhuri, "Faster r-cnn and yolo based vehicle detection: A survey," in *2021 5th international conference on computing methodologies and communication (ICCMC)*. IEEE, 2021, pp. 1442–1447.

[23] X. Ou, P. Yan, Y. Zhang, B. Tu, G. Zhang, J. Wu, and W. Li, "Moving object detection method via resnet-18 with encoder–decoder structure in complex scenes," *IEEE Access*, vol. 7, pp. 108 152–108 160, 2019.

[24] Q. A. Al-Haija, M. A. Smadi, and S. Zein-Sabatto, "Multi-class weather classification using resnet-18 cnn for autonomous iot and cps applications," in *2020 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 2020, pp. 1586–1591.

[25] B. Koonce and B. Koonce, "Resnet 50," *Convolutional Neural Networks with Swift for Tensorflow: Image Recognition and Dataset Categorization*, pp. 63–72, 2021.

[26] Y. Wang and Q. Bao, "Adapting a container infrastructure for autonomous vehicle development," in *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2020, pp. 0182–0187.

[27] M. Steger, C. A. Boano, T. Niedermayr, M. Karner, J. Hillebrand, K. Roemer, and W. Rom, "An efficient and secure automotive wireless software update framework," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 5, pp. 2181–2193, 2017.

[28] P. Dakić and M. Živković, "An overview of the challenges for developing software within the field of autonomous vehicles," in *7th Conference on the Engineering of Computer Based Systems*, 2021, pp. 1–10.

[29] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.