High Definition Map Data Optimization for Autonomous Driving in Vehicular Named Data Networks

Daniel Mawunyo Doe*, Dawei Chen†, Kyungtae Han†, Haoxin Wang‡, Jiang Xie§, and Zhu Han*
*Electrical and Computer Engineering Department, University of Houston, Houston, TX, USA
†InfoTech Labs, Toyota Motor North America R&D, Mountain View, CA, USA
‡Department of Computer Science, Georgia State University, GA, USA
§Department of Electrical and Computer Engineering, The University of North Carolina at Charlotte, Charlotte, NC, USA

***Abstract*—High-definition (HD) map is an essential building block in the autonomous driving era, which enables fine-grained environmental awareness, exact localization, and route planning. However, because HD maps include rich, multidimensional information, the volume of HD map data is enormous, making it expensive and time-consuming to transmit on vehicular networks. Therefore, in this paper, we propose a data optimization scheme for effective HD map updates in vehicular named data networking (NDN) scenarios. We formulate the HD map data optimization problem as a convex optimization problem and solve it with modified convolutional neural networks (CNNs) from YOLOX's real-time object detection system. Specifically, we modify the YOLOX object detection algorithm to detect and compress redundant pixels in local map data before transmission to the MEC server. To deploy our proposed scheme, we construct a vehicular NDN environment for data collection, processing, and transmission using the CARLA simulator and robot operating system 2 (ROS2). Extensive simulations show that our proposed scheme can significantly reduce the transmission data size and time by $48.25\% - 65.78\%$ and $46.85\% - 78.84\%$ compared with state-of-the-art HD map update techniques like RLSS, Pro-RTT, and Loss-based systems.**

***Index Terms*—High definition map, data optimization, object detection, named data networking, and vehicular networks.**

I. Introduction

A. Background and Motivation

With the rapid advancement of mobile communications, vehicular sensing technologies, and autonomous driving, the Internet of autonomous vehicles (AVs) has become a popular topic [1], [2]. Recent autonomous vehicles can establish their precise positions and design collision-free paths using high-definition (HD) maps. HD maps provide more trustworthy sensing capabilities and help the autonomous driving decision-making layer, where latency is essential. However, due to advanced processing, recent HD map updates suffer from low latency. For example, the HD map data capacity is relatively large compared to the typical electronic map, making it challenging to generate, transmit, or store onboard.

Named data networking (NDN) is a potential future networking architecture in which each network component is considered to be an entity. NDN has tremendous potential for the automobile network, such as easing user mobility, data sharing, data naming, name resolution, and a naming-based route forwarding strategy [3]. NDN can effectively minimize data retrieval delays due to request aggregation, allowing vehicles to obtain or exchange data through a single data connection procedure. However, NDN uses floods to acquire data, which causes excessive vehicle traffic and high data collection costs when data quantities are enormous. Therefore, investigating map data optimization techniques is an important and relevant problem.

Crowdsourced data has recently garnered much attention for HD map updates, as seen in [4] and [5]. Crowdsourced data is road observation data collected by low-cost crowdsourcing devices, which often include a camera and a global positioning system (GNSS) sensor [6]. Crowdsourcing devices are put on automobiles that travel the same routes regularly, making a massive amount of environmental data publicly available [3] and [7]. The fundamental downside of crowdsourcing data is its high level of uncertainty, especially in fast-moving situations. Also, when the vehicle population grows, the throughput of crowdsourced data reduces dramatically due to the volume of data gathered [8].

Vehicle wireless communication, processing, and caching capabilities have lately improved considerably in [9]. The HD map update task may be subdivided into several subtasks and processed via vehicular distributed computing. First, idle computing resources in vehicles are fully exploited, which may improve resource usage and cloud server performance. Second, in vehicular NDN, using vehicles as data collectors and processors can reduce the transfer of large amounts of raw environmental data while decreasing overall system latency [10]. Extensive research has been undertaken on distributed computing, as shown in [11] and [12]. For example, [11] and [12] worked on improving input data transfer and task allocation in wired data centers (DCs) to reduce inter-DC traffic, increase throughput, and reduce delays. Furthermore, reducers and route selection are adjusted to decrease transmission costs in wireless sensor networks [13]. However, because the input data for the HD map update is large, the data transmission scenarios are not practical for NDN when crowd-sensing is employed.

Most existing studies on HD map updates have the common limitation of ignoring the optimization of HD map data before transmission. This oversight can have significant consequences, including inaccurate navigation, high resource utilization, and decreased efficiency and reliability in vehicular data networks. For example, [14] proposed a probability-based handover approach to choose a new data source by monitoring the round-trip time, which decreases the frequency

978-1-5386-7462-8/23/$31.00 ©2023 IEEE

of handovers for improved HD map updates. From [15], due to the imbalance in the dataset, [16] proposed an HD-Map-guided rapidly-exploring random tree (HDM-RRT) by combining an HD-Map and a sampling-based approach to quickly obtain high-quality and feasible map updates in complex campus scenarios. [17] suggested a reinforcement learning-based data source selection method (RLSS) for selecting HD map data sources in vehicular NDN. However, these methods fail to consider the optimization of HD maps before transmission. As a result, optimizing HD map data before transmission in vehicular NDN environments is a problem that needs to be solved.

B. Contributions

In this paper, we propose an HD map data optimization approach for autonomous driving in vehicular NDN. First, we formulate the HD map data optimization problem as a convex optimization problem and solve it with modified convolutional neural networks (CNNs) from YOLOX's real-time object detection system. Specifically, we modify the YOLOX object detection algorithm in our map data optimization to detect and compress superfluous pixels in onboard map data for transmission to the multi-access edge computing (MEC) server via the roadside unit (RSU). To deploy our proposed scheme, we construct a vehicular NDN simulation environment for our experiment, employing the CARLA simulation environment combined with the robot operating system 2 (ROS2). And then, we conduct extensive simulations to confirm the performance advantages generated by our proposed map data optimization approach. The significant contributions of this work are summarized as follows:

In summary, the following are the significant contributions of this paper:

- Our research is one of the first to explore the optimization of HD map data before transmission in vehicular NDN scenarios. We propose optimizing the HD map data to remove unnecessary pixels to reduce data size before transmission.
- We formulate the HD map data optimization problem as a convex optimization problem and solve it with modified convolutional neural networks (CNNs) from YOLOX's real-time object detection system.
- We perform extensive simulations to validate the performance gains achieved by our proposed scheme. The simulation results show that our approach can significantly reduce the transmission data size and time by $48.25\% - 65.78\%$ and $46.85\% - 78.84\%$ compared to state-of-the-art techniques like RLSS, Pro-RTT, and Loss-based systems.

The following is an overview of the paper's structure. The system model is described in Section II. Section III presents the YOLO-based map data optimization to solve the data size reduction issue. Section IV discusses the simulation results and analysis. Finally, Section V concludes our discussion.

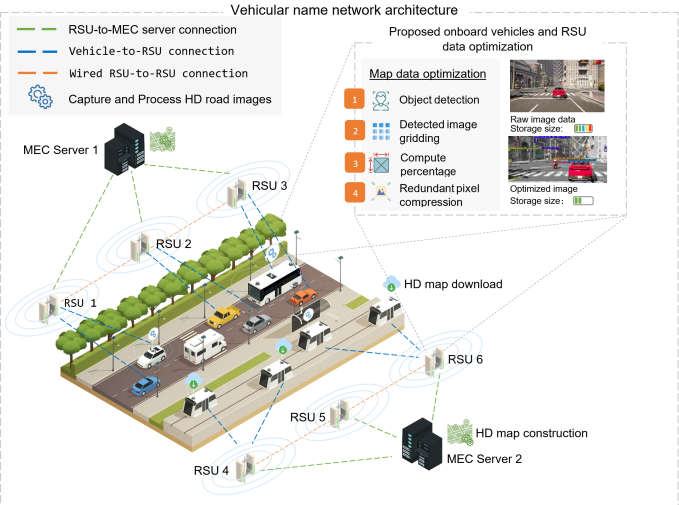


Fig. 1: Vehicular network architecture with proposed latency optimization scheme.

II. System Model

A. System Overview

We explore a hierarchical architecture consisting of vehicles, roadside units (RSUs), and MEC servers for our HD map update model [18]. We simulate several vehicles equipped with sensors, communication, computation, and caching resources in our CARLA environment. Then, we install RSUs along the route with sensors, such as high-definition cameras to collect environmental data. Each RSU is co-located with a MEC server to ensure sufficient computational capabilities for constructing HD maps. Assume that environmental data are continually gathered and stored locally for a length of time (i.e., the duration of the HD map update). At the start of each update period, the vehicles capture images of their surroundings for constructing HD maps. And then, using the YOLOX object detection framework, we execute onboard object recognition on image data acquired by each vehicle. Redundant image pixels with no detection (e.g., bounding boxes), such as those with roadside trees, sky, clouds, airplanes, distant buildings, roadside blocks, etc., are compressed to minimize image data size.

Moreover, each target cell's RSUs gather vehicle data such as location, speed, sensing range, wireless transmission capabilities, computing capability, and optimum picture data size. The optimized image data from vehicles is transmitted to a high-capacity MEC server, which processes the data to construct HD maps for AVs. Vehicles with low computing capabilities rely significantly on the computation of the surrounding vehicles or RSU for map data optimization and only have an update mode toggled on at any target cell. Also, vehicles departing the target cell use wireless networks to send interim map data to the present RSUs, which is passed to other RSUs through wired fronthaul links for distributed map accessibility. The vehicular network architecture with our suggested map data optimization strategy is shown in Fig. 1.

B. Network Model

We consider a population of V vehicles, U RSUs, and M MEC servers, such that $\mathcal{V} = \{1, 2, \ldots, V\}$, $\mathcal{U} = \{1, 2, \ldots, U\}$, $\mathcal{M} = \{1, 2, \ldots, M\}$ denote a set of vehicles, RSUs, and MEC servers, respectively. We assume that the quantity of raw environmental data (in bits) is Q. Let $v_n, v \in \mathcal{V}$ and $u_n, u \in \mathcal{U}$ represent vehicle v and RSU u belonging to target cell n, such that v_n comprises a computing power F_v (CPU cycles/bits) and an environmental data Q_v. We distinguish u_n by computing power F_u and an environmental data Q_u. The MEC server $m \in \mathcal{M}$ has computing power F_m for the HD map construction process. Suppose that the target cell can be subdivided into K sub-regions containing the same quantity of environmental data as Q/K.

The image data Q_k gathered and pre-processed using our map data optimization technique in the kth sub-region is expressed as

$$Q_k = \begin{cases} Q_{v,k} = x_{v,k} \cdot \hat{S}_{v,k} \cdot \frac{Q}{K}, & \text{if } v_n \geq 1, \\ Q_{u,k} = x_{u,k} \cdot \hat{S}_{u,k} \cdot \frac{Q}{K}, & \text{if } v_n = 0, \end{cases} \tag{1}$$

where $Q_{v,k}$ and $Q_{u,k}$ denote the data from on v_n and u_n, respectively. $x_{v,k}$ and $x_{u,k}$ determine the vehicles or RSUs nearest to sub-region k for data collection and pre-processing. $\hat{S}_{v,k}$ and $\hat{S}_{u,k}$ represent the sensing variable for vehicle v and RSU u. Time $t_k = \{t_{v,k}, t_{u,k}\}$ taken by vehicle v_n or RSU u_n for map data pre-processing is defined as

$$t_k = \begin{cases} t_{v,k} = \frac{Q_{v,k} \cdot F_{Q_{v,k}}}{F_v}, & \text{if } v_n \geq 1, \\ t_{u,k} = \frac{Q_{u,k} \cdot F_{Q_{u,k}}}{F_u}, & \text{if } v_n = 0, \end{cases} \tag{2}$$

where $t_{v,k}$ and $t_{u,k}$ represent the time taken by v_n and u_n, respectively. $F_{Q_{v,k}}$ and $F_{Q_{u,k}}$ denote the required computing intensity of $Q_{v,k}$ and $Q_{u,k}$, respectively.

The corresponding transmission data Π_{Q_k} can be expressed as

$$\Pi_{Q_k} = \begin{cases} \Pi_{Q_{v,k}} = Q_{v,k} \cdot \left(|\frac{\hat{L}_{v_n}}{\hat{S}}| + 1 \right), & \text{if } v_n \geq 1, \\ \Pi_{Q_{u,k}} = Q_{u,k} \cdot \left(|\frac{\hat{L}_{u_n}}{\hat{S}}| + 1 \right), & \text{if } v_n = 0, \end{cases} \tag{3}$$

where $\Pi_{Q_{v,k}}$ and $\Pi_{Q_{v,k}}$ represent the transmission data from v_n and u_n, respectively. $|\frac{\hat{L}_{v_n}}{\hat{S}}|$ represents the number of relay hops.

Moreover, we express the wireless transmission rate of v_n to RSU u_n separated by the distance d_{v_n} as

$$R_{v_n}^{u_n} = b_{v_n} \cdot \log \left(1 + \frac{P_{v_n} \cdot |h_{v_n}|^2 \cdot (d_{v_n})^{-\varrho_{v_n}}}{N_{v_n}} \right), \tag{4}$$

where b_{v_n}, P_{v_n}, h_{v_n}, ϱ_{v_n}, and N_{v_n} represent vehicle v_n's allocated bandwidth, transmission power, complex channel fading coefficient, path-loss exponent, and noise power, respectively. Likewise, the wireless transmission rate of RSU u_n to MEC server m separated by distance d_{u_n} can be expressed as

$$R_{u_n}^{m} = b_{u_n} \cdot \log \left(1 + \frac{P_{u_n} \cdot |h_{u_n}|^2 \cdot (d_{u_n})^{-\varrho_{u_n}}}{N_{u_n}} \right), \tag{5}$$

where b_{u_n}, P_{u_n}, h_{u_n}, ϱ_{u_n}, and N_{u_n} represent RSU u_n's allocated bandwidth, transmission power, complex channel fading coefficient, path-loss exponent, and noise power, respectively. Additionally, in one update period T, the round trip time (RTT) t_{RTT} taken by either vehicle v_n or RSU u_n can be expressed as

$$t_{RTT} = \begin{cases} t_{Q_{v,k}} + t_{v,k} + t_{v_n}^{u_n} + t_{u_n}^{m}, & \text{if } v_n \geq 1, \\ t_{Q_{u,k}} + t_{u,k} + t_{u_n}^{m}, & \text{if } v_n = 0, \end{cases} \tag{6}$$

where $t_{Q_{v,k}}$, $t_{v_n}^{u_n}$, $t_{u_n}^{m}$, and $t_{Q_{u,k}}$ denote the time taken to collect data on v_n, transmission time of vehicle v_n to RSU u_n, transmission time from RSU u_n to MEC server m, and data collection time of RSU u_n, respectively.

C. Utility Model

Due to bandwidth limitations, it is necessary to explore methods for reducing transmission data volume and the number of transmissions between vehicles and RSUs. Therefore, the goal of this research is to minimize the total amount of transmission data under the HD Map update period T constraints, which can be provided by

$$\min_{Q_k} \quad \Pi_{Q_k} \tag{7a}$$

$$\text{s.t.} \quad t_k < t_{RTT} \leq T, \tag{7b}$$

$$\sum_{v=0}^{V} (x_{v,k} \cdot \hat{S}_{v,k}) \geq 1, \tag{7c}$$

$$F_{Q_{v,k}} \leq F_v, \quad F_{Q_{u,k}} \leq F_u, \tag{7d}$$

$$H(Q_{v,k}) \geq 1, \tag{7e}$$

$$Q_{v,k} \leq Q_{v,k}^*, \quad Q_{u,k} \leq Q_{u,k}^*, \tag{7f}$$

$H(Q_{v,k}) = \mathbb{E}[-\log P(Q_{v,k})]$ is the information entropy (e.g., amount of detected objects) of any transmission data and the objective function seeks to minimize the total amount of data transmissions concerning map data size subjected to the constraints of the HD map update period T. Constraint (7b) stipulates that the entire HD map period T shall not be exceeded by the round-trip time t_{RTT} and map processing duration t_k. Next, sub-region k will always have at least one entity that is able to collecting and processing HD map data according to constraint (7c). The third constraint in (7d) ensures that the amount of resources needed for data collection and processing does not exceed the capabilities of the vehicle or RSU. Constraint (7e) requires at least one detected object in the image data and constraint (7f) states that the optimized data size $\{Q_{v,k}^*, Q_{u,k}^*\}$ does not exceed the original data size $\{Q_{v,k}, Q_{u,k}\}$, respectively.

Due to our environment's complex nature (e.g., real-time constraints, large amounts of data, dynamic network topology, and limited network resources), we propose using CNNs to solve our optimization problem. CNNs can handle complex, non-linear problems, making them a promising approach for optimization problems. CNNs can learn patterns and relationships within the data, allowing for effective optimization. Additionally, CNNs are highly parallelizable, making them efficient for solving large optimization problems. Consequently,

by modifying the CNNs in YOLOX, we model and enforce the constraints from our optimization problem to achieve a near-optimal HD map data optimization.

III. YOLOX-BASED HD MAP DATA OPTIMIZATION

The YOLOX algorithm is the most recent addition to the YOLO family of algorithms, with image inference time as quick as 0.007s. YOLOX can process 140 frames per second, making it adequate for real-time image detection. Due to its small structure, the weight data file of the YOLOX-S version is 16 MB, which is one-tenth the size of the YOLO-V5 [19], making it less resource-intensive and simple to install. The YOLOX is a one-stage target identification technique based on regression that divides the input image into $N \times N$ grids. If the target's center falls within a specified grid, the grid is in charge of detecting the target. Furthermore, the YOLOX employs multi-scale detection technology, which increases the recognition of objects of varied scales.

A. HD Map Data Optimization with YOLOX

The YOLOX technique normalizes all input images to 640×640 pixels. The normalized images are then downsampled using the feature extraction network to three scales of 80×80, 40×40, and 20×20. The resulting feature maps are numerous times smaller than the original image, enabling the detection network's feature description of small objects to be much more easily acquired. YOLO then uses a convolutional neural network to predict several bounding boxes and class probabilities for those boxes to recognize objects in the downsampled images. Like humans, YOLO can instantly determine where and what things are within a given image. As a result, we use YOLOX's efficient object detection technology for our data optimization stage. Furthermore, the proposed HD map data optimization consists of four stages: HD map object detection, detected image gridding, compute percentage utilization, and redundant pixel compression. More particularly, we have

- ***HD map object detection:*** We use the YOLOX algorithm to detect objects in image data recorded by vehicles or RSUs. YOLO divides the image into grids and then predicts the positions, sizes, and confidence scores of a set number of bounding boxes within each grid cell. YOLO, in essence, predicts the class and likely location of an object. If the center of an object falls inside a grid cell, the grid cell's bounding boxes are in charge of precisely finding and estimating that object. Each bounding box will contain five predictions: x-axis, y-axis, width, height, and confidence. The estimated confidence score reflects how certain the model is that a class exists within the bounding box and how well it believes the class fits within the box, using an Intersection over Union metric.
- ***Detected image gridding:*** In this phase, we divide YOLOX output image into dynamic grids (e.g., 100×100, 50×50, and 20×20) to apply our pixel compression technique. Using numerous grids boosts the pixel compression rate but decreases the latency of the proposed compression technique. In this study, we obtained our best results with a grid size of 100×100 with reasonable latency and compression rate.
- ***Compute percentage utilization:*** At this stage, we calculate the percentage area used by our grids to establish the best compression rate for that pixel. For example, suppose a portion of a detected object falls within a grid. In that case, we compute the area it occupies in the grid in comparison to the overall area of the grid to obtain the grid's percentage utilization. Based on this percentage utilization, we can estimate the relative importance of each grid and whether or not to give a compression rate. In Algorithm 1, we demonstrate how to compute the percentage utilization and the computation can also be expressed as

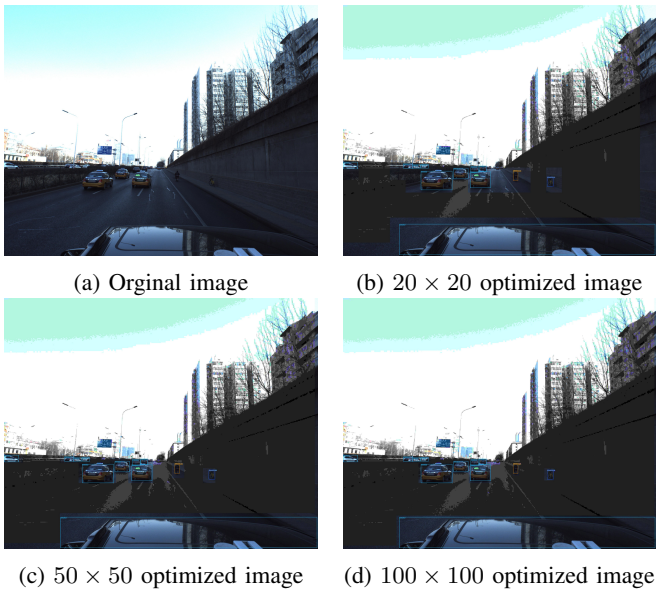
$$g_{used} = g_{area} - \bar{g}_{area}, \tag{8}$$

where g_{used}, g_{area}, and $\bar{g}_{area}$ denote the area used, total grid area, and detected object area, respectively.
- ***Redundant pixel compression:*** After calculating the percentage utilization of each grid, we perform pixel compression on it. For example, a grid (e.g., skies, clouds, skyscrapers, roadblocks, etc.) with no detected objects will have a zero percentage utilization, necessitating greater pixel compression. Grids with identified objects have a greater percentage utilization, such as 100%,

Algorithm 1: HD map data optimization with YOLOX

input : Q: raw image data
output: Q^*: compressed image data

```
/* finds object in any grid                    */
void compute_object_location(r_box, b_box):
    if b_box ≥ r_box and b_box ≤ r_box then
        {dy,dx} = b_box - r_box ;
    else if b_box ≥ r_box then
        {dy,dx} = r_box - b_box ;
    else if b_box ≥ r_box and b_box ≤ r_box then
        {dy,dx} = b_box - b_box ;
    else if b_box < r_box and b_box > r_box then
        {dy,dx} = b_box - b_box ;
    return dx × dy;
return
/* computes percentage utilization  */
void compute_p_area(r_box, b_boxes):
    r_box_area = (r_box.x1 - r_box.x0) × (r_box.y1 -
     r_box.y0);
    box_areas = 0;
    for box in b_boxes do
        b_areas += compute_object_location(r_box,
         box);
        return (b_areas/r_box_area) × 100;
    end
return
```



(a) Orginal image (b) 20 × 20 optimized image

(c) 50 × 50 optimized image (d) 100 × 100 optimized image

Fig. 2: HD image data optimization with our modified YOLOX.

and so do not get any compression. In particular, if $g_{used} < g_{max}$ we perform corresponding compression, where g_{max} is a defined set of thresholds. Additionally, we present Algorithm 1 to demonstrate how to compress the pixels in each grid based on their percentage utilization.

Fig. 3 illustrates the proposed image data optimization with an original and optimized image with varying grid sizes (20×20, 50×50, and 100×100). We can observe the detected object within the bounding boxes, and the pixel compression is done around the object detection in Figs. 2b, 2c, and 2d.

B. Complexity Analysis of Proposed Scheme

YOLOv3-5 uses an anchor-based mechanism for object detection, where fixed reference bounding boxes are placed throughout the image to check if they contain an expected class. This approach can find more than one object in the same grid. However, it also has several drawbacks. Optimal anchor boxes need to be determined through clustering analysis before training, adding extra time and complexity. The number of predictions per image is increased, leading to longer inference times. Finally, the anchor-based mechanism increases the complexity of the detection head and overall network. YOLOX, the latest version of YOLO, uses an anchor-free mechanism, reducing the number of predictions per image cell from 3 to 1 and simplifying the overall network. Suppose the time complexity of the YOLOX algorithm is given by O(Y). Our modification offers an asymptotic time complexity of O(M×N), where M and N represent the number of rows and columns of grid shapes. As a result, the time complexity can be expressed as O(M×N)+O(Y).

Parameters	Value
Number of vehicles	$10 - 50$
Number of RSU	20
Number of routers	10
MEC servers	5
Communication range	$200 - 300$m
P2P link delay/ bandwidth	10ms/200Mbps
CARLA version	0.9.12
Environment setting	Town 1
Traffic	50 pedestrians/30 cyclists
Road speed	$30 - 60$m/s
Transportation protocol	IEEE 802.11ac
Map data size	$50 - 400$MB

TABLE I: Experiment Parameter List

IV. EXPERIMENT RESULTS AND ANALYSIS

A. System Configuration

We conduct the research using the Python 3.8 environment on a Core i7 CPU machine with a 3.9 GHz clock speed and 64GB of RAM. Using the CARLA simulation environment, we deploy the proposed latency optimization technique for HD map updates in vehicular settings. CARLA facilitates the creation, training, and validation of autonomous driving systems and provides open digital assets (urban layouts, buildings, and vehicles), open-source protocols, and technologies. The simulation platform provides dynamic sensor packages, ambient conditions, comprehensive control of static and dynamic actors, and more. We employ a multilane urban traffic flow vehicle trajectory, including vehicles, pedestrians, crossings, cross traffic, traffic laws, and other complexities that differentiate urban driving from track racing. In addition, we connected ROS2 with CARLA (ROSbridge) to simulate computations on multiple vehicles and RSUs, such as router and RSU connections, HD map data collection, optimization, and construction. Using ROS2, vehicles in our CARLA environment can also execute complicated algorithms, such as our improved YOLOX HD map data optimization. Multiple Nvidea Jetson AGX models with a high processing capability are utilized on the MEC server for HD map creation and dissemination. Also, we assume the P2P link latency between the router, vehicles, RSU, and MEC server to be 10 milliseconds with a bandwidth of 200 Mb/s. Then, using ROSbridge, we transmit the vehicle data from CARLA, including the number of vehicles, driving direction, speed, and road network. To boost the efficiency of HD map transmission, we employ the IEEE 802.11ac protocol to send the map data. Table I presents the parameter values in detail.

B. Performance Metrics and Experiment Benchmarks

1) Baseline schemes (Benchmarks): To show the performance of our proposed latency optimization scheme, we implement the three equivalent baseline schemes listed below:

1) *RLSS Technique:* This approach employs a deep reinforcement learning-based architecture to train a neural network as an agent to decide on data source selection to improve HD map update action performance concerning latency, throughput, and packet loss.

2) *Loss-based Approach:* This technique forces the vehicle to switch data sources during an HD map update if packet loss occurs in the current data source. The authors seek to achieve high connection usage by filling the buffer to capacity, which may result in severe packet loss during handovers.
3) *Pro-RTT:* In this system, the vehicle employs the probability-based handover approach to choose a new data source by monitoring the RTT, which decreases the frequency of handovers.

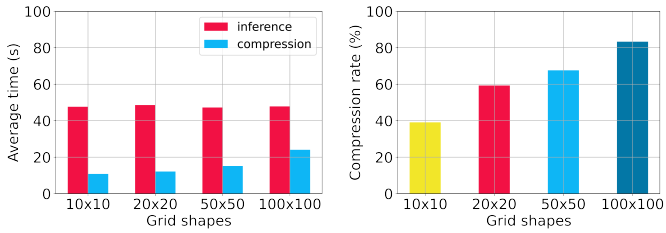
2) Metrics of Performance: To evaluate the performance of our proposed map optimization scheme, we employ the following performance metrics:

1) *Compression Rate:* Compression rate, also termed compression power, measures how much a data compression technique reduces the size of collected data. This metric applies to our modified YOLOX-based map data optimization technique.
2) *Detection Error Rate:* This is the percentage of incorrectly detected objects in a set of our HD map images compared to the total number of actual objects detected by our various grid shapes in those images.
3) *Computation Time:* Computation time is the amount of time necessary to complete a computational operation. In this measure performance study, we take into account the map data optimization performed on vehicles or the RSU.
4) *Transmission Time and Data Size:* This is the overall amount of time spent on the HD map updating process, including data collection, data optimization, data transfer, and HD map construction. The transmission data size represents the average size (Mb) of map data sent to the RSUs for HD map construction.

C. Implementation Discussions

This experiment discussion investigates how different pixel compression grid sizes affect computation time and compression rate. Figs. 3a and 3b depict the compression time and rate of our suggested picture data optimization strategy. We first discuss inference and compression time to understand how computing time is computed. The inference time is the time it takes YOLO to recognize an item, and the compression time is the time it takes to compress the redundant picture pixels. Fig. 3a shows that the inference time is constant at 0.39s in all grid sizes, although the compression time varies. The 10×10 grid has the shortest compression time, followed by the 20×20, 50×50, and 100×100 grids. The number of pixels to compress in each grid size is responsible for this pattern in the findings. More grids necessitate more processing, affecting our suggested approach's latency.

In addition, as shown in Fig. 3b, the 100×100 grid obtains the maximum compression rate, followed by 50×50, 20×20, and 10×10. The 100×100 grid achieves 24%, 37%, and 98.55% more compression rates compared to 10×10, 20×20, and 50×50, respectively. This result is due to the number of pixels compressed in each grid size, as larger grids require more compression, reducing the image data size to optimize transmission data. From Figs. 3a and 3b, we select the 100×100 grid size because of its relatively low computation time and high compression rate.



(a) Computational time (b) Compression rate

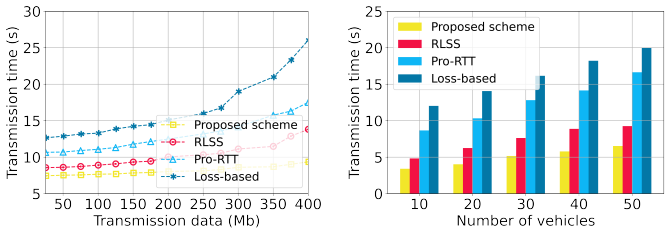
Fig. 3: Impact of map data optimization.

Additionally, the detection error rate decreases as the grid shapes increase. This is because a finer grid can result in more precise bounding box predictions, while a coarser grid can lead to less precise predictions but faster computation time. Using 100×100 grid shapes, we achieve a lower object detection error rate but a higher computational time than 10×10, 20×20, and 50×50. On average, 100×100 produces 75% more accuracy but 22% more computational time than the other baseline grid shapes.

In this part of the experiment, we investigate the influence of HD map data size and the number of vehicles on transmission time using various baseline techniques. Figs. 4a and 4b depict the transmission time in different baseline systems with varied data volumes and number of vehicles. The following are our observations: 1) The proposed scheme outperforms other baseline schemes considerably under different data sizes, and 2) when data size and the number of vehicles increase, the transmission time of the baseline schemes increases drastically compared to the proposed schemes. Our proposed scheme maintains a consistent rise compared to other baseline systems due to image data optimization. In NDN vehicular settings, data size substantially impacts transmission time performance. For example, when the data increases to 400Mb, the media transmission time increases to 9s, 16s, 20s, and 30s in the proposed scheme, RLSS, Pro-RTT, and Loss-based schemes, respectively. Regarding the number of vehicles, the average transmission time for the proposed scheme, RLSS, Pro-RTT, and Loss-based schemes is 4.98s, 9.37s, 14.72s, and 19.8s, respectively.

V. Conclusion

In this study, we designed and implemented a data optimization scheme for HD map updates in vehicular NDN scenarios. We first formulated our HD map optimization problem as a convex optimization problem and solved it with CNNs from the YOLOX algorithm. In particular, we modified the YOLOX object detection algorithm to detect and compress redundant pixels in local map data before transmission to the MEC server. To deploy our proposed scheme, we constructed a vehicular NDN environment for data collection, processing,



(a) Transmission data size (Mb/s) (b) Transmission time (s)

Fig. 4: Transmission data size and time.

and transmission using the CARLA simulator and ROS2. We show that the proposed HD map data optimization algorithm can guarantee improved performance by significantly reducing transmission data and time. Compared to the baseline schemes discussed, our proposed approach cuts the amount of data transmitted and the transmission time by $48.25\% - 65.78\%$ and $46.85\% - 78.84\%$.

REFERENCES

[1] L. Liu, Y. Zhou, V. Garcia, L. Tian, and J. Shi, "Load aware joint comp clustering and inter-cell resource scheduling in heterogeneous ultra dense cellular networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 3, pp. 2741–2755, Nov. 2017.

[2] C. Xing, S. Ma, and Y. Zhou, "Matrix-monotonic optimization for mimo systems," *IEEE Transactions on Signal Processing*, vol. 63, no. 2, pp. 334–348, Nov. 2014.

[3] M. Amadeo, C. Campolo, and A. Molinaro, "Information-centric networking for connected vehicles: a survey and future perspectives," *IEEE Communications Magazine*, vol. 54, no. 2, pp. 98–104, Feb. 2016.

[4] Y. Zhao and Q. Zhu, "Evaluation on crowdsourcing research: Current status and future direction," *Information Systems Frontiers*, vol. 16, no. 3, pp. 417–434, Jul. 2014.

[5] B. Camburn, R. Arlitt, D. Anderson, R. Sanaei, S. Raviselam, D. Jensen, and K. L. Wood, "Computer-aided mind map generation via crowdsourcing and machine learning," *Research in Engineering Design*, vol. 31, no. 4, pp. 383–409, Oct. 2020.

[6] K. Kim, S. Cho, and W. Chung, "Hd map update for autonomous driving with crowdsourced data," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1895–1901, Feb. 2021.

[7] S. H. Ahmed, S. H. Bouk, M. A. Yaqub, D. Kim, and H. Song, "Difs: Distributed interest forwarder selection in vehicular named data networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 9, pp. 3076–3080, Nov. 2017.

[8] J. Wang, C. Jiang, K. Zhang, T. Q. Quek, Y. Ren, and L. Hanzo, "Vehicular sensing networks in a smart city: Principles, technologies and applications," *IEEE Wireless Communications*, vol. 25, no. 1, pp. 122–132, Oct. 2017.

[9] Y. Zhou, L. Tian, L. Liu, and Y. Qi, "Fog computing enabled future mobile communication networks: A convergence of communication and computing," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 20–27, Mar. 2019.

[10] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 3860–3873, Feb. 2016.

[11] P. Li, S. Guo, T. Miyazaki, X. Liao, H. Jin, A. Y. Zomaya, and K. Wang, "Traffic-aware geo-distributed big data analytics with predictable job completion time," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 6, pp. 1785–1796, Nov. 2016.

[12] W. Wang, K. Zhu, L. Ying, J. Tan, and L. Zhang, "Maptask scheduling in mapreduce with data locality: Throughput and heavy-traffic optimality," *IEEE/ACM Transactions On Networking*, vol. 24, no. 1, pp. 190–203, Nov. 2014.

[13] A.-C. G. Anadiotis, G. Morabito, and S. Palazzo, "An sdn-assisted framework for optimal deployment of mapreduce functions in wsns," *IEEE Transactions on Mobile Computing*, vol. 15, no. 9, pp. 2165–2178, Nov. 2015.

[14] Z. Jiang, C. Xu, J. Guan, H. Zhang, and S. Yu, "Loss-aware adaptive scalable transmission in wireless high-speed railway networks," in *2017 IEEE International Conference on Communications (ICC)*, 2017, pp. 1–6.

[15] J. Leng, G. Ruan, Y. Song, Q. Liu, Y. Fu, K. Ding, and X. Chen, "A loosely-coupled deep reinforcement learning approach for order acceptance decision of mass-individualized printed circuit board manufacturing in industry 4.0," *Journal of cleaner production*, vol. 280, p. 124405, 2021.

[16] X. Guo, Y. Cao, J. Zhou, Y. Huang, and B. Li, "Hdm-rrt: A fast hd-map-guided motion planning algorithm for autonomous driving in the campus environment," *Remote Sensing*, vol. 15, no. 2, p. 487, 2023.

[17] F. Wu, W. Yang, J. Lu, F. Lyu, J. Ren, and Y. Zhang, "Rlss: A reinforcement learning scheme for hd map data source selection in vehicular ndn," *IEEE Internet of Things Journal*, Nov. 2021.

[18] S. Arshad, M. Sualeh, D. Kim, D. V. Nam, and G.-W. Kim, "Clothoid: an integrated hierarchical framework for autonomous driving in a dynamic urban environment," *Sensors*, vol. 20, no. 18, p. 5053, Sep. 2020.

[19] G. Lu and Y. Wang, "The improved yolo-v5 based automatic non-parking overloading detection method," in *2022 5th International Symposium on Autonomous Systems (ISAS)*, vol. 2, no. 1, Apr. 2022.