SAFE: Intelligent Online Scheduling for Collaborative DNN Inference in Vehicular Network

Ruiting Zhou*†, Ziyi Han*, Yifan Zeng*, Zhi Zhou‡, Libing Wu*, Wei Wang§
*School of Cyber Science and Engineering, Wuhan University, China
†School of Computer Science and Engineering, Southeast University, China
‡School of Computer Science and Engineering, Sun Yat-sen University, China
§Department of Computer Science and Engineering, Hong Kong University of Science and Technology, China
Email: ruitingzhou@seu.edu.cn, {ziyihan, yifanzeng}@whu.edu.cn, zhouzhi9@mail.sysu.edu.cn, wu@whu.edu.cn, weiwa@cse.ust.hk

***Abstract*—Recent years have witnessed a widespread use of deep neural networks (DNNs) in providing various intelligent services, and vehicular networks are no exception. Given the limited computing capabilities of vehicles, collaborative vehicle-edge DNN inference has emerged as a viable alternative. This approach employs DNN partitioning, where a part of DNN is computed on vehicles, and the other part on the edge, *e.g.,* roadside unit (RSU), aiming to enhance the inference accuracy and reduce the inference latency. In this setting, deriving an optimal DNN partitioning scheme becomes critical, yet challenging given the constant movement of vehicles and the highly dynamic wireless connections. Furthermore, vehicles may move out of the signal coverage of an RSU, making it difficult to receive the inference results. To this end, we propose a two-stage intelligent scheduling framework named Soft Actor-critic for discrete actions (SAC-D) based collaborative DNN inference FramEwork (`SAFE`). *SAFE* engages multiple RSUs to assist vehicles in completing inference tasks sequentially and ensuring reliable data transmission. It can learn the dynamic vehicular network and make scheduling decisions to minimize the overall latency of vehicle inference tasks. Extensive experimental results show that *SAFE* can reduce up to 80% of the overall latency with a lower failure rate, compared to four baselines.**

I. INTRODUCTION

The recent technological advancement of deep neural networks (DNNs) has enabled an increasing number of intelligent services in vehicular networks. However, DNN inference tasks are usually compute-intensive [1] and latency-sensitive [2], making it infeasible to deploy complex models and independently execute them on vehicles with only limited onboard computing resources. In contrast, roadside units (RSUs) equipped with edge servers possess stronger computational and storage capabilities. This opens up a potential opportunity for collaborative vehicle-edge inference, in which a DNN model is partitioned into two parts, one computed on the vehicle and the other on a nearby RSU. The vehicle computes the first part and transfers the intermediate output to the RSU, which computes the remaining part and returns the result back to the vehicle [3]. Collaborative vehicle-edge inference not only reduces the inference latency but also effectively combines the dynamic information from vehicles with the static information from RSUs, resulting in significantly better inference results. For example, vehicle-mounted cameras and sensors can capture data, and collaborative vehicle-edge inference can be employed to reduce latency and enhance the accuracy of video analysis and object detection/tracking, thereby ensuring safe driving [4].

However, collaborative DNN inference in vehicular networks faces several challenges. **First**, DNNs have chain or directed acyclic graph (DAG) topologies, where each layer's operation depends on the previous layers' outputs [5]. The computing demand and the size of the intermediate data generated at different layers in DNN models vary dramatically. For instance, the intermediate data given by the conv1 layer in YOLOv2 is about 5 MB, while the data generated by the middle layer max5 is only 0.08 MB [6]. As a result, partitioning at different layers results in diverse computation and transmission times, leading to different inference latencies [7]. Optimally partitioning DNNs is hence essential for efficient computation and data transfer during collaborative inference. **Second**, as the environments of vehicular networks change dynamically, such as uneven workload distribution and unstable wireless connections [8], the allocation of computing resources and wireless bandwidth to vehicles should be adjusted accordingly, so does the DNN partitioning scheme [9]. Therefore, optimal DNN partitioning must account for the dynamic vehicular network environment. **Third**, vehicles are constantly moving, while some inference tasks require a few seconds to finish computing, *e.g.,* trajectory prediction with a large-scale model and data [10]. The vehicle may move out of an RSU's signal coverage before the inference completes, making it unable to receive the inference results.

Existing work in achieving efficient DNN inference with

Corresponding author: Ziyi Han.
This work is supported in part by the NSFC Grants (62072344, U20A20177and 62232004).

edge collaboration includes methods like model compression [11], [12] and model partition [13], [14]. These methods focus on optimizing model structures and DNN partitioning in an end-edge-cloud environment, without considering the mobility of vehicles, which may lead to task failures. Research on vehicular networks, on the other hand, mainly addresses task offloading and resource allocation [15], [16], but these approaches primarily target general computing tasks and do not exploit the unique characteristics of DNN inference tasks. To our knowledge, there are only a few studies on DNN inference in vehicular networks. Notably, Wang *et al.* [9] proposed an algorithm to select one RSU for collaborative inference, but did not address the potential failures of data transmission caused by vehicle mobility (details in Sec. V).

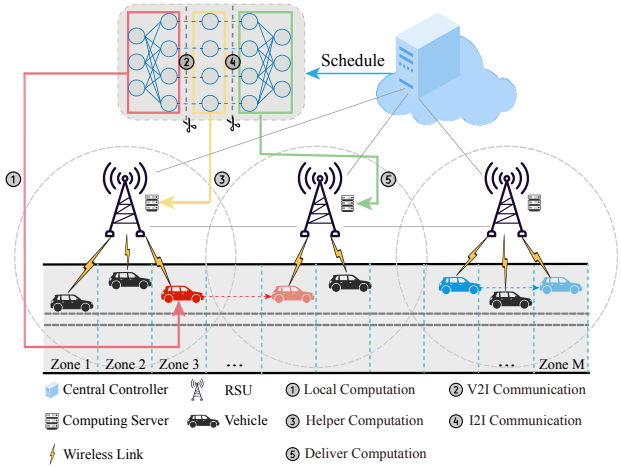

Fig. 1: Collaborative DNN inference in vehicular networks.

In this paper, we design a Soft Actor-critic for discrete actions (SAC-D) based collaborative DNN inference FramEwork (SAFE) to achieve reliable and low-latency DNN inference in vehicular networks. To our knowledge, this is the **first formal study** of online scheduling design for collaborative DNN inference across multiple RSUs in vehicular networks. To better illustrate this idea, we refer to Fig. 1. For the red vehicle, the first RSU is selected to execute part of DNN inference as the helper and the second RSU is used as the deliverer for the remaining part of computation and result delivery; in contrast, for the blue car, only one RSU is selected. Note that SAFE can be easily extended to scenarios where multiple RSUs are used for collaborative inference with more complex road conditions (detailed discussions are given in Sec. III-E).

We propose to use deep reinforcement learning (DRL) to learn the implicit relation between coupled scheduling decisions, and make fast decisions for latency-sensitive inference tasks under the typical traffic pattern in vehicular networks (*e.g.,* higher vehicle volume and slower speeds during peak hours). However, due to the large decision space, it usually takes a long time for the learning process to converge. To reduce the exploration space and expedite the learning process, we propose a two-stage approach. In the first stage, SAFE employs SAC-D, an efficient DRL algorithm designed for learning implicit relation between two discrete decisions (*i.e.,* RSU selection and DNN partitioning). This stage enables effective adaptation to dynamic vehicular networks, vehicle mobility, and the characteristics of DNN models. The resource allocation of one RSU is relatively static and depends on the first stage decision. So the optimization method can be used to find the solution quickly without adding the scheduling overhead. We summarize our main contributions as follows:

- *Collaborative DNN Inference Model.* We analyze the characteristics of DNN models and the mobility of vehicles, and explicitly express the collaborative inference latency in vehicular networks. Based on that, an average latency minimization problem is formulated to evaluate the performance of SAFE.
- *Two-stage Collaborative Inference Framework.* To solve the formulated problem, SAFE first decomposes the problem into two subproblems, workload distribution and resource allocation. For the first subproblem, SAFE utilizes SAC-D, which combines four crucial elements to enhance performance: a maximum entropy framework for exploration and stability, an actor-critic architecture for efficient and faster training, an off-policy approach with experience replay to improve convergence efficiency, and discrete actions for determining RSU selection and DNN partitioning. In the second stage, SAFE further decomposes the resource allocation subproblem into a sequence of single-slot computing resource and wireless bandwidth allocation problems. By leveraging Karush–Kuhn–Tucker (KKT) conditions, which are usually applied to solve convex optimization problems, SAFE makes the optimal resource allocation decisions for each RSU in linear time.
- We evaluate the effectiveness of SAFE through extensive experiments. The results show that i) SAFE achieves the low average latency and the high success rate; ii) SAFE significantly outperforms four baselines in various vehicular network scenarios, with different resource capacity, input scale, or DNN model types; iii) SAFE improves the success rate by up to 65% while reducing the average latency by up to 80%, compared to four baselines.

II. System Model

A. DNN Inference in Vehicular Networks

System Overview. As shown in Fig. 1, we consider a vehicular network along a road with a number of RSUs, denoted as $\mathcal{S}$. Each RSU $s \in \mathcal{S}$ is equipped with a computing server and has a signal coverage of radius r. Denote the computing resource and the wireless bandwidth capacity of RSU s as C_s and B_s, respectively. Vehicles traversing the road can communicate with a nearby RSU and execute DNN inference tasks in cooperation with RSUs. When vehicles submit their tasks to the nearby RSU, the central controller connected to all RSUs can collect these tasks as well as the information of vehicles to make decisions. Let $\mathcal{X}$ denote the integer set $\{1, 2, \ldots, X\}$.

Vehicle Information. On the road, I vehicles travel at constant speeds. To model the mobility of vehicles, we adopt

zone-based and time-slotted models. The road is divided into M zones with equal lengths. Let $L_s^m \in \{0,1\}$ denote whether zone m is within the signal coverage of RSU s. We assume that the zone location of vehicles remains unchanged during a time slot, and updates at the beginning of the next time slot. Over a time span $\mathcal{T}$, vehicles generate DNN inference tasks randomly and request processing. We assume that there will not be two uncompleted tasks on a vehicle at the same time. Therefore, each vehicle $i \in \mathcal{I}$ can be represented by a tuple $\{v_i, c_i, a_i, l_i(t)\}$, where v_i denotes the travel speed of vehicle i, c_i represents the computing capacity of vehicle i, $a_i \in \mathcal{T}$ indicates the request time for the task of vehicle i, and $l_i(t) \in \mathcal{M}$ denotes the location of vehicle i at time slot t.

Collaborative Inference Model. To overcome the significant latency incurred by resource-limited vehicles in completing inference tasks independently, we adopt a collaborative model that leverages the resources of RSUs to accelerate the inference process. Considering the mobility of vehicles, a vehicle may leave the signal coverage of the cooperative RSU when the RSU completes the task, such as the red vehicle in Fig. 1. To address this, we extend the collaborative model by selecting two RSUs, referred to as the helper and the deliver [17]. Both the helper and the deliver can assist the vehicle in completing the inference task sequentially, with the deliver responsible for sending the results to the vehicle. Note that one RSU can be selected as the helper and the deliver simultaneously. It means that the vehicle is within the signal coverage of this RSU for the entire duration of task processing and is assisted only by this RSU, such as the blue vehicle in Fig. 1.

DNN Model Partition. DNN models are well-trained and pre-installed on both vehicles and RSUs. Due to limited computing and memory resources in vehicles, the deployed models on vehicles are compressed, reducing the computing workload while preserving models' topology and intermediate data size [18]. Conversely, RSUs deploy complete and uncompressed models. Similar to [19], the architecture of a DNN model is structured as a sequence of logical layers. The DNN model of vehicle i consists of K_i logical layers. The output of k-th layer is called the intermediate data of k-th layer, which is required as the input by the $(k+1)$-th layer. Each layer $k \in \mathcal{K}_i$ can be represented by a tuple $\{x_{i,k}, x'_{i,k}, d_{i,k}\}$. $x_{i,k}/x'_{i,k}$ indicates the complete/compressed computing workload of k-th layer, which is determined by the layer type, input size, and output size. $d_{i,k}$ denotes the intermediate data of the k-th layer. Each DNN model follows a fixed structure. Once the specific DNN model type employed by vehicle i is known, the values of K_i and $\{x_{i,k}, x'_{i,k}, d_{i,k}\}_{\forall k \in \mathcal{K}_i}$ can be obtained. Based on the settings of helper and deliver, it is necessary to select two partition layers to divide the model into three parts. The first part of the DNN model is executed on the vehicle, and the remaining two parts are executed sequentially on the helper and the deliver.

Decision Variables. After the DNN inference task of vehicle i arrives at time slot a_i, the decisions made by the central controller include: i) $h_{i,s}, d_{i,s} \in \{0,1\}$, binary variables which represent whether RSU s is selected as the helper/deliver for vehicle i; ii) $p_{i,k_1}, e_{i,k_2} \in \{0,1\}$, binary variables which indicate whether layers k_1 and k_2 are selected as two partition layers for vehicle i; iii) $\alpha_{i,s}^t \in [0,1]$, the ratio of computing resources allocated to vehicle i in RSU s at t; iv) $\beta_{i,s}^t \in [0,1]$, the ratio of wireless bandwidth allocated to vehicle i in RSU s at t. For convenience, let $\kappa_{i,1}$, $\kappa_{i,2}$ denote the partition layers of vehicle i, *i.e.*, $\kappa_{i,1} = \sum_{k \in \mathcal{K}_i} k \cdot p_{i,k}$ and $\kappa_{i,2} = \sum_{k \in \mathcal{K}_i} k \cdot e_{i,k}$. Important notations are listed in TABLE I for easy reference.

TABLE I: List of Notations

I	# of vehicles	S	# of RSUs
M	# of zones	T	# of time slots
$l_i(t)$	vehicle i's location at t	v_i	vehicle i's speed
a_i	vehicle i's request time	μ_i	vehicle i's total latency
c_i	computing resource capacity of vehicle i		
C_s	computing resource capacity of RSU s		
B_s	wireless bandwidth capacity of RSU s		
K_i	# of layers of DNN model for vehicle i		
L_m^s	whether zone m is within the coverage of RSU s		
$x_{i,k}/x'_{i,k}$	complete/compressed computation workload of the k-th layer of vehicle i		
$d_{i,k}$	intermediate data size of the k-th layer of vehicle i		
$r_{i,s}^t$	data rate between vehicle i and RSU s at t		
$R_{h,d}$	data rate between RSU h and RSU d		
$h_{i,s}/d_{i,s}$	whether RSU s is selected as helper/deliver for vehicle i		
$p_{i,k_1}/e_{i,k_2}$	whether layers k_1 and k_2 are selected as two partition layers for vehicle i		
$\alpha_{i,s}^t$	ratio of resources allocated to vehicle i in RSU s at t		
$\beta_{i,s}^t$	ratio of bandwidth allocated to vehicle i in RSU s at t		

Discussion. Here, we outline four unique situations. i) $\sum_{s \in \mathcal{S}} s \cdot h_{i,s} = \sum_{s \in \mathcal{S}} s \cdot d_{i,s}$, *i.e.*, RSU s acts as both the helper and the deliver for vehicle i, which is mentioned earlier. ii) $\kappa_{i,1} = \kappa_{i,2} = \mathcal{K}_i$, *i.e.*, all computations of DNN inference are completed on the vehicle i without any assistance from RSUs. iii) $\kappa_{i,1} = \kappa_{i,2} < \mathcal{K}_i$, *i.e.*, the helper directly transmits the intermediate data to the deliver after receiving it, and is not responsible for any computation. This usually happens when the workload of the helper is extremely heavy. iv) $\kappa_{i,1} < \kappa_{i,2} = \mathcal{K}_i$, *i.e.*, the deliver only perform result delivery and does not do any computation.

B. Latency Model

Inference Latency. The total latency of the processing task for vehicle i consists of three types of computation latency and two types of data transmission latency, which is calculated as:

$$\mu_i = \mu_i^l + \mu_i^o + \mu_i^h + \mu_i^g + \mu_i^d, \tag{1}$$

where μ_i^l, μ_i^h and μ_i^d are the computation latency for executing the first, second and third part of vehicle i's DNN model at local, helper and deliver, respectively; μ_i^o is the communication latency for transferring intermediate data of the first partition layer from vehicle i to helper to continue execution; and μ_i^g is the communication latency for transmitting intermediate data of the second partition layer of vehicle i from helper to deliver.

Computation Latency. i) *Local Computation:* The local computation latency is the execution time of the first part of the DNN model (from layer 1 to the first partition layer $\kappa_{i,1}$), which can be calculated as $\mu_i^l = \sum_{k:k<=\kappa_{i,1}} x'_{i,k}/c_i$. ii) *Helper Computation:* In the helper computation phase, the latency of vehicle i consists of the execution time of layer $\kappa_{i,1}$ to layer $\kappa_{i,2}$, which can be represented by

$\mu_i^h = \sum_{s\in\mathcal{S}} \sum_{k:\kappa_{i,1}<k<=\kappa_{i,2}} \frac{x_{i,k}}{\alpha_{i,s}^t h_{i,s} C_s}$. iii) *Deliver Computation:* The latency of vehicle i in the deliver computation phase is the execution time of the remaining layers, which can be given by $\mu_i^d = \sum_{s\in\mathcal{S}} \sum_{k:k>\kappa_{i,2}} \frac{x_{i,k}}{\alpha_{i,s}^t d_{i,s} C_s}$.

Communication Latency. i) *Vehicle-to-infrastructure (V2I) communication:* Vehicles communicate with RSUs via a wireless network connection (*e.g.,* 4G, 5G, and wifi). To ensure reliable and high-performance communication, network slicing technology is applied, allowing the allocation of bandwidth resources based on individual requirements of vehicles or services [20]. In this paper, network slicing is implemented using the orthogonal frequency division multiple access technique, which enables simultaneous transmission of multiple signals without causing interference [21]. Then, the data rate between vehicle i and RSU s at time slot t is obtained by the Shannon formula as $r_{i,s}^t = \beta_{i,s}^t B_s \log_2(1 + \frac{\rho_i g_{i,s}^t}{\sigma^2})$ [22], where ρ_i indicates the transmission power of vehicle i, $g_{i,s}^t$ represents the channel gain between vehicle i and RSU s at time slot t, and σ^2 denotes the power of the Gaussian noise in the V2I communication channel. The wireless transmission latency between vehicle i and the helper can be calculated by $\mu_i^o = \sum_{s\in\mathcal{S}} \frac{d_{i,\kappa_{i,1}}}{h_{i,s} r_{i,s}^t}$. ii) *Infrastructure-to-infrastructure (I2I) communication:* RSUs communicate with each other through wired links (*e.g.,* optical fiber). Because of the stability of wired links, we assume that the transmission rate remains constant. Let $R_{s,s'}$ denote the data rate between RSU s and RSU s'. The communication latency between the helper and the deliver is $\mu_i^g = \sum_{s\in\mathcal{S}} \sum_{s'\in\mathcal{S}} \frac{d_{i,\kappa_{i,2}}}{h_{i,s} d_{i,s'} R_{s,s'}}$. It is worth noting that the transmission latency for result delivery is ignored due to the relatively small size of the result [23].

C. Problem Formulation

Problem Formulation. Our objective is to minimize the DNN inference latency while ensuring task delivery, subject to the mobility of vehicles and the limited communication range of RSUs. The online problem for DNN inference in vehicular networks can be formulated as follows.

$$\text{minimize} \quad \frac{1}{I}\sum_{i\in\mathcal{I}} \mu_i \tag{2}$$

$$\sum_{i\in\mathcal{I}} \alpha_{i,s}^t <= 1, \forall s \in \mathcal{S}, \forall t \in \mathcal{T}, \tag{2a}$$

$$\sum_{i\in\mathcal{I}} \beta_{i,s}^t <= 1, \forall s \in \mathcal{S}, \forall t \in \mathcal{T}, \tag{2b}$$

$$\sum_{\forall s\in\mathcal{S}} h_{i,s} L_{l_i(a_i+\mu_i^l)}^s = 1, \forall i \in \mathcal{I}, \tag{2c}$$

$$\sum_{\forall s\in\mathcal{S}} d_{i,s} L_{l_i(a_i+\mu_i)}^s = 1, \forall i \in \mathcal{I}, \tag{2d}$$

$$\sum_{\forall s\in\mathcal{S}} h_{i,s} = 1, \sum_{\forall s\in\mathcal{S}} d_{i,s} = 1, \forall i \in \mathcal{I}, \tag{2e}$$

$$\sum_{\forall k\in\mathcal{K}_i} p_{i,k} = 1, \sum_{\forall k\in\mathcal{K}_i} e_{i,k} = 1, \forall i \in \mathcal{I}, \tag{2f}$$

$$t <= t' < t'', \forall t : \beta_{i,h_i}^t > 0, \forall t' : \alpha_{i,h_i}^{t'} > 0,$$
$$\forall t'' : \alpha_{i,d_i}^{t''} > 0, \forall i \in \mathcal{I}, \tag{2g}$$

$$h_{i,s}, d_{i,s}, p_{i,k}, e_{i,k} \in \{0,1\}, \forall i, \forall s, \forall k. \tag{2h}$$

Constraint (2a) guarantees the allocated computing resources within the capacity of each RSU. The bandwidth capacity of RSUs for data transmission is formulated by constraint (2b). To ensure reliable communication, constraints (2c) and (2d) ensure that vehicles are within the signal coverage of the corresponding RSUs for data transmission and result delivery, respectively. Constraint (2e) means that only one RSU is selected as the helper/deliver for each vehicle. Constraint (2f) represents that only one layer is selected for each partition point. Constraint (2g) enforces the execution sequence of vehicles' inference phases.

Challenge. The above problem (2) is a mix-integer nonlinear optimization problem. Integer linear programming (ILP), which is known as NP-hard [24], is reducible to it. So the problem (2) is also NP-hard even in the offline setting, which is challenging to be solved by conventional optimization methods. Moreover, the network condition dynamic changes and the arrival of vehicles are unknown. Finally, problem (2) is time coupled, making it more difficult to be addressed.

III. The Design of SAFE

A. Main Idea

In order to solve the average latency minimization problem (2), we first decouple the problem into two subproblems: workload distribution and resource allocation. Then, a two-stage SAC-D based framework, SAFE, is proposed to solve them jointly. SAFE includes the following steps:

i. We first transform the workload distribution subproblem into an MDP, and then present a SAC-D based algorithm to solve the MDP. The central controller acts as an agent, which observes the state information s_t from the vehicular network environment. The agent then makes discrete action a_t to decide RSUs selection and DNN partition with s_t based on its policy.

ii. After determining workload distribution, the resource allocation subproblem can be decomposed into a series of independent single-slot resource allocation problems. For each single-slot problem, we derive an optimal solution based on convex optimization theory and KKT condition.

iii. After vehicles perform DNN inference tasks, the environment updates the state to s_{t+1} and feeds back the corresponding reward r_t to the agent. Then the tuple $\{s_t, a_t, r_t, s_{t+1}\}$ is stored into the replay buffer, which is sampled by the agent for its actor and critic networks update. The agent continues to make action with state s_{t+1} until the training of the DRL network is completed. The trained network can then be used for online workload distribution decisions.

B. Solution for Workload Distribution

Problem Transformation. We first reformulate the workload distribution subproblem into an MDP. An MDP can be denoted by the tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma\}$, where $\mathcal{S}$ indicates the state space, $\mathcal{A}$ denotes the action space, $\mathcal{P}$ represents the state transition probability, $\mathcal{R}$ denotes the reward, and $\gamma \in [0,1]$ is the discount factor. In our scenario, the central controller is considered an agent. The MDP can be defined as follows.

i) *State*. At each time slot t, the agent observes the state information from the current vehicular network environment. The state s_t can be formulated as: $s_t = \{\mathcal{S}_t, \mathcal{V}_t\}$, where $\mathcal{S}_t$ indicates the state of all RSUs and $\mathcal{V}_t$ represents the information of currently arriving vehicles' tasks $\mathcal{I}_t$. The state of each RSU s includes: the computing resource capacity C_s, the bandwidth resource capacity B_s, and the workload already assigned to RSUs at the current time slot $D_{s,t}^{load} = \sum_{t':t'>=t} \sum_{i\in\mathcal{I}} \alpha_{i,s}^{t'} C_s$, *i.e.,* $\mathcal{S}_t = \{C_s, B_s, D_{s,t}^{load}\}_{s\in\mathcal{S}}$. The information of vehicle i includes: the travel speed v_i, the computing resource capacity c_i, the transmission power ρ_i, and the location $l_i(t)$, *i.e.,* $\mathcal{V}_t = \{v_i, c_i, \rho_i, l_i(t)\}_{i\in\mathcal{I}_t}$.

ii) *Action*. Given the observed state s_t, the agent determines the action a_t of the workload distribution for all vehicles $\mathcal{I}_t$ at time slot t, *i.e.,* the RSUs selection $\{h_{i,s}, d_{i,s}\}_{i\in\mathcal{I}_t, s\in\mathcal{S}}$ and DNN partition $\{p_{i,k}, e_{i,k}\}_{i\in\mathcal{I}_t, k\in\mathcal{K}_i}$. Therefore, the action a_t is defined as: $a_t = \{h_{i,s},\ d_{i,s}, p_{i,k}, e_{i,k}\}_{i\in\mathcal{I}_t, s\in\mathcal{S}, k\in\mathcal{K}_i}$.

iii) *Reward*. Given the state-action pair, the agent will receive a reward r_t from the environment to evaluate the quality of action a_t. Based on the objective of minimizing latency, the reward function follows the principle that actions leading to smaller latency are returned with larger rewards. Moreover, considering the constraints (2c) and (2d), we develop a reward to encourage the agent to select actions that satisfy constraints. The reward function is defined as follows:

$$r_t = -\frac{1}{\mathcal{I}_t} \sum_{i\in\mathcal{I}_t} r_{t,i}, \text{ where} \tag{3}$$

$$r_{t,i} = \begin{cases} -\mu_i, & \text{constrains (2c) and (2d) are satisfied,} \\ -\omega\mu_i, & \text{otherwise.} \end{cases}$$

$r_{t,i}$ denotes the reward of vehicle i, and ω indicates the penalty factor. When the constraints (2c) and (2d) are satisfied, the reward is the negative value of the latency. And an extremely small value $-\omega\mu_i$ is returned as a penalty for violating constraints. Note that there must be $\omega >> 1$ to incentivize the agent to select actions that satisfy the constraints. The specific set of parameters is listed in Sec. IV.

Challenge and Solution. In vehicular networks, it is challenging to model the environment state accurately due to the limited knowledge of transition probability and task arrival patterns of all vehicles. High-dimensional continuous state spaces and *high-dimensional discrete action spaces* further exacerbate convergence issues due to their computational cost [25]. Traditional dynamic programming solutions are ineffective in solving this MDP problem. DRL has emerged as a promising approach for MDP problems [26]. The selection of DRL is essential for achieving fast training speed and exploration capability given the high-dimensional tuple information and *multiple discrete actions* of the MDP. In this paper, we exploit the SAC-D [27], an off-policy actor-critic algorithm with soft policy updating based on the maximum entropy RL framework, to address the above MDP problem. It is specifically designed for discrete action spaces and can be applied in large-scale networks without requiring statistics on network dynamics, which is described in detail in Sec. III-D.

C. Solution for Resource Allocation

After obtaining the RSUs selection and DNN model partition strategy through the SAC-D based algorithm, the original problem (2) can be simplified to the following resource allocation subproblem:

$$\text{minimize} \quad \frac{1}{I} \sum_{i\in\mathcal{I}} \mu_i \tag{4}$$

$$\sum_{i\in\mathcal{I}} \alpha_{i,s}^t <= 1, \forall s \in \mathcal{S}, \forall t \in \mathcal{T}, \tag{4a}$$

$$\sum_{i\in\mathcal{I}} \beta_{i,s}^t <= 1, \forall s \in \mathcal{S}, \forall t \in \mathcal{T}. \tag{4b}$$

Since the workload distribution strategy is known, and the DNN inference phases are executed sequentially, the start time of each phase can be obtained. Therefore, resource allocation on each RSU is independent and problem (4) can be decomposed into a series of single-slot wireless bandwidth allocation problems and computing resource allocation problems.

Wireless Bandwidth Allocation. The wireless bandwidth allocation problem of RSU s at t can be modeled as follows:

$$\text{minimize} \sum_{i\in\mathcal{I}_{s,t}} \mu_i^o \tag{5}$$

$$\sum_{i\in\mathcal{I}} \beta_{i,s}^t <= 1, \tag{5a}$$

where $\mathcal{I}_{s,t}$ denotes the set of vehicles that need to allocate bandwidth by the RSU s for intermediate data transmission at time slot t. The above problem (5) is a convex optimization problem, which can be addressed directly by a convex solver, *e.g.*, CVX. But, considering the low latency requirement of online inference, we derive the following equations to quickly calculate the optimal solution according to the convex optimization theory and KKT condition [15], [28]:

$$\begin{aligned} \nabla(\sum_{i\in\mathcal{I}_{s,t}} \mu_i^o + \delta(\sum_{i\in\mathcal{I}} \beta_{i,s}^t - 1)) = 0, \\ \delta >= 0, \\ \delta(\sum_{i\in\mathcal{I}} \beta_{i,s}^t - 1) = 0, \\ \sum_{i\in\mathcal{I}} \beta_{i,s}^t - 1 <= 0. \end{aligned} \tag{6}$$

By solving the equations, the optimal bandwidth resource allocation for vehicles $i \in \mathcal{I}_{s,t}$ can be obtained as follows:

$$\begin{aligned} \beta_{i,s}^t &= \frac{\sqrt{\eta_i}}{\sum_{i\in\mathcal{I}_{s,t}} \sqrt{\eta_i}}, \forall i \in \mathcal{I}_{s,t}, \text{where} \\ \eta_i &= \frac{d_{i,\kappa_{i,2}}}{B_s \log_2(1 + \frac{\rho_i g_{i,s}^t}{\sigma^2})}, \forall i \in \mathcal{I}_{s,t}. \end{aligned} \tag{7}$$

Computing Resource Allocation. The computing resource allocation problem of RSU s at t can be formulated as follows:

$$\text{minimize} \sum_{i\in\mathcal{I'}_{s,t}} \frac{X_{i,s}}{\alpha_{i,s}^t C_s} \tag{8}$$

$$\sum_{i\in\mathcal{I}} \alpha_{i,s}^t <= 1, \tag{8a}$$

where $\mathcal{I'}_{s,t}$ denotes the set of vehicles that need to allocate computing resources by the RSU s for task inference at time slot t, $X_{i,s}$ indicates the computation workload of

vehicle i in RSU s. In helper computation phase, $X_{i,s} = \sum_{k:\kappa_{i,1}<k<=\kappa_{i,2}} x_{i,k}$, and $X_{i,s} = \sum_{k:k>\kappa_{i,2}} x_{i,k}$ for deliver computation phase. Similar to problem (4), problem (8) is also a convex optimization problem, and its solution is given by:

$$\alpha_{i,s}^{t} = \frac{\sqrt{\zeta_i}}{\sum_{i\in\mathcal{I}'_{s,t}} \sqrt{\zeta_i}}, \forall i \in \mathcal{I}'_{s,t}, \text{where}$$
$$\zeta_i = X_{i,s}/C_s, \forall i \in \mathcal{I}'_{s,t}. \tag{9}$$

D. SAC-D Based Collaborative Inference Framework

***Design of* SAFE.** As illustrated in Fig. 2, we propose a two-stage SAC-D based collaborative inference framework, SAFE, to handle the problem (2). SAFE includes four crucial elements to enhance performance.

1) Maximum entropy framework to ensure exploration and stability. For our algorithm, the goal of the agent is to find a policy π^* that maximizes the maximum entropy objective:

$$\pi^* = \arg\max_{\pi} \sum_{t\in\mathcal{T}} \mathbb{E}_{(s_t,a_t)\sim\xi_\pi}[\gamma^t(r_t + \lambda\mathcal{H}(\pi(.|s_t)))], \tag{10}$$

where λ denotes the temperature parameter that balances the reward and entropy, ξ_π indicates the distribution of trajectories induced by policy π, and $\mathcal{H}(\pi(.|s_t))$ represents the entropy of the policy π at state s_t.

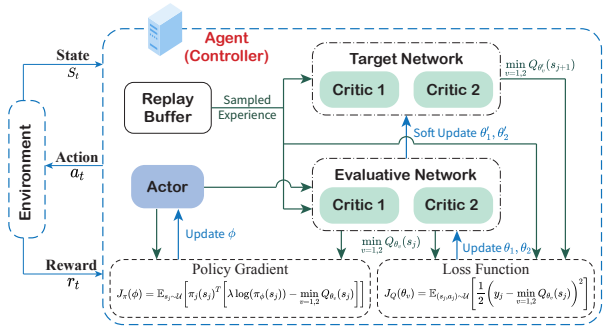

Fig. 2: The structure of SAFE.

2) An actor-critic architecture with an actor network, a pair of evaluative critic networks, and a pair of target critic networks. The clipped double Q-networks technique is utilized for both the evaluation critic networks and the target critic networks to mitigate the problem of Q-value overestimation and speed up training [29]. The actor makes action decisions based on its policy $\pi_\phi(s)$. The evaluative critic networks provide a pair of Q-values $(Q_{\theta_1}, Q_{\theta_2})$ to evaluate the actor's actions, while the target critic networks calculate $(Q_{\theta'_1}, Q_{\theta'_2})$.

3) An off-policy way with the experience replay technique to accelerate the convergence efficiency. The experiences of the agent (*i.e.*, transitions $\{s_t, a_t, r_t, s_{t+1}\}_{t\in\mathcal{T}}$) are stored into the replay buffer, which will be sampled randomly to train the network parameters later in the training stage.

4) Discrete actions which are suitable for our transformed MDP problem. In the output layer of the actor network, decision elements are addressed via discretization. And critic networks output the Q-value of each possible action rather than simply providing the action as input.

***DRL Network Parameter Update*.** During the training stage of our algorithm, the agent randomly samples a mini-batch experiences $\{s_j, a_j, r_j, s_{j+1}\}_{j\in\mathcal{F}_j}$ from replay buffer $\mathcal{D}$ to update the parameters of the actor network and critic networks. Take the j-th transition $\{s_j, a_j, r_j, s_{j+1}\}$) as an example. The target Q-values are calculated by target critic networks as:

$$y_j = r_j + \gamma(\min_{v=1,2} Q_{\theta'_v}(s_{j+1}) - \lambda \log \pi_\phi(s_{j+1})). \tag{11}$$

Then evaluative critic networks update the parameters independently by minimizing the loss function, which is given by:

$$J_Q(\theta_v) = \mathbb{E}_{(s_j,a_j)\sim\mathcal{U}}[\frac{1}{2}(y_j - \min_{v=1,2} Q_{\theta_v}(s_j))^2], \tag{12}$$

where $\mathcal{U}$ denotes the mini-batch sampled from the replay buffer. Using the Stochastic Gradient Descent (SGD) method, the parameters θ_v are adjusted in the direction of $\nabla_{\theta_v} J_Q(\theta_v)$.

To update the actor network, the objective function is:

$$J_\pi(\phi) = \mathbb{E}_{s_j\sim\mathcal{U}}[\pi_j(s_j)^T[\lambda \log(\pi_\phi(s_j)) - \min_{v=1,2} Q_{\theta_v}(s_j)]], \tag{13}$$

and the temperature parameter λ can be updated through minimizing entropy objective function, which is defined as:

$$J(\lambda) = \pi_j(s_j)^T[-\lambda(\log(\pi_t(s_t)) + \hat{H})], \tag{14}$$

where $\hat{H}$ is a constant and denotes the target entropy. Similar to evaluative critic networks, the parameters of the actor network and the temperature are updated using the SGD method.

***Algorithm Details*.** Our SAC-D based collaborative inference framework, SAFE, is presented in Alg. 1 as follows:

Initialization. First, initialize the parameters of the actor network, evaluative critic networks, and target critic networks of the agent (lines 1-2). Furthermore, line 3 initializes an empty experience replay buffer $\mathcal{D}$ and several related hyper-parameters. The algorithm requires E episodes of iterations. At the beginning of each episode, initialize the environment and the agent can observe the initial state s_1 (line 5).

Environment Step. Given the state information $s_t = \{\{C_s, B_s, D_{s,t}^{load}\}_{s\in\mathcal{S}}, \{v_i, c_i, \rho_i, l_i(t)\}_{i\in\mathcal{I}_t}\}$ from the current environment, the actor of the agent decides discrete actions of workload distribution $a_t = \{h_{i,s}, d_{i,s}, p_{i,k}, e_{i,k}\}_{i\in\mathcal{I}_t, s\in\mathcal{S}, k\in\mathcal{K}_i}$ based on its policy π_ϕ (line 7). Then in line 8, the decisions of resource allocation $a^{res} = \{\alpha_{i,s}^t, \beta_{i,s}^t\}_{i\in\mathcal{I}_t}$ are obtained based on the decomposition method and convex theory. After vehicles $\mathcal{I}_t$ carry out the joint actions $\{a_t, a^{res}\}$, the environment is updated and interacts with the agent. The agent receives the reward r_t and observes the next state s_{t+1} (line 9). Finally, line 10 stores the tuple $\{s_t, a_t, r_t, s_{t+1}\}$ of experience into replay buffer $\mathcal{D}$, which will be sampled for networks training.

Training Step. In the training stage, the agent updates the parameters of the actor network and critic networks. In line 11, the agent randomly samples a mini-batch of U experiences from replay buffer $\mathcal{D}$. Based on these experiences, the parameters of the evaluate critic networks θ_v, the actor networks ϕ, and the temperature λ are updated by minimizing Eq. (12), Eq. (13), and Eq. (14) (lines 12-14). Finally, the soft-updating method is used to update the parameters θ'_v of target critic networks, where τ denotes the update factor (line 15).

E. Discussion

SAFE can be expanded to include more RSUs. For example, using three RSUs would involve three model partition layers. The third RSU only assists in computing intermediate layers of the DNN model. In this case, the decision variables need to include three RSUs and three partitioning layers. To handle these additional decisions, binary variables are introduced to

Algorithm 1 SAC-Discrete Based Collaborative Inference FramEwork (SAFE)

Initialize actor network $\pi_\phi(s)$ and evaluate critic networks $Q_{\theta_1}, Q_{\theta_2}$ with parameters ϕ, θ_1, θ_2;
Initialize target critic networks $Q_{\theta_1'}, Q_{\theta_2'}$ with $\theta_1' \leftarrow \theta_1, \theta_2' \leftarrow \theta_2$;
Initialize an empty experience replay buffer $\mathcal{D}$ and hyperparameters γ, λ, τ, ℓ_Q, ℓ_π, ℓ_λ, $\hat{\mathcal{H}}$;
for $episode = 1$ to E **do**
  Initialize the environment, and receive the initial state s_1;
  for $t = 1$ to T **do**
    Select action with the current policy $a_t \sim \pi_\phi(s_t)$;
    Obtain the resource allocation decisions a^{res} by Eq. (7) and Eq. (9);
    Apply the actions $\{a_t, a^{res}\}$ in the environment, receive the reward r_t, and observe the next state s_{t+1};
    Store $\{s_t, a_t, r_t, s_{t+1}\}$ into replay buffer $\mathcal{D}$;
    Randomly sample a mini-batch of U experiences $\{s_j, a_j, r_j, s_{j+1}\}_{j \in \mathcal{D}'}$ from replay buffer $\mathcal{D}$;
    Update θ_1, θ_2 by minimizing the loss function in Eq. (12): $\theta_v \leftarrow \theta_v - \ell_Q \nabla_{\theta_v} J_Q(\theta_v), v = 1, 2$;
    Update policy weight ϕ via gradient of Eq. (13): $\phi \leftarrow \phi - \ell_\pi \nabla_\pi J_\pi(\phi)$;
    Update temperature λ via gradient of Eq. (14): $\lambda \leftarrow \lambda - \ell_\lambda \nabla_\lambda J(\lambda)$;
    Update the parameters of target critic networks: $\theta_v' \leftarrow \tau\theta_v + (1-\tau)\theta_v', v = 1, 2$.
  end for
end for

expand the action structure of the MDP. Same as SAFE, a DRL algorithm can be trained to solve this extended MDP and determine the RSU selection and model partition. The optimal resource allocation also can be calculated using Eq. (7) and Eq. (9). Furthermore, except for the straight road, SAFE can also handle complex road conditions. It is feasible because vehicles are often guided by the navigation system, which can provide information such as the vehicle's speed and movement direction, the set of available RSUs, and so on.

IV. Performance Evaluation

A. Experiments Setup

Vehicle-edge System. We implement a vehicle-edge system with two types of devices: MacBook Pro 2020 with chip M1 and desktop PC. We take ten MacBook Pros (which utilize only one CPU core) to emulate vehicles. Five desktop PCs are employed as the RSUs. Each desktop PC is equipped with 12 CPU cores, 16GB RAM, 500GB HDDs, and a dual-port 1GbE NIC. In addition, another desktop PC is served as the central controller, which does not participate in task inference. We consider a road with five RSUs whose signal coverage radius is 300 m. The length of each zone is 20 m. The wireless bandwidth capacity of RSU is $[5, 20]$ MHz (default $B_s = 10$ MHz). Vehicles generate inference tasks randomly. The travel speed v_i is $[36, 72]$ km/h. Due to budget limitations, we cannot deploy physical vehicles and RSUs. Therefore, we used a data simulation approach to realize the movement of vehicles and the communication between vehicles and RSUs. Similar to [22], the transmission power ρ_i is set within $[5, 10]$ dBm, the channel gain $g_{i,s}^t$ follows $-(128.1 + 37.6 \log_{10} d)$, where d (in km) indicates the distance between vehicle i and RSU s, and the Gaussian noise σ^2 is set to -174 dBm/Hz.

Workload. In our experiments, three well-known DNN models, AlexNet [30], VGG-16 [31], and ResNet50 [32] are considered (VGG-16 is used by default). We implement all DNN models with PyTorch in Python. Both training and inference of DNN models are performed using the Berkeley Deep Drive data set (BDD100k) [33]. This dataset contains 120M images from 100K videos captured by cameras on self-driving cars. Specifically, the input data is a $1280 * 720$ image with 3 channels. All DNN models are pre-trained and then pre-allocated on both vehicles and RSUs before the inference.

TABLE II: Parameter Setting of SAFE

Parameter	Value	Parameter	Value
Number of episodes E	3000	Number of steps T	200
Replay buffer size $\lvert\mathcal{D}\rvert$	10000	Mini-batch size U	100
Learning rate $\ell_Q, \ell_\pi, \ell_\lambda$	0.0001	Discount factor γ	0.99
Temperature initial λ	1.0	Target entropy $\hat{H}$	-log$(1+\iota)$
Soft update factor τ	0.01	Optimizer	Adam
Hidden layer act.	ReLU	Actor output act.	Softmax

Algorithm Networks. In our DRL, the actor network and critic networks of the agent are all four-layer neural networks, which consist of an input layer, an output layer, and two hidden layers. The number of neurons in the hidden layers are 512 and 256, respectively. The penalty factor of reward ω is set to 3. Other parameters of SAFE are listed in TABLE II, where ι denotes the output dimension of the actor network. The DRL in our framework is also implemented by PyTorch.

Baselines. To evaluate the performance of SAFE, the following four baselines are compared.

- *Local*: all DNN inference computations are completed on vehicles without any assistance.
- *Edge*: vehicles directly upload the input data to the nearby RSU, and all computations are done on this RSU.
- *DSL* [6]: *DSL* is a state-of-art DNN partition scheme, which executes the inference tasks with the cooperation of the nearby RSU. In this approach, both RSUs and vehicles are capable of running only one task at a time, utilizing all available resources.
- *INSGA* [34]: *INSGA* is a specialized algorithm designed for resource allocation in vehicular networks. It offloads inference tasks to nearby RSUs and makes resource allocation decisions based on an improved non-dominated sorting genetic algorithm.

B. Evaluation Results

Evaluation Metrics. Given that vehicles may be driven out of the signal coverage of the assist RSU, not all vehicles can complete the inference task computation and delivery. Therefore, we consider the following metrics to evaluate the performance of SAFE. i) **Success rate**, which is defined as the number of completed tasks over the total number of inference tasks. ii) **Average latency** of all completed tasks. iii) **Success rate improvement**, which represents the increased success rate compared to the solution found by *Edge*. Since the reliability of the wireless link is not taken into account

by *Local*, we compare the success rate improvement with *Edge* instead. iv) **Latency improvement**, which indicates the percentage reduction in average latency compared to the value returned by *Local*. Note that the success rate of the *Local* is 100% since it does not involve wireless transmission, and thus does not need to consider the reliability of the wireless link. Therefore, we did not compare *Local* with other algorithms on this metric.

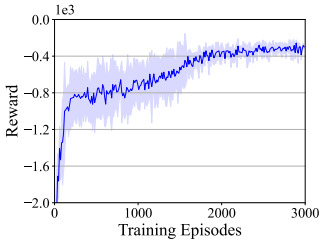

Fig. 3: The convergence performance of SAFE.

Fig. 4: The convergence performance details of SAFE.

***Convergence of* SAFE.** The convergence performance of our framework SAFE is presented in Fig. 3. The light blue region represents the standard deviation. We can observe that as the number of training episodes increases, the reward value rises gradually until it reaches a relatively stable value. It validates that SAFE converges after parameter iterations for 1800 episodes. Specifically, we plot the success rate and the average latency under each iteration in Fig. 4. It can be observed that the average latency decreases and the success rate rise gradually as the number of episodes grows, which further proved that SAFE has a good convergence effect, *i.e.*, small average latency with a high success rate.

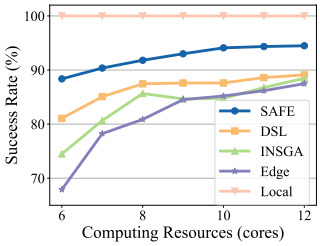

Fig. 5: Success rate on VGG-16 with different C_s.

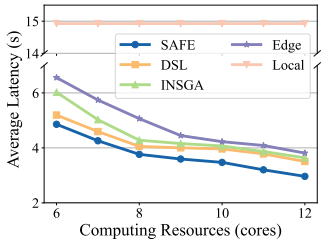

Fig. 6: Average latency on VGG-16 with different C_s.

Impact of Computing Resources. After well offline training, we evaluate the online performance of SAFE. Fig. 5 and Fig. 6 illustrate the success rate and the average latency of all algorithms with VGG-16 in terms of the computing resources capacity in each RSU, respectively. The results show that the average latency decreases as the amount of resources increases, while the success rate increases gradually. This is attributed to the fact that the computation latency on RSUs is influenced by the available resource capacity. With sufficient resources, the latency decreases, reducing the likelihood of vehicles traveling out of RSU signal coverage and thereby increasing the success rate. In addition, SAFE performs better than the four baselines. This is due to SAFE's ability to capture dynamic changes in the environment, such as vehicle location, RSU workload, and channel conditions. The DRL-based framework SAFE targets the long-term performance to adapt to a dynamic environment, while baselines focus only on the immediate performance.

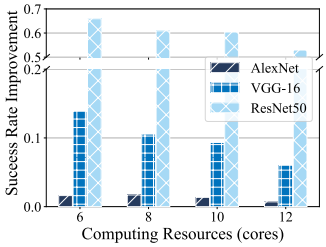

Fig. 7: Success rate improvement with different C_s.

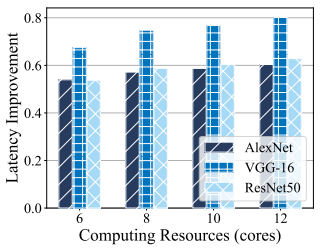

Fig. 8: Latency improvement with different C_s.

We further depict the impact of computing resources on the performance of SAFE on different DNN models in Fig. 7 and Fig. 8. Note that, to provide a comprehensive analysis, we evaluate the performance of SAFE in comparison to baselines using two previously defined metrics: success rate improvement and latency improvement. Several conclusions can be drawn from the figures. First, SAFE surpasses baselines in terms of both latency and success rate across all DNN models. Second, the success rate improvement is more significant for models with higher computational requirements, such as ResNet50. These models have longer inference latency, increasing the likelihood of the vehicle leaving the RSU's signal coverage. SAFE adaptively selects two RSUs to ensure reliable inference. Third, computation-intensive models are more sensitive to changes in computing resources. Specifically, SAFE can boost success rate by up to 65% while reducing latency by 50% to 80%.

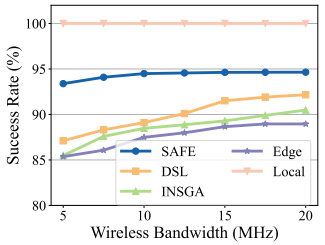

Fig. 9: Success rate on VGG-16 with different B_s.

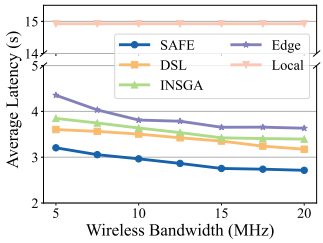

Fig. 10: Average latency on VGG-16 with different B_s.

Impact of Wireless Bandwidth. Fig. 9 and Fig. 10 present the impact of wireless bandwidth on the performance of SAFE. It can be observed that as bandwidth grows, average latency decreases, and the success rate rises. The reason is that sufficient wireless bandwidth resources reduce the data transmission latency in V2I communication. Furthermore, SAFE consistently outperforms the four baselines and can adapt to changes in the available wireless bandwidth.

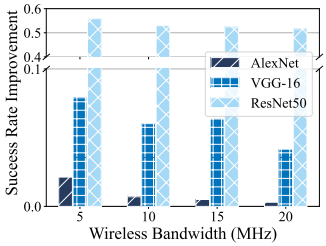

Fig. 11: Success rate improvement with different B_s.

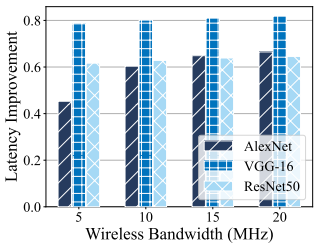

Fig. 12: Latency improvement with different B_s.

Similarly, the success rate improvement and latency improvement of SAFE under different wireless bandwidths on different DNN models are displayed in Fig. 11 and Fig. 12. Two figures further demonstrate the excellent performance of SAFE and its application features in different DNN models. In addition, DNN models with lower compute requirements (*e.g.*, AlexNet) are more sensitive to changes in bandwidth. This is because, despite having the same unit input/output data size, models with lower compute requirements have a higher communication-to-computation ratio. Therefore, changes in bandwidth have a greater impact on the overall latency.

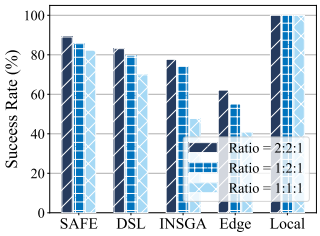

Fig. 13: Success rate with mixed DNN models.

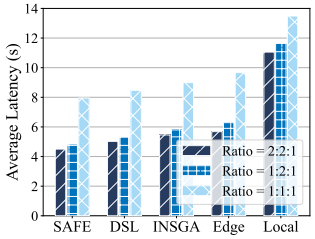

Fig. 14: Average latency with mixed DNN models.

Mixed DNN models. In real-world scenarios, vehicles often employ a variety of DNN models. To address this, we conducted experiments using mixed DNN models. The task's DNN model type $u_i \in \{0, 1, 2\}$ employed by vehicle i is added to the state so that SAFE can adapt to different model features. The results under the different ratios of the three models used for inference tasks (*i.e.*, ratio = AlexNet : VGG-16 : ResNet50) are presented in Fig. 13 and Fig. 14. It can be seen that SAFE enhances the success rate by 10% to 20% while reducing the average latency by up to 50%. Although the performance of SAFE is slightly lower compared to the single model scenario (Fig. 5 - Fig. 12), it still outperforms the four baselines.

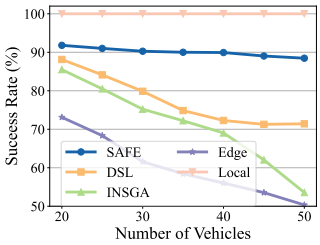

Fig. 15: Success rate on VGG-16 with different I.

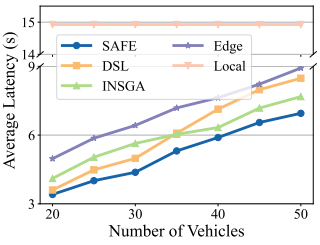

Fig. 16: Average latency on VGG-16 with different I.

Large Scale. To assess the effectiveness of SAFE on a broader scale, we conducted experiments with 10 RSUs and $[20, 50]$ vehicles. The results are presented in Fig. 15 and Fig. 16. It can be observed that SAFE consistently outperforms the four baselines, especially in larger-scale scenarios. In addition, as the number of vehicles increases, average latency of *DSL* becomes worse compared to *INSGA*. This is because suitable resource allocation strategies effectively improve resource utilization, reducing latency in resource-constrained scenarios.

V. Related Work

A. DNN Inference Acceleration

DNN Model Optimization. This type of work focuses on reducing parameters and computation in DNN models, using techniques including model pruning [35], model quantization [36], model compression [11], and knowledge distillation [37]. For example, Liu *et al.* [12] propose AutoCompress, an automatic systematic pruning framework with a high compression ratio. Moreover, the early-exit technique is commonly used to reduce computation without compressing the model [38]. This approach capitalizes on the fact that outputs from earlier layers are usually adequate for inference, avoiding further computation at higher layers [37], [39]. However, these methods often come at the cost of sacrificing DNN inference accuracy. To ensure inference accuracy, we keep DNN models unchanged and adopt the DNN partitioning method.

Collaborative DNN inference. To support latency-sensitive applications, collaborative DNN inference [5], [40] have been used to reduce inference latency by leveraging DNN partitioning and offloading partial computation from the resource-constrained local to nearby edge devices [41], edge servers [13], and powerful cloud [42]. Mohammed *et al.* [14] design a fine-grained adaptive DNN partitioning strategy and a distributed offloading algorithm based on a matching game to minimize latency. Hu *et al.* [43] propose a distributed inference mechanism with progressive model partitioning to enhance run-time performance on edge devices. Huang *et al.* [44] present a novel adversarial group linear bandits algorithm for collaborative edge inference. However, these works overlook the mobility of vehicles in dynamic vehicular networks, which may lead to inference task failures. In this paper, considering the features of vehicular networks such as dynamic changes in vehicle location and network conditions, we adapt the collaborative vehicle-edge inference method and offload partial workload to RSUs to speed up inference.

B. Vehicular Edge Computing

In recent years, great efforts have been paid to computation offloading and resource allocation in vehicular networks [15], [16]. Wu *et al.* [45] propose a dynamic radio access network (RAN) slicing framework for different latency-sensitive vehicle network tasks. Considering the vehicle mobility dynamics, Li *et al.* [46] present a stochastic scheduling scheme to minimize the traveled distance of vehicles. To ensure task computation and delivery, Li *et al.* [17] select three roles of RSUs for task offloading, computing, and result delivery respectively. There are few existing studies on DNN inference tasks in vehicular networks. Yang *et al.* [47] present a two-tier edge AI empowered autonomous driving framework. Wang *et al.* [9] design a chemical reaction optimization based algorithm with the best partition point selection scheme for joint vehicle-edge DNN inference. However, they only focus on the allocation of computing resources and ignore the wireless bandwidth, which is scarce and unstable in the edge environment. The above DNN inference researches all neglect the dynamic mobility of

vehicles, which is a major feature of vehicular networks. In this paper, we comprehensively consider DNN model characteristics and vehicle mobility, and select a cooperative RSU and a result-delivered RSU for the vehicle task to complete the inference and minimize the latency.

VI. CONCLUSION

In this paper, we propose a collaborative DNN inference framework, SAFE, for vehicular networks. Our goal is to reduce the inference latency under the limited computing resource and wireless bandwidth capacity constraints of RSUs. We decouple the problem into two subproblems: workload distribution and resource allocation. We first present a SAC-D based algorithm to solve the workload distribution subproblem to decide the RSU selection and DNN partition. Second, the resource allocation subproblem which calculates the amount of allocated computing resources and wireless bandwidth of selected RSUs to the vehicle is solved by convex optimization theory. Extensive experiments show that SAFE can adapt to a highly dynamic environment and significantly reduce inference latency. For future work, we will consider the synchronization of DNN model parameter updates at different locations to achieve accurate and reliable inference services. Moreover, privacy-preserving will also be considered, such as security problems when transferring shallow data of DNN models between vehicle and helper.

REFERENCES

[1] *Self-driving Safety Report*, 2020, https://resources.nvidia.com/en-us-auto-safety/auto-safety-report.

[2] Y. Zhang, L. Zhao, G. Zheng, X. Chu, Z. Ding, and K.-C. Chen, "Resource allocation for open-loop ultra-reliable and low-latency uplink communications in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 3, pp. 2590–2604, 2021.

[3] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1655–1674, 2019.

[4] E. Arnold, O. Y. Al-Jarrah, M. Dianati, S. Fallah, D. Oxtoby, and A. Mouzakitis, "A survey on 3d object detection methods for autonomous driving applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 10, pp. 3782–3795, 2019.

[5] N. D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, L. Jiao, L. Qendro, and F. Kawsar, "Deepx: A software accelerator for low-power deep learning inference on mobile devices," in *Proc. of IPSN*, 2016.

[6] C. Hu, W. Bao, D. Wang, and F. Liu, "Dynamic adaptive dnn surgery for inference acceleration on the edge," in *Proc. of IEEE INFOCOM*, 2019.

[7] L. Zhang, L. Chen, and J. Xu, "Autodidactic neurosurgeon: Collaborative deep inference for mobile edge intelligence via online learning," in *Proc. of WWW '21*, 2021, p. 3111–3123.

[8] B. Ma, Z. Ren, and W. Cheng, "Traffic routing-based computation offloading in cybertwin-driven internet of vehicles for v2x applications," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 5, pp. 4551–4560, 2022.

[9] Q. Wang, Z. Li, K. Nai, Y. Chen, and M. Wen, "Dynamic resource allocation for jointing vehicle-edge deep neural network inference," *Journal of Systems Architecture*, vol. 117, p. 102133, 2021.

[10] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," in *Proc. of Springer ECCV*, 2020.

[11] M. Yin, Y. Sui, S. Liao, and B. Yuan, "Towards efficient tensor decomposition-based dnn model compression with optimization framework," in *Proc. of IEEE CVPR*, 2021.

[12] N. Liu, X. Ma, Z. Xu, Y. Wang, J. Tang, and J. Ye, "Autocompress: An automatic dnn structured pruning framework for ultra-high compression rates," in *Proc. of AAAI*, 2020.

[13] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge ai: On-demand accelerating deep neural network inference via edge computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 447–457, 2020.

[14] T. Mohammed, C. Joe-Wong, R. Babbar, and M. D. Francesco, "Distributed inference acceleration with adaptive dnn partitioning and offloading," in *Proc. of IEEE INFOCOM*, 2020.

[15] P. Dai, K. Hu, X. Wu, H. Xing, and Z. Yu, "Asynchronous deep reinforcement learning for data-driven task offloading in mec-empowered vehicular networks," in *Proc. of IEEE INFOCOM*, 2021.

[16] J. Zhang, S. Chen, X. Wang, and Y. Zhu, "Deepreserve: Dynamic edge server reservation for connected vehicles with deep reinforcement learning," in *Proc. of IEEE INFOCOM*, 2021.

[17] M. Li, J. Gao, L. Zhao, and X. Shen, "Deep reinforcement learning for collaborative edge computing in vehicular networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 4, pp. 1122–1135, 2020.

[18] N. Shan, Z. Ye, and X. Cui, "Collaborative intelligence: Accelerating deep neural network inference via device-edge synergy," *Security and Communication Networks*, vol. 2020, pp. 1–10, 2020.

[19] C. Dong, S. Hu, X. Chen, and W. Wen, "Joint optimization with dnn partitioning and resource allocation in mobile edge computing," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 3973–3986, 2021.

[20] C. Cai, G. Zhou, H. Zhuo, G. Tan, H. Deng, and J. Chen, "Empowering vehicular networking capability: White paper for operators," China Mobile Research Institute, 2022.

[21] F. M. C. Forum, "5g vehicular communication technology," 2017.

[22] L. Xu, M. Qin, Q. Yang, and K.-S. Kwak, "Learning-aided dynamic access control in mec-enabled green iot networks: A convolutional reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 2, pp. 2098–2109, 2022.

[23] Y. Chen, Z. Liu, Y. Zhang, Y. Wu, X. Chen, and L. Zhao, "Deep reinforcement learning-based dynamic resource management for mobile edge computing in industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 4925–4934, 2021.

[24] A. Schrijver, *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[25] W. Zhang, D. Yang, H. Peng, W. Wu, W. Quan, H. Zhang, and X. Shen, "Deep reinforcement learning based resource management for dnn inference in industrial iot," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 8, pp. 7605–7618, 2021.

[26] A. Mekrache, A. Bradai, E. Moulay, and S. Dawaliby, "Deep reinforcement learning techniques for vehicular networks: Recent advances and future trends towards 6g," *Vehicular Communications*, vol. 33, p. 100398, 2022.

[27] P. Christodoulou, "Soft actor-critic for discrete action settings," *CoRR*, vol. abs/1910.07207, 2019.

[28] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[29] S. Chen, X. Qiu, X. Tan, Z. Fang, and Y. Jin, "A model-based hybrid soft actor-critic deep reinforcement learning algorithm for optimal ventilator settings," *Information Sciences*, vol. 611, pp. 47–64, 2022.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Proc. of NIPS*, 2012.

[31] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[32] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. of IEEE CVPR*, 2017.

[33] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving video database with scalable annotation tooling," *arXiv preprint arXiv:1805.04687*, vol. 2, no. 5, p. 6, 2018.

[34] W. Wei, R. Yang, H. Gu, W. Zhao, C. Chen, and S. Wan, "Multi-objective optimization for resource allocation in vehicular cloud computing networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 25 536–25 545, 2022.

[35] T. Jian, D. Roy, B. Salehihikouei, N. Soltani, K. Chowdhury, and S. Ioannidis, "Communication-aware dnn pruning," in *Proc. of IEEE INFOCOM*, 2023.

[36] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "Haq: Hardware-aware automated quantization with mixed precision," in *Proc. of IEEE CVPR*, 2019.

[37] W. Fang, F. Xue, Y. Ding, N. Xiong, and V. C. M. Leung, "Edgeke: An on-demand deep learning iot system for cognitive big data on industrial edge devices," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 9, pp. 6144–6152, 2021.

[38] S. Teerapittayanon, B. McDanel, and H. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *Proc. of ICPR*, 2016.

[39] Z. Huang, F. Dong, D. Shen, J. Zhang, H. Wang, G. Cai, and Q. He, "Enabling low latency edge intelligence based on multi-exit dnns in the wild," in *Proc. of IEEE ICDCS*, 2021.

[40] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGARCH Computer Architecture News*, vol. 45, no. 1, pp. 615–629, 2017.

[41] N. Chen, Z. Shuai, S. Zhang, Y. Yan, Y. Chen, and S. Lu, "Resmap: Exploiting sparse residual feature map for accelerating cross-edge video analytics," in *Proc. of IEEE INFOCOM*, 2023.

[42] S. Laskaridis, S. I. Venieris, M. Almeida, I. Leontiadis, and N. D. Lane, "Spinn: synergistic progressive inference of neural networks over device and cloud," in *Proceedings of MobiCom*, 2020.

[43] C. Hu and B. Li, "Distributed inference with deep learning models across heterogeneous edge devices," in *Proc. of IEEE INFOCOM*, 2022.

[44] Y. Huang, L. Zhang, and J. Xu, "Adversarial group linear bandits and its application to collaborative edge inference," in *Proc. of IEEE INFOCOM*, 2023.

[45] W. Wu, N. Chen, C. Zhou, M. Li, X. Shen, W. Zhuang, and X. Li, "Dynamic ran slicing for service-oriented vehicular networks via constrained learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 2076–2089, 2021.

[46] M. Li, J. Gao, L. Zhao, and X. Shen, "Adaptive computing scheduling for edge-assisted autonomous driving," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 6, pp. 5318–5331, 2021.

[47] B. Yang, X. Cao, K. Xiong, C. Yuen, Y. L. Guan, S. Leng, L. Qian, and Z. Han, "Edge intelligence for autonomous driving in 6g wireless system: Design challenges and solutions," *IEEE Wireless Communications*, vol. 28, no. 2, pp. 40–47, 2021.