Multi-modal Data and Model Reduction for Enabling Edge Fusion in Connected Vehicle Environments

Samuel Thornton, *Student, IEEE,* and Sujit Dey, *Fellow, IEEE*

***Abstract*—The emergence of edge computing in Intelligent Transportation Systems (ITS) has shown promise in enabling real-time sensor fusion applications. In this paper, we explore how to utilize edge computing to aid in collaborative vehicular perception, an emerging topic within ITS which involves vehicles sharing sensor data with one another to extend each vehicle's perception beyond what its individual sensors can see. However, achieving real time collaborative perception is a challenge even with the utilization of edge computing; due to the large amount of multi-modal sensor data produced by modern intelligent vehicles, the amount of data transmitted over wireless channels and the complexity of computational tasks will need to be managed dynamically based on the wireless network conditions and available computing resources. As such we propose REFO, a Real-time Edge Fusion Optimization method that combines task partitioning with data reduction and model compression to maximize sensor fusion accuracy while adhering to Quality of Service (QoS) requirements. We define a performance metric termed effective mean average precision (EmAP), which incorporates both QoS and fusion accuracy, and show that our proposed neural network based REFO action decision framework can outperform the best comparison models by approximately 10% in terms of EmAP over 18 test combinations of network and computing conditions.**

***Index Terms*—Connected Vehicles, Data Reduction, Model Compression, Task Partitioning, Machine Learning**

I. Introduction

The intelligence of vehicles on the road today is evolving at a rapid pace. Production vehicles are being released with more sensors and computing power with each passing year that enable the vehicle to sense its mechanical performance as well as the presence and activity of objects inside the cabin and in the surrounding environment. In recent years, external sensing has become an important topic due to increased public interest in vehicular safety and autonomous driving. The Advanced Driver Assistance System (ADAS), which gives alerts to the driver about surrounding or upcoming hazards, is now standard in newly produced vehicles. The ADAS is entirely reliant on the vehicle's external sensors which can include instruments such as RGB cameras, depth cameras, lidar, radar, or ultrasonic sensors. While the ADAS does provide important safety benefits, these systems are still far from perfect and are prone to errors. Additionally, environmental conditions such as bad weather or objects occluding sensors on the vehicle can affect the ability of the ADAS to accurately perceive its surroundings.

One approach to minimize the perception errors experienced by a vehicle's ADAS is to have multiple vehicles share their sensor data with each other [1] [2]. In this way, a more complete perception of an area can be achieved; gaps in coverage from one vehicle may be filled in by another and objects that were seen by multiple vehicles can have their perception improved. The emergence of edge-based communications can provide a suitable infrastructure for facilitating this sensor sharing, but creating this cooperative perception in real time is challenging. Having to aggregate all of the sensor data from each vehicle in a particular area, apply the sensor fusion algorithms, and send the information back to each vehicle quickly enough to satisfy a Quality of Service (QoS) requirement is difficult considering the size of the sensor data and challenges experienced with wireless vehicular networking.

To deal with the dynamic nature of vehicular environments, new methods must be developed in order to meet the latency demands of these real-time systems. While emerging 5G networks can be utilized to bring new levels of wireless communication to vehicles, there are always going to be situations encountered in vehicular environments that can affect the vehicles ability to communicate; driving through a tunnel or simply being surrounded by large objects/buildings can have a significant effect on the amount of data that the vehicle can transmit and receive. As such, the type and amount of data that should be transmitted along with the corresponding fusion models must be adjusted dynamically so that the QoS of this edge fusion system can be maintained while maximizing the sensor fusion accuracy. To address these issues we created **REFO**, a **R**eal-time **E**dge **F**usion **O**ptimization method. The focus of this work is not on creating new methods for improving the accuracy of object detection/association or related vehicular sensor fusion tasks, but rather to use a variety of existing data extraction, classification, and fusion techniques to explore the feasibility of real-time edge fusion in real-world environments. The REFO methodology we propose provides a model for facilitating vehicular data exchange over the edge that provides the computational tools and decision making algorithms for executing the end-to-end collaborative sensor fusion process. The testing results of our proposed method demonstrates successful performance over 18 different combinations of network and computing conditions while our proposed action decision framework outperforms the best comparison models by approximately 10% over all test cases. More specifically, the research contributions of this paper are as follows:

- We present REFO, a method for dynamically reducing the

Copyright (c) 2024 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to pubs-permissions@ieee.org.

Manuscript submitted October 30, 2023; revised January 23, 2024; accepted March 5, 2024.

The authors are with the department of Electrical and Computer Engineering and the Center for Wireless Communications, University of California San Diego 92093 USA (e-mail: {sjthornt, dey}@ucsd.edu)

amount of data and computational inference experienced in wireless edge-based vehicular sensor fusion in connected vehicle environments.
- We propose an ensemble fusion model which contains all the perception and fusion abilities needed for collaborative perception and has the capacity to be actively compressed and partitioned based on the current networking, computing, and traffic conditions.
- We have created an action decision framework to determine the optimal data reduction, model compression, and task offloading actions that should be chosen to satisfy a given QoS requirement while maximizing perception accuracy.
- We have developed a Deep Neural Network (DNN) based latency prediction model to estimate whether the latency requirements will be met for a given set of conditions.

The remainder of this paper will be organized as follows: Section II will be a review of related works in the area of vehicular edge computing, specifically in other works that involve external sensing, task partitioning, data reduction, and model compression. In Section III, this edge fusion problem is examined in more detail and an overview of the REFO method is presented as well as our problem formulation and latency model. In Section IV, we discuss our methodology for generating and executing actions decisions which includes the action decision framework and ensemble fusion model as well as an overview of the different offloading levels. In Section V, we describe our research setup and present the results of our action decision framework on both cellular vehicle-to-everything (C-V2X) and 5G cellular test sets as well as comparing the performance to related models. Finally, we conclude the paper and discuss our plans for future work in Section VI.

II. Related Work

A. External Sensing

Advancements in methods for utilizing sensor data to detect the presence of objects and characteristics of the surrounding environment have paved the way for the intelligence that exists inside modern vehicles. The most studied type of external sensing is that involving RGB cameras and the most common tasks for sensing using RGB images are image classification, image segmentation, and object detection. Early methods for solving these types of problems included transforms [3] [4] [5], feature descriptors [6] [7] [8] [9], part-based models [10] [11], and bag-of-words methods [12] [13], but the invention of AlexNet [14] and subsequent neural network architectures such as GoogleNet [15] and ResNet [16] have shown new heights for accuracy in all of these sensing tasks. As such even more advanced methods for general image classification [17], object detection [18], and image segmentation [19] have been developed in the past decade as well as the emergence of task specific models that are trained for more specific purposes such as defect detection [20] [21] [22], medical imagery [23] [24] [25], and satellite imagery [26] [27] [28].

The advancements in external sensing using images has extended to other sensors as well. Radar is another modality that has been around for a long time for external sensing purposes. As with images, previous methods for external sensing with physics based methods [29] [30] have been largely replaced by machine learning methods [31] [32] [33] [34]. However, lidar, a relatively new sensing modality has shown to have the best achievable performance for 3-D external sensing for a single sensor and thus there have been many methods proposed over the past decade for external sensing tasks that use lidar data as input [35] [36] [37]. A main drawback of using lidar is the massive amount of data the sensor can produce, which makes processing and potentially transmitting this type of data in real-time a challenge.

B. Data Reduction

Data reduction is a topic that has been studied for decades and for many years was mainly focused on creating new or improved data compression methods [38]. However, the emergence and scale of new sensor applications in edge environments has produced new needs for task specific data reduction solutions. As such, there have been new works focusing on data reductions for tasks such as fault detection [39], mobile health [40], and internet of things (IoT) applications [41] [42]. Improvements in vehicular sensor technology have brought about new levels of environmental perception for vehicles, but at the cost of producing large amounts of sensor data every second that can potentially affect the ability for real-time processing. As such, there have been several works focused on data reduction for vehicular sensor data in recent years, particularly relating to cooperative perception since the multi-source aspect of this problem creates an even greater need for reducing data. Autocast [43] reduces the amount of point cloud data generated by lidar sensors by intelligently selecting sections of points clouds to transmit to surrounding vehicles based on what is going to be more useful to the receiving vehicle. Coopernaut [44] also attempts to reduce the amount of point cloud data through the use of point transformer [45] as well as V2V-Net [46] which use variational image compression techniques [47] to reduce the size of encoded point cloud feature maps. However, all of these methods are based on vehicle-to-vehicle (V2V) communications which has difficulty with large scale implementations, as opposed to the vehicle-to-infrastructure (V2I) method of aggregating data at road side units (RSU), which is already beginning to see adoption and deployment [48]. EMP [49] does however utilize V2I links while reducing the amount of point cloud transmission by selective partitioning of the point cloud. While these methods do present novel solutions for reducing the point cloud data produced by lidar and similar sensors, they do not provide a general data reduction framework that can incorporate other types of sensor data such as RGB cameras.

C. Task Partitioning

Beyond simply reducing the amount of data each vehicle transmits, another way to improve the end-to-end latency in an edge fusion system is to incorporate computational task offloading. Since the amount of computing available on vehicles can vary greatly and in many cases can be quite limited, the

total computational inference time can be reduced by offloading certain tasks to the edge where more powerful computing exists. As the topics of collaborative perception and edge computing have emerged, so have many methods in task partitioning in vehicular edge computing [50]. Choosing the optimal selection of which tasks should be executed on the vehicle and which should be executed on the edges as well as potentially partitioning computationally expensive tasks between the vehicle and the edge can have a large impact on the overall quality of service of an edge fusion system. As such, many different techniques have been applied to this problem including game theory [51], convex optimization [52], load balancing [53], multi-armed bandits [54], dynamic programming [55], and reinforcement learning [56] [57].

D. Model Compression

As the paradigm of edge computing continues to evolve, there have been continual strides to discover new methods for reducing the inference time as well as the memory and energy consumption of the models that are being executed at the edge as well as the local devices. Many of these improvements fall under the umbrella of model compression [58] [59]. The most common type of model compression is model pruning, such as model slimming [60] or early exit [61], where a smaller version of the model with less parameters is utilized, which has less inference time at the cost of reduced accuracy. Other types of model compression that exist are parameter quantization [62], low-rank factorization [63], and knowledge distillation [64]. Some of these model compression methods have been combined with task partitioning to provide a suitable solution for task partition point selection for computational offloading in mobile edge networks. Edgent [65] combines task partitioning along with DNN early exit methods to select the optimal DNN size and partitioning points to adapt to the current channel bandwidth. In [66], a reinforcement learning (RL) model is created to determine the offloading decisions that utilizes both task partitioning as well as a number of model compression methods.

However, to the best of our knowledge, there is no work which combines simultaneous data reduction and model compression with task partitioning for creating computational offloading decisions in mobile edge environments. This work goes beyond the ones presented in this section by combining data reduction, model compression, and task partitioning simultaneously which we propose will lead to more optimal decisions in maximizing the utility of an edge computing architecture.

III. Simultaneous Data Reduction, Model Compression, and Task Partitioning

While collaborative perception can bring about new levels of safety and awareness for intelligent vehicles, accomplishing this sensor fusion task in connected vehicle environments can be difficult. Some factors that contribute to this difficulty are the highly variable and dynamic nature of vehicular communications and the limited amount of computational power available on vehicles. Additionally, the raw amount of sensor data produced by vehicular sensors can be quite large, even from just a single RGB camera. As such, a system to dynamically reduce the amount of data that each vehicle is transmitting based on network conditions is needed in order for the data sharing to take place in real time. However, data reduction alone may not always be enough to satisfy a given end-to-end latency requirement; many of the most accurate sensor fusion models also have high execution latency and may not be suitable for all situations. To this end, multiple different sensor fusion models that vary in complexity can be used. In this way, more lightweight models can be chosen in poor conditions where the total inference time must be reduced in order to satisfy the latency requirement. To provide even more optimal solutions, model compression is also utilized in order to increase the granularity of the decision space; each machine learning model that is used in this end-to-end sensor fusion task can have their parameters dynamically adjusted to decrease the size and corresponding latency of the chosen fusion model which creates more available options in the trade off between latency and accuracy. The final knob in this optimization problem is task partitioning; since in most cases the amount of computing available at the edge is more powerful than that of the vehicle's, partitioning the tasks that need to be executed through computational offloading can reduce the end-to-end latency of the system.

A. Method Overview

A visual overview of the REFO method we have created to accomplish this edge fusion task is shown in Fig. 1. In this method, it is assumed that each vehicle has an on-board unit (OBU) that can communicate with the RSU at the edge to fuse its sensor data with other nearby vehicles; both the RSU and the OBU are assumed to have computational capacity as well. In this system, how much computation occurs at the OBU versus at the RSU is dependent on the REFO action decision which will determine which tasks to execute at the RSU and which to execute at the OBU. Data reduction can be utilized in the form of choosing not to send certain sensor data modalities as well as choosing to compress the sensor data that has been selected for transmission. Multiple object detection methods as well as feature extractors and their associated fusion classifiers exist in the ensemble fusion model and these components can be compressed as chosen by the action decision framework.

Each component of REFO provides a trade-off space for this optimization problem that makes having to simultaneously select the offloading level, fusion models, and amount data/model compression a challenging task. Choosing to do less computational offloading and execute more tasks at the OBU increases total computational inference time but reduces the wireless transmission latency. Choosing a more lightweight object detector and feature extractor and/or utilizing model compression also reduces the computational inference time but at the cost of overall fusion accuracy and any data reduction will reduce the transmission latency at the cost of fusion accuracy. It is necessary to design a system that is able to intelligently balance these distinct trade-offs simultaneous to make the optimal action decisions for the current situation.

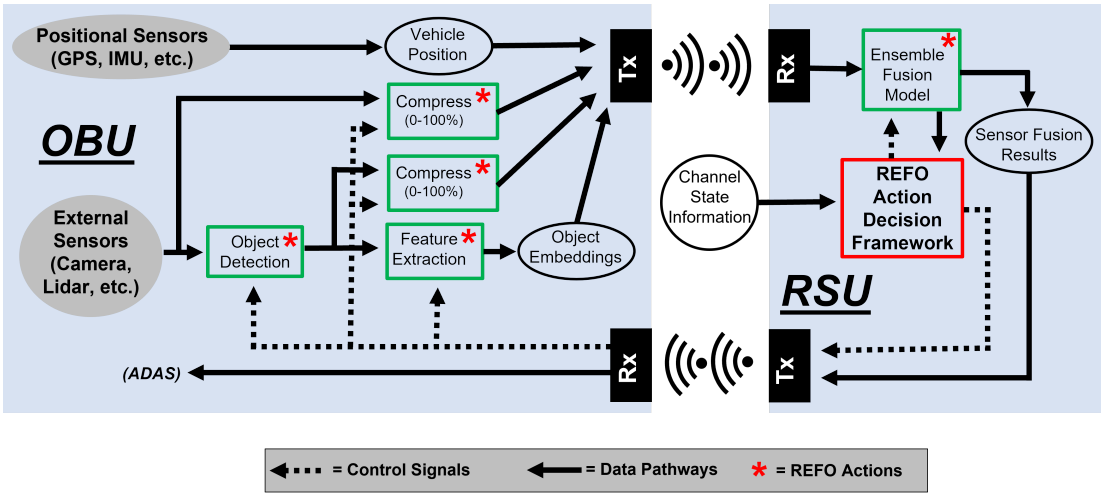


Fig. 1: REFO method overview showing the different data pathways and decision points involved in achieving multi-modal fusion in edge-based connected vehicle environments. The REFO action decision framework presented in Section IV-A is shown in the red box and generates the REFO action decision which acts as a control signal for the associated tasks marked with red asterisks. The ensemble fusion model presented in Section IV-B is shown in the green box at the RSU, but certain elements of this model are also available to execute at the OBU (also shown in green boxes) to allow for task partitioning.

TABLE I: Summary of Key Notations with Descriptions

Notation	Description
OBU	On-board Unit
RSU	Road-side unit
MEC	Mobile Edge Computing
L_{E2E}	End to end latency of the REFO process
L_{OBU}	OBU computation inference latency
L_{RSU}	RSU computation inference latency
L_{UL}	OBU to RSU transmission latency
L_{DL}	RSU to OBU transmission latency
s	Vector of states (contains state variables)
a	Vector of actions (contains action variables)
v	State/Action vector
S	Set of State/Actions vectors
r_{tp}	Wireless channel throughput (state variable)
c_{mec}	RSU MEC computing capacity (state variable)
n_{obj}	Number of objects detected in the previous frame (state variable)
n_{veh}	Number of participating vehicles in the previous frame (state variable)
d	Object Detector (action variable)
e	Feature Extractor (action variable)
o	Offloading level (action variable)
QoS	Quality of service
$EmAP$	Effective mean average precision

B. Problem Formulation

In this work, we aim to show that collaborative perception can be achieved in real-time even with limited channel throughput and computing power available. Our goal is to create a method to choose the set of actions given information about the current state of the environment so that accuracy is maximized while ensuring that service is maintained at all times. The choice of what actions (a) to take affects both the accuracy and latency with the latency value also being affected by the current state (s), so we will define two functions which produce the associated accuracy and latency values as follows:

$$f_{AP(t)}(a) = AP(t) \tag{1}$$

$$f_{L_{E2E}(t)}(s, a) = L_{E2E}(t) \tag{2}$$

To formalize what should be considered real-time in collaborative perception, successful service delivery (D) for REFO can be expressed as:

$$D(t) = \begin{cases} 1, & \text{if } L_{E2E}(t) < \tau. \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Essentially, this is defining an end-to-end latency threshold τ which will act as the QoS requirement. As such, the overall QoS for this system at some point in time $t = N$ is defined as:

$$\text{QoS} = \frac{1}{N} \sum_{t=1}^{N} D(t) \tag{4}$$

This metric represents the percentage of time that the actions chosen by the REFO action decision framework were able to achieve real-time collaborative perception.

Vehicular sensor fusion tasks are centered on creating detections of objects in the vehicle's surroundings or creating temporal/spatial associations between objects. As such, standard detection theory metrics such as precision and recall, which represent ratios of true positive (TP) detections/associations to false positives (FP) or false negatives (FN), will be used to assess the accuracy. Specifically, precision (P) is defined by:

$$\text{P} = \frac{TP}{TP + FP} \tag{5}$$

and recall (R) is defined by:

$$\text{R} = \frac{TP}{TP + FN} \tag{6}$$

However, these sensor fusion models are parameterized by an output threshold that lets the user adjust how sensitive the model should be; high output thresholds will produce higher precision but lower recall and vice versa. Average precision (AP), which is the area below the precision-recall curve, is

the most widely used metric for evaluating the performance of this type of sensor fusion model since all thresholds are represented in the precision-recall curve. More formally stated, AP is defined as:

$$AP = \sum_{i=1}^{K} (R_i - R_{i-1}) P_i \tag{7}$$

for all possible K output thresholds. For this REFO task, we will define mean average precision (mAP) in the context of this problem as the moving average of chosen actions associated sensor fusion AP:

$$mAP = \frac{1}{N} \sum_{t=1}^{N} AP(t) \tag{8}$$

To evaluate end-to-end performance, a constrained optimization problem is defined to express the conditions that are trying to be met. The goal in this work is to choose the actions that maximize the sensor fusion accuracy while ensuring QoS is being met and as such the optimization problem is defined as follows:

$$\max_{a} \quad mAP \tag{9a}$$

$$s.t. \quad QoS = 1 \tag{9b}$$

Considering this optimization problem, a new metric can be defined which we have termed effective mean average precision (EmAP) that simplifies this optimization using the fact that $QoS \in [0, 1]$ and thus can be used as an indicator function and multiplied with the objective function to produce an equivalent optimization:

$$EmAP = QoS \times mAP \tag{10}$$

This metric accurately reflects what needs to be optimized considering that the goal is to produce the most accurate perception possible while still ensuring the QoS requirement is met. The chosen actions cannot just maximize the fusion accuracy as the loss in QoS due to the high latency of the most accurate fusion models will prevent any acceptable EmAP values. Similarly, always choosing very lightweight machine learning models will lead to good QoS, but poor fusion accuracy which will carry into poor EmAP performance. To perform well, the action decision framework must be designed or trained to choose the best performing fusion model for the current conditions while also putting a very strong bias on ensuring that the latency threshold is not exceeded. As such, the objective of the REFO action decision framework will be to maximize the EmAP.

For any method that is used to generate action decisions, it is assumed that a selection choice will be provided for each time step. In certain situations, such as when the wireless link is completely lost, there may be no possible action decision that will allow the collaborative perception system to execute in the required latency; in these cases, the data point will be discarded in performance evaluation as to not punish models for exceeding the latency requirement when doing so is entirely unavoidable.

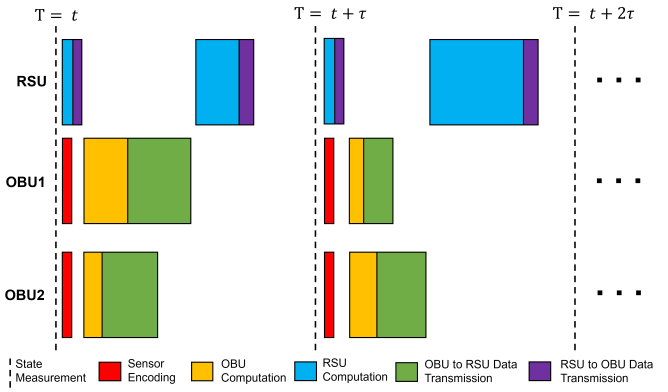


Fig. 2: Time series diagram showing the different components that make up the total end-to-end latency for a each time step in REFO.

C. Latency Model

We model the end-to-end latency (L_{E2E}) experienced by the following equation:

$$L_{E2E} = L_{OBU} + L_{UL} + L_{RSU} + L_{DL} \tag{11}$$

(All terms used in (11) are explained in Table I). This equation just shows all the different terms that sum up to create the total end-to-end latency. Both L_{UL} and L_{DL} use a simple channel delay model of transmission data size divided by channel throughput, but the values for L_{OBU} and L_{RSU} are dependent on which actions are selected and must be measured; the values presented in this section and remainder of the paper are specific to our hardware and choices of object detector, sensor fusion model, and compression algorithms.

A visual diagram of how these latency accumulate to complete this end-to-end fusion process can be seen in Fig. 2. For each time step, the process begins with each vehicle producing it's sensor data for that time step including any sensor encoding needed. At the same time at the RSU, the current channel state information is measured to determine the instantaneous channel throughput and the action decision process is executed immediately followed by transmitting the action selection to each of the nearby participating vehicles. By the time each vehicle receives the action decision, the sensor data is ready to be processed and any corresponding computation occurs. The output data for the corresponding offloading level chosen is then transmitted from each vehicle to the RSU; once all the participating vehicle's data has been received at the RSU, the remaining computation tasks for the sensor fusion can take place before finally broadcasting the fused results back to the vehicle's OBUs to be ingested by the ADAS.

The biggest challenge encountered in this REFO process is dealing with the uncertainty in latency prediction. While certain processes like sensor encoding and image compression have low variance in their execution latency, the latency experienced by wireless data transmission depends on the amount of data to be transmitted and amount of throughput available while the computation latency depends on how much input data is present and which models are chosen. The challenge is that at the start of each time step, only limited information

is available: the current channel throughput and the previous time step's information about the total number of participating vehicles and total objects detected. Knowing the current channel throughput is useful for determining how long it will take to broadcast the action decision to all vehicles in the area of the RSU, but the channel conditions will likely have changed by the time the vehicles transmit their chosen data to the RSU making it necessary to make predictions about the future channel throughput at each time step. Similarly, with only knowing information about the previous time step's data and not how many total vehicles will successfully transmit data to the RSU this time step and how many objects will be detected makes additional predictions necessary, though it is expected that these values will not shift dramatically between successive frames. This was a main factor in choosing a machine learning approach to this problem, since machine learning models have the ability to detect patterns and make predictions based on these sorts of uncertainties and should outperform dynamic programming or numerical optimization based methods.

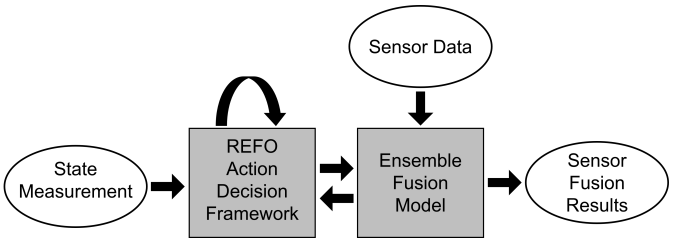


Fig. 3: Block diagram of REFO showing the flow of data that occurs in each time step. The REFO action decision framework stores the previous state information to use a feature, as represented by the return arrow at the top of the corresponding block.

IV. Methodology

In this section, we present our methodology for generating and executing action decisions in REFO. We will first introduce our selections for offloading levels before describing the action decision framework, which uses information about the current conditions to predict the optimal action decision for each time step. The later part of this section will discuss our proposed ensemble fusion model; the ensemble fusion model contains all perception and fusion abilities needed for collaborative perception and can also be dynamically compressed and partitioned. Each action selected by the action decision framework corresponds to a particular execution configuration of the ensemble fusion model. A block diagram showing the data flow for REFO is seen in Fig. 3. This figure shows a more simplified view of the data flow in REFO that doesn't include all possible data paths that occur between the RSU and OBU for the ensemble fusion model as shown in Fig. 1, while still displaying the process by which state measurements and sensor data are transformed into sensor fusion results.

For this work, we are only considering the use of RGB cameras, depth sensors, and positional trackers as our sensor modalities and the chosen sensor fusion task is object detection association. As such, our implementation of the action decision framework and ensemble fusion model will reflect these choices. However, any set of sensor modalities could be used as input if the corresponding sensor fusion task can be partitioned into subtasks for the associated ensemble fusion model. Both the action decision framework and the ensemble fusion model are integral components of our REFO method and will be discussed in detail for the remainder of this section. However, before beginning this discussion we must first define the different levels of offloading that we are utilizing.

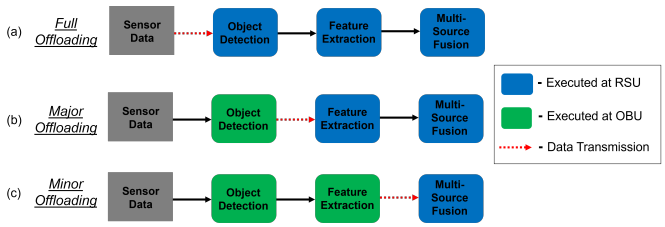


Fig. 4: Block diagram showing the three different offloading strategies that are considered in this work: (a) Full, (b) Major, and (c) Minor. These offloading levels determine how the different layers of the ensemble fusion model get partitioned between the OBU and RSU.

A. Task Partitioning - Offloading Selection

In this work, there are three levels of computational offloading that will be considered:

- Minor Offloading - All object detection and feature extraction is done at the OBU. Only the fusion classifier and REFO action decision framework are executed at the RSU.
- Major Offloading - All object detection is done at the OBU. Execution of feature extraction, fusion classifier, and REFO action decision framework occurs at the RSU.
- Full Offloading - All computation is done at the RSU.

The differences between these three offloading levels can also be seen in Fig. 4. Here we can see the affect this offloading decision has on the amount of data being transmitted. The amount of data transmitted over the wireless channel decreases from Full Offloading to Major Offloading and decreases even more significantly from Major Offloading to Minor Offloading. The trade-off here is that the amount of time that is spent on computational inference is inversely proportional to the amount of data transmitted since more of the computational tasks are occurring at the OBU, which is assumed to be much less powerful than that of the RSU. As such, the amount of sensor data generated by the vehicle and the wireless throughput will have the largest factors on which level of offloading should be chosen. The offloading level for each time step will be decided by the action decision framework, which will be described in the next subsection.

B. Action Decision Framework

In this section, we describe our REFO action decision framework, with an overview of the framework shown in Fig. 5. The action space (a) of the decision consists of three elements: object detector choice (d), feature extractor choice (e),

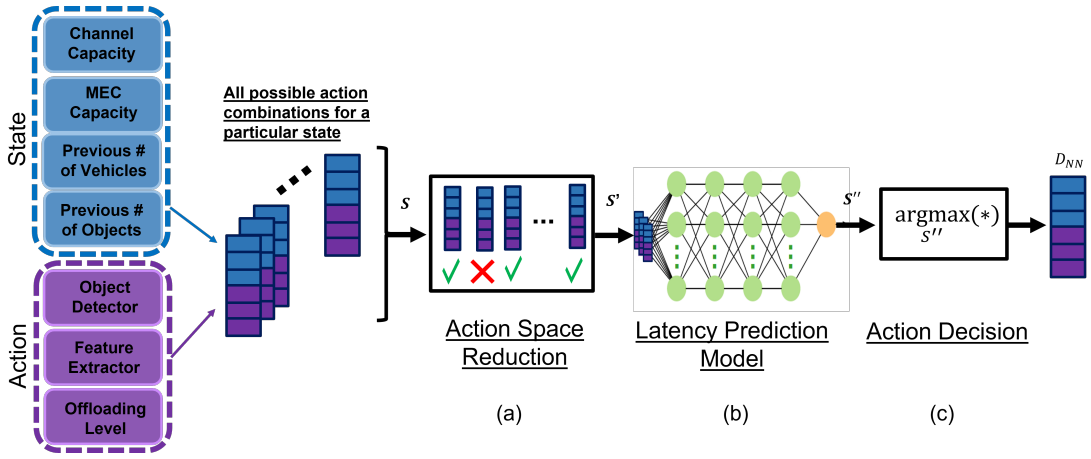


Fig. 5: Overview of the REFO action decision framework showing how an action is selected given the current state information; input state action combinations go though action space reduction and latency prediction before the final action decision is selected. The objective of this framework is to choose the action with the highest EmAP and its performance is largely dependent on the accuracy of the latency prediction model.

and offloading level (o). For each object detector and feature extractor the model as well as the compression level must be chosen and for offloading the offloading level as well as the data compression level must be chosen. Note that since none of the fusion models used require raw RGB images as input, compression does not need to be considered in the case of minor offloading. The state space (s) consists of four elements: current channel throughput (r_{tp}), RSU mobile edge computing (MEC) capacity (c_{mec}), the number of participating vehicles in the previous frame (n_{veh}), and the total number of objects detected by all vehicles in the previous frame (n_{obj}). For our REFO action decision framework, state-action pairs are created to form vectors (v):

$$v = < \underbrace{r_{tp},\ c_{mec},\ n_{veh},\ n_{obj}}_{\text{state:(s)}},\ \underbrace{d,\ e,\ o}_{\text{action:(a)}} > \tag{12}$$

which serve as the input and the output. For each time step, the input for our REFO action decision framework (S) is generated by combining all possible action space combinations with the current state:

$$S = \{v_1, v_2, ..., v_N\} \tag{13}$$

Each vector v_i in S has the same state space, but a unique action space. The first step in our REFO action decision framework is action space reduction, where the number of action state combinations are reduced by removing vectors from S that contain actions that have no possibility of being selected (Fig. 5(a)). There are three types of action space reduction that are conducted:

- Impossible Combinations - Remove any v_i that are not possible to execute. An example would be the action choice to send only color histograms as visual features when the chosen fusion model requires RGB images.
- Nonsense Combinations - Remove any v_i that do not make any sense to include. An example would be the action choice to send RGB images when the chosen fusion model only requires color histograms as its visual feature.
- Outliers - Remove any v_i that are likely to be outliers. For example, cases of sending raw RGB images with very poor channel conditions and medium to high vehicle density should be discarded.

The remaining state action combinations that remain after action space reduction (S') are used as input for the latency prediction model f_{NN} (Fig. 5(b)). A DNN is used as our latency prediction network which consists of four hidden layers with ReLU activation and batch normalization between each and a sigmoid output layer. It is trained for 5 epochs using a batch size of 512 and the ADAM [67] optimizer. The latency prediction DNN predicts for each $v_i{}'$ in S' whether it will meet the QoS requirement or not. The state action combinations that the latency prediction model predicts as having a latency less than the QoS threshold form the set S'':

$$S'' = f_{NN}(S') \tag{14}$$

The final step after applying the latency prediction model is to choose the action combination from S'' that maximizes the AP (Fig. 5(c)). Since EmAP is the product of QoS and mAP, the option v_* that will maximize EmAP from S'' should always contain the actions with the highest associated AP since it is assumed that S'' does not contain any options that would not meet the QoS latency requirement. More formally stated, since it is assumed QoS = 1 $\forall\ v \in S''$ the action decision produced by our REFO decision framework a_* is defined as:

$$a_* = \underset{a''}{\text{argmax}}[f_{AP}(a'')] \tag{15}$$

If there is ever a frame where the neural network predicts that no action will be able to execute within the chosen latency threshold, the lowest latency action decision is selected.

C. Ensemble Fusion Model

Once the REFO action decision framework has selected the actions for the current time step, this information can be forwarded to the vehicle's OBU. The ensemble fusion model is what embodies all of the trade-offs that are considered in this problem and where the data reduction, model compression, and task partitioning are being applied. As seen in Fig. 6(a),

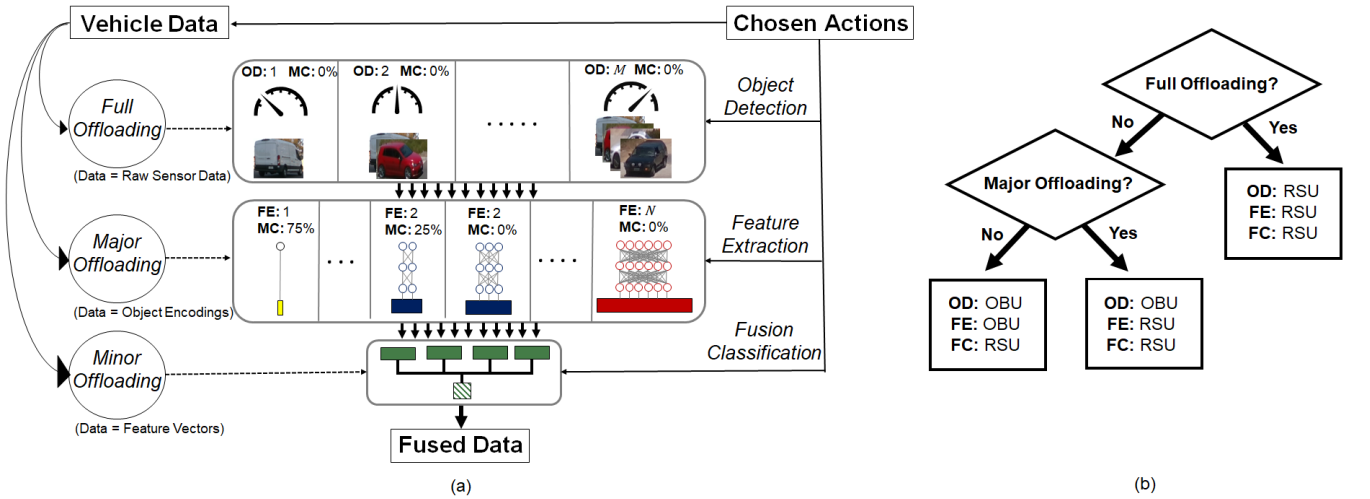


Fig. 6: The ensemble fusion model consists of an object detection (OD), feature extraction (FE), and fusion classification layer. (a) Shows an example ensemble fusion model which has N object detectors and M feature extractors, with each of these having the ability to be slimmed by a certain % of model compression (MC). (b) Shows a decision tree showing how the different layers will be partitioned based on the offloading level chosen by the action decision framework.

there are three layers to this ensemble fusion model: Object Detection, Feature Extraction, and Fusion Classification.

The entire ensemble fusion model is located at the RSU, but the object detection and feature extraction layers will also be available at each OBU. The object detection and feature extraction layers are constructed very similarly with each containing numerous models for accomplishing the corresponding task which vary in complexity with all or some models having additional compressed versions of themselves. Using these compressed models further increases the action decision space which helps provide more optimal solutions as the additional granularity allows further refinement in evaluating the trade-offs. The last layer is the fusion classification layer; here is where the final sensor fusion takes place and the corresponding classifier is chosen based on the output of the feature extraction.

The majority of the overhead in this model is contained in the first two layers of object detection and feature extraction and as such choosing the proper action for these two layers is important as the effects of these actions are cascaded through the entire model. For example, with the object detection layer we must simultaneously decide which object detector to use, what amount of model compression is needed if any, as well as determine whether it should be executed locally at the OBU or offloaded to execute at the RSU. By choosing the more heavyweight object detector and utilizing low or no model compression, more objects are going to be detected on average; this is good for overall fusion accuracy, but more objects detected means more data to be transmitted since more region of interest (ROI) images were produced. On the other hand, more lightweight object detectors and/or moderate to high amounts of model compression will be more likely to fail to detect some objects which will reduce the end to end latency significantly, both in computational inference time but also in transmission latency due to transmitting less ROI image data, at the cost of reduced fusion accuracy for the missed objects. There is a similar trade-off for the feature extraction layer,

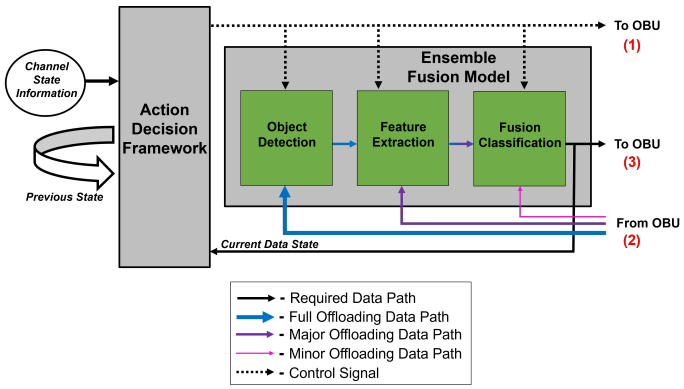


Fig. 7: Overview of the data pathways of the action decision framework and ensemble fusion model and their interaction with each other at the RSU. The order that data is received/transmitted is shown in red numbers on the right which corresponds to the process described in Fig. 2

but instead of more or less ROI images being produced, it is smaller or larger feature vectors that are used for the fusion classification.

A more detailed view of the RSU from Fig. 1 is shown in Fig. 7, displaying the interaction of the action decision framework and ensemble fusion model and the associated process that occurs at each time step. If major or minor offloading are chosen, this ensemble fusion model will be partitioned to execute some layers at the OBU which reduces the amount of data that needs to be transmitted over the wireless channel at the cost of increased computational inference time; as such, utilizing higher levels of offloading is more useful when the wireless channel conditions are good. Additionally, if full or major offloading is chosen, data compression can be applied to the transmitted data. In any case, using an ensemble fusion model provides a complete toolkit of knobs that should allow for real-time collaborative perception in nearly any condition given the presence of RSUs and participating vehicles.

V. Experimental Results

Now that we have presented our proposed REFO method, including the action decision framework and ensemble fusion model, we will show how they perform on real testing data. We first describe the research setup used to collect this data followed by a discussion on the action selections that were chosen for testing and process of evaluating their execution latency. Finally, the results of our framework on 5G and C-V2X network traces as well as a comparison of our model with related methods are presented to conclude this section.

A. Research Setup

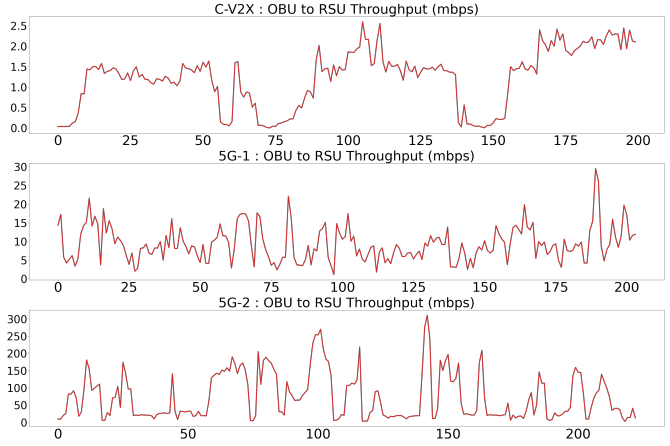


Fig. 8: Plots showing the throughput of the three different network traces that are being used as test scenarios.

The research setup used for this work is a hybrid between real-world wireless throughput traces and data recorded in a digital simulation environment. For the real-world wireless communications data, we use both 5G and C-V2X communication standards for vehicles to enable the sensor sharing [68]. We chose C-V2X as it has one of the highest throughput of all the current established vehicular communications standards, but even so the channel throughput can be quite limited. Data was recorded using C-V2X radios that we have set up on the UCSD campus; a research vehicle equipped with a Commsignia OBU kit [69] and a C-V2X OBU car antenna was driven along a road near the RSU driving 10 mph. The RSU is mounted to a light pole and consists of Commsignia RSU kit [70] and two 8m height C-V2X Urban Antennas. Both Commsignia kits are powered by Qualcomm C-V2X 9150 radio [71] running 3GPP Release 14 C-V2X standard [72]. The throughput that was achieved using this setup is 0-3 Mbps, which is low compared to emerging 5G networks or even current 4G LTE networks, but does provide an opportunity to explore collaborative perception in situations with more limited wireless networking (see our previous paper for more information about our C-V2X setup and data recording [73]).

For 5G networks traces, there are a number of open source datasets that exist that include 5G network data in various situations; [74] presents one such 5G dataset that includes 5G network traces from a moving vehicle and two example traces were chosen for use as test sets: one where 0-30 Mbps of

(a) Overhead view of CARLA street environment

(b) Positive Association

(c) Negative Association

Fig. 9: Example images from the virtual dataset used for testing showing: (a) An overview of the virtual environment, (b) an example of a positive object detection association and (c) an example of a negative object detection association.

throughput is achieved (5G-1) and another where the throughput varied between 0-300 Mbps (5G-2). These two networks traces along with the C-V2X network trace that we recorded provide three very different wireless network conditions to demonstrate the robustness of our proposed system. Visualisations of these three networks traces can be seen in Fig. 8.

Since we only had access to a single vehicle outfitted with external sensors and a OBU, we needed to obtain sensor data for the sensor fusion from another source. We have recorded RGB image data from moving vehicles in the CARLA [75] self driving simulator that can be used for this purpose [1]. While the wireless channel conditions of the physical environment are going to be different than that of the virtual simulation, we are using the images from CARLA and treating them as if vehicles in the real world were recording them.

For the proposed REFO method, the sensor fusion task is not set in stone and instead will be whatever is chosen by the end user. For this work, we are using object detection association [1] as the sensor fusion task that the performance evaluation will be based on. In this task, the goal is to determine all associations between objects detected by different vehicles to establish which objects were seen by multiple vehicles and which objects were only seen by a single vehicle. The final accuracy results, measured in AP, reveals how accurately the fusion classifier was able to correctly label these associations; this accuracy is affected by the choice of object detector, feature extractor, and amount of data/model reduction. An overview of the virtual environment that the images were recorded in as well as example associations are given in Fig. 9; for each example image pair, if the two images are of the same vehicle they labeled to be a positive association, as shown in Fig. 9(b), and if the image pair contains pictures of different vehicles then it is labeled to be a negative association, as shown in Fig. 9(c). The virtual data set that we are using for this work has object association labels for ground truth provided. For generating ground truth training data for the DNN within the action decision framework, we test every action combination using data from the vehicles in the virtual training data to provide the number of vehicles and objects detected per frame and sweep over all computing configurations (cc1 & cc2) and wireless throughput values (0-300 mbps) for the remaining state values. For each data point, if the particular action combination produces a latency $<\tau$ then it is labeled *True* (1) and if the

TABLE II: All possible actions that can be selected. :% values represent the amount of data/model compression that is utilized for that selection.

d $\rightarrow \{0,1,2,3\}$	0.YOLOv5x	1.YOLOv5l	2.YOLOv5m	3.YOLOv5s					
e $\rightarrow \{0,1,2,3,4,5,6,7,8\}$	0.ResNet-50:0%	1.ResNet-50:25%	2.ResNet-50:50%	3.ResNet-50:75%	4.MobileNet:0%	5.MobileNet:25%	6.MobileNet:50%	7.MobileNet:75%	8.Color Histogram
o $\rightarrow \{0,1,2,3,4,5,6,7,8\}$	0.Full:0%	1.Full:25%	2.Full:50%	3.Full:75%	4.Major:0%	5.Major:25%	6.Major:50%	7.Major:75%	8.Minor

latency value $\geq \tau$ then it is labeled *False* (0); accordingly multiple different versions of the DNN were trained, one for each of the chosen values of τ. Generating testing data is the same process but instead of using a sweep of throughput values, the data from the wireless traces seen in Fig 8 is used.

B. Action Selection

For this edge fusion task, we wanted to provide different trade-off options for each computation task to create a usable action space so that a machine learning model can learn what the best options are for different state combinations. YOLOv5 [76] was chosen as object detector since it is both lightweight and accurate; additionally there are multiple weights of the YOLOv5 model that have different amounts of parameters which correspond to different levels of object detection accuracy and execution latency; the 4 weights of YOLOv5 that were chosen are:

1) YOLOv5s
2) YOLOv5m
3) YOLOv5l
4) YOLOv5x

For features extractors, we wanted to provide similar accuracy/inference trade-offs and as such three different feature extractors were chosen:

1) ResNet-50 [16]
2) MobileNet [77]
3) Color Histogram Extraction

The rationale between choosing these three models was to choose a high (ResNet-50), medium (MobileNet), and low (Color Histograms) complexity feature extraction method and then use model compression to expand the high and medium models to fill in the gaps in the decision space. We choose color histograms as our low complexity model to provide very low computation inference time as well as data size but could have chosen lightweight multi-layer perceptron (MLP) DNN models as well as other image descriptors [78] [79] or image transforms [80] [81] if a larger decision space was needed.

The two deep CNN models can be compressed in a number of ways, one being model slimming where a number of the weights/connections within the neural network are removed to provide a fast executing model at the cost of fusion accuracy [60]. We consider compression for both compressing the models as well as compressing the sensor data being transmitted from the OBU to the RSU in the cases of Full or Minor offloading. The different compression levels we are considering for these two tasks are shown below:

1) 0%
2) 25%
3) 50%
4) 75%

The combinations of all object detectors, feature extractors, data/model compression levels along with the offloading levels described in Section IV-A form the set of actions $a = (d, e, o)$ that form the state-action pairs for the vectors v that form the input set S of the action decision network. All the individual actions that are available in our ensemble fusion model we are testing with can be seen in Table II.

C. Evaluating Execution Latency and Sensor Fusion Accuracy

In order to simulate the latency experienced by the system in terms of computational processing time, all feature extractors, object detectors, fusion classifiers and the latency prediction network were executed and averaged over 1000 inferences in order to get an accurate estimate for the predicted amount of computation time for each task. The computation tasks that are most affected by the increasing in computing power, specifically the GPU's floating point operations per second (FLOPS), are the deep convolutional neural networks (CNN) used for object detection and feature extraction. As such, the inference time varies significantly depending on the hardware used. For this work, we are considering three levels of computing represented by three different devices that we are executing the chosen models on: NVIDIA Jetson TX2 (1.33 TFLOPS), NVIDIA RTX 1080Ti (11.3 TFLOPS), and NVIDIA Tesla V100 (130 TFLOPS). With these three devices, two different computing configurations are defined in Table III named CC1 and CC2.

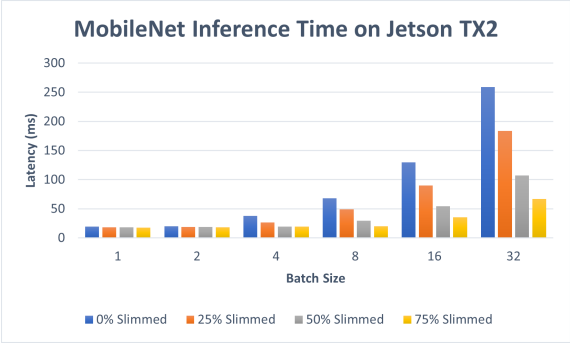
TABLE III: Defining the two different computing configurations that we are using for performance evaluation.

CC1:	OBU = Jetson TX2 , RSU = RTX 1080Ti
CC2:	OBU = RTX 1080Ti , RSU = Tesla V100

TABLE IV: YOLOv5 inference times on Nvidia Jetson TX2

	Yolov5s	Yolov5m	Yolov5l	Yolov5x
Latency (milliseconds)	76.31	166.44	300.88	534.14

The computation inference latency of these deep CNNs is also affected by the batch size, which in the case of feature extraction is how many ROI images are produced by the object detector. This latency trade-off is shown in Fig. 10 and Fig. 11, showing how the execution latency of ResNet-50 and MobileNet change depending on the hardware used and input batch size. The same trade-off is seen for object detection in Fig. 12 which use raw RGB images as input; since the Jetson TX2 is not being considered as RSU computing hardware, only a object detection batch size of 1 is needed since the OBU will at most have to process a single image per time step and these values are shown in Table IV. Many of the inference times for the Tesla V100 were available from NVIDIA [82]



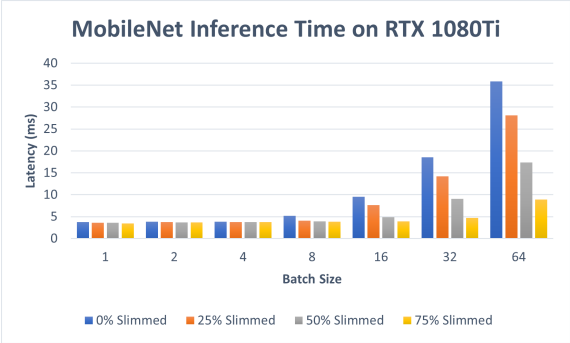
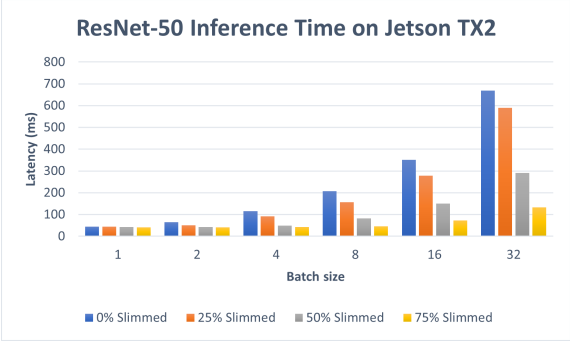


Fig. 10: MobileNet inference times on Nvidia RTX 1080Ti and Jetson TX2



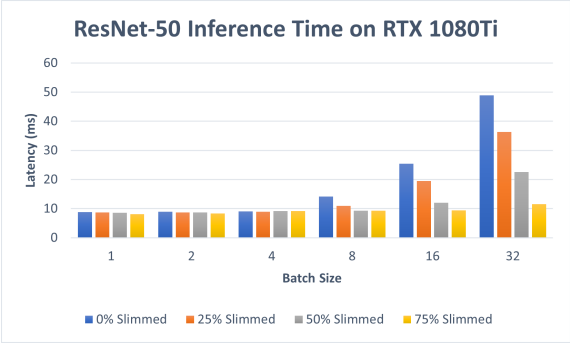


Fig. 11: ResNet-50 inference times on Nvidia RTX 1080Ti and Jetson TX2

or the creator of the particular model [83] and any missing values were interpolated.

The choice of hardware does not make as large of a difference on inference time for the more lightweight tasks like the latency prediction network. The inference times for different batch sizes

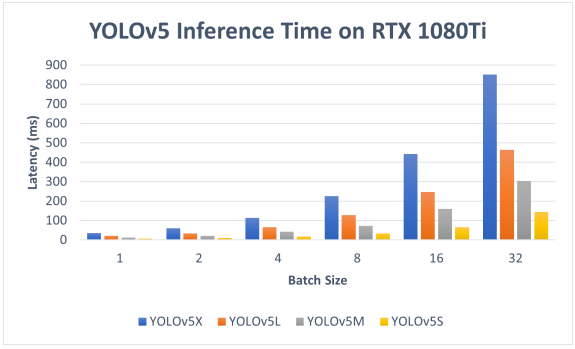


Fig. 12: YOLOv5 inference times on Nvidia RTX 1080Ti

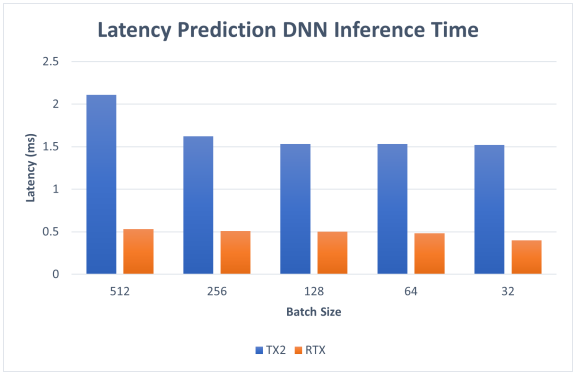


Fig. 13: Latency Prediction DNN inference times on Nvidia RTX 1080Ti and Jetson TX2

of the REFO action decision framework's latency prediction network are shown in Fig. 13. In the case of the latency prediction network, the batch size is equal to the size of S' which is the total number of state action combination that remain after action space reduction. Even on the NVIDIA Jetson TX2, our latency prediction DNN only takes about 2ms with the maximum possible batch size. Similarly, the fusion classification layer of the ensemble fusion model takes 2ms or less on all hardware.

TABLE V: The different levels of data and model compression that are being considered with the corresponding amount of mAP reduction that is experienced by that particular combination of feature extractor and compression.

	ResNet-50	MobileNet
Data Compression: 25%	1.19%	1.23%
Data Compression: 50%	1.90%	1.91%
Data Compression: 75%	2.35%	2.41%
Model Compression: 25%	0.68%	0.23%
Model Compression: 50%	1.14%	0.46%
Model Compression: 75%	1.82%	1.16%

Each combination of object detector, feature extractor, and compression level produces different accuracy results for this fusion task; using more heavyweight object detectors and feature extractors with little to no compression produces better results, but the effects as more lightweight models and more compression are chosen is not uniform. This result can be seen in Table V, which lists the percentage in AP reduction experienced by compression when averaged over all testing configurations. While this table shows how model

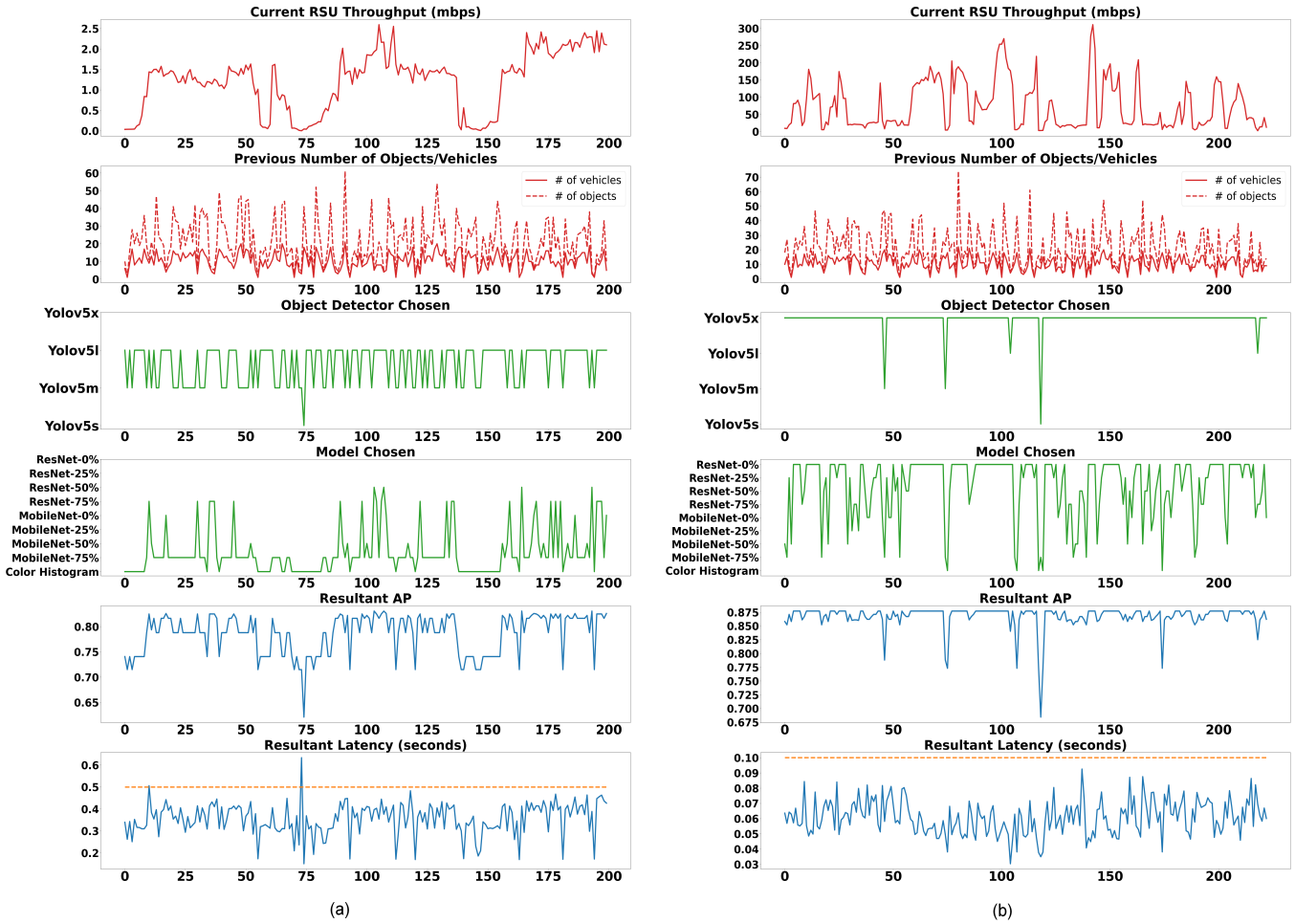


Fig. 14: Time series plots for two different combinations of computing configurations, network traces, and latency thresholds: [CC1, C-V2X, 500ms] (a) and [CC2, 5G-2, 100ms] (b). The red lines show the state space for each time step while the green show the action decisions chosen by REFO and blue shows the resultant latency and accuracy. The horizontal orange line in the latency plot represents the latency threshold (τ) for that scenario.

compression has a less detrimental effect on fusion accuracy than compression, their benefits are quite distinct with the main benefit of data compression being in reducing transmission latency while model compression is reducing computational latency.

The more lightweight object detectors actually produce better fusion accuracy due to fewer objects being detected by the object detector, but we wanted to make sure that the system is not rewarded for potentially missing important detections by always using a more lightweight object detector, so the accuracy results were weighted according to the number of images each object detector detected. Since YOLOv5x is the most heavyweight object detector being considered in this work, we assume this object detector produces a true set of detections and will penalize the more lightweight YOLO models according to how many fewer objects they detected.

D. Performance Evaluation

For the purpose of performance evaluation of our proposed REFO method, we examine 2 different scenarios of computing/network/latency threshold combinations to get some perspective on how our REFO action decision framework performs in these situations. The two configurations are:

1) Computing = CC1, Network = C-V2X, τ = 500ms
2) Computing = CC2, Network = 5G-2, τ = 100ms

In Fig. 14 we have presented two time series plots on these two configurations which show the state space, actions selected, and resultant latency and fusion mAP that were achieved for each time step. For the latency plots, a horizontal line is also included to show the latency threshold. As seen in this figure, our model is able to successfully stay below the latency threshold while actively adapting to the changing state conditions; the ability to be just below the threshold for most time steps shows that the system is optimizing the action selection process well, but of course at the cost of the occasional missed prediction where the latency threshold was exceeded. Each subplot for the pair of plots is temporally aligned so that the vector v chosen by the action decision framework can be seen for each time step. For example, at $t = 0$ in the [CC2, 5G-2, 100ms] plot in Fig. 14(b) the chosen vector is $v_* = <9.426mbps, CC2, 10, 18, YOLOV5x, MobileNet - 75\%, Minor>$.

TABLE VI: The different static baselines used for performance evaluation.

	Object Detector	Feature Extractor	Data Compression	Model Compression	Offloading Level
Low latency action decision - a_L	YOLOv5s	Color Histogram	0%	0%	Minor
Medium-Low latency action decision - a_{ML}	YOLOv5m	MobileNet	0%	50%	Minor
Medium-High latency action decision - a_{MH}	YOLOv5l	MobileNet	50%	0%	Major
High latency action decision - a_H	YOLOv5x	ResNet-50	0%	0%	Full

TABLE VII: Performance comparison showing the EmAP results of different methods tested on various configurations of wireless network conditions, latency thresholds, and computing configurations.

	Computing Config. 1 (CC1)		Computing Config. 2 (CC2)			
Network = C-V2X	τ=500ms	τ=250ms	τ=500ms	τ=250ms	τ=100ms	τ=50ms
a_L	0.621	0.621	0.621	0.621	0.621	0.621
a_{ML}	0.476	0.089	0.557	0.397	0.144	0.028
a_{MH}	0.102	0.000	0.364	0.119	0.028	0.004
a_H	0.000	0.000	0.000	0.000	0.000	0.000
Greedy	0.579	0.564	0.645	0.579	0.554	0.539
RF [84]	0.657	0.685	0.727	0.705	0.677	0.704
GBT [85]	0.604	0.678	0.657	0.661	0.688	0.693
ALTO [54]	0.069	0.010	0.139	0.041	0.011	0.007
Edgent [65]	0.562	0.670	0.580	0.583	0.618	0.724
REFO-ADF*	**0.767**	**0.712**	**0.816**	**0.800**	**0.764**	**0.757**

(a) EmAP results for the C-V2X network trace.

	Computing Config. 1 (CC1)		Computing Config. 2 (CC2)			
Network = 5G-1	τ=500ms	τ=250ms	τ=500ms	τ=250ms	τ=100ms	τ=50ms
a_L	0.621	0.621	0.621	0.621	0.621	0.621
a_{ML}	0.732	0.639	0.740	0.743	0.698	0.421
a_{MH}	0.632	0.000	0.755	0.686	0.342	0.050
a_H	0.012	0.000	0.020	0.000	0.000	0.000
Greedy	0.680	0.690	0.753	0.753	0.642	0.251
RF [84]	0.578	0.563	0.847	0.771	0.693	0.652
GBT [85]	0.554	0.558	0.848	0.739	0.580	0.675
ALTO [54]	0.422	0.152	0.467	0.375	0.213	0.074
Edgent [65]	0.512	0.463	0.831	0.673	0.498	0.563
REFO-ADF*	**0.816**	**0.772**	**0.863**	**0.856**	**0.846**	**0.792**

(b) EmAP results for the 5G-1 network trace.

	Computing Config. 1 (CC1)		Computing Config. 2 (CC2)			
Network = 5G-2	τ=500ms	τ=250ms	τ=500ms	τ=250ms	τ=100ms	τ=50ms
a_L	0.621	0.621	0.621	0.621	0.621	0.621
a_{ML}	0.759	0.726	0.759	0.762	0.742	0.712
a_{MH}	0.756	0.000	0.787	0.780	0.647	0.248
a_H	0.235	0.049	0.444	0.214	0.034	0.000
Greedy	0.763	0.423	0.811	0.649	0.393	0.108
RF [84]	0.632	0.552	0.867	0.852	0.794	0.663
GBT [85]	0.621	0.519	0.862	0.852	0.754	0.659
ALTO [54]	0.616	0.349	0.641	0.593	0.377	0.230
Edgent [65]	0.476	0.428	0.861	0.854	0.707	0.541
REFO-ADF*	**0.828**	**0.780**	**0.869**	**0.868**	**0.857**	**0.839**

(c) EmAP results for the 5G-2 network trace.

*Our proposed model

One thing that stood out about our model's action decision is that it selects minor offloading for every single time step in both of these situations, which is why a plot with this value is not represented in Fig. 14. Since the difference in data is so large between major and minor offloading (transmitting RoI images versus transmitting feature vectors), there are only a small percentage of cases where full or major offloading actually improve the end-to-end latency of the system and as such our latency prediction model has learned to almost always choose minor offloading. For major and full offloading to be a more effective tool in REFO, there would need to be a larger difference in the computing power between that of the OBU and RSU, as well as higher levels of wireless throughput than what we are considering in this work. However, there is a large amount of adaptation achieved by our model in terms of object detector and feature extractor choices which allow the model to perform well in both scenarios even though the network type, computing configuration, and latency threshold are different.

To further validate the performance of our method, we implemented a number of alternate methods to compare their performance with that of our model; the full results of these comparisons can be seen in TableVII and will be discussed in the remainder of this section. While we presented detailed

results from our method in two different scenarios, we wanted to ensure that our model would continue to perform well in all scenarios. As such, we have tested the EmAP of our model over all permutations of computing configurations [CC1, CC2], latency thresholds [500/250/100/50ms], and networks conditions [C-V2X, 5G-1, 5G-2]. This produces 24 different scenarios, but we excluded the scenario of CC1 and 50/100ms thresholds since this computing configuration was too weak for these latency thresholds. We will be using the same ensemble fusion model for all methods as well as training/testing data sets for all models that require training. Since it is just the action decision process that we are comparing, we will refer to our method as REFO-ADF to show that we are comparing the REFO performance measured im EmAP our our action decision framework (ADF) compared to other methods when used in place of our ADF.

Four static models were created to act as performance baselines and are defined in Table VI. The lowest latency action decision (a_L) is the main performance baseline for determining how successful a model is. This is because the EmAP performance of a_L will never change by definition of the EmAP metric; if there is a situation where even a_L will exceed the latency threshold then no action exists which can satisfy the QoS for this time step and the data point is discarded for EmAP calculations. As such, the EmAP for a_L will be .621 for all cases. Since this level of performance can be achieved by a static model, the expectation is that models that can successfully adapt to the changing conditions can provide more optimal levels of performance. The other rows at the top of each table show the other three static action selection strategies, whose performance varies greatly depending on the situation. a_H however performs poorly in most cases; this is due to the fact that full offloading is used in this strategy which does not work well for situations with low throughput since all sensor data will be sent to the RSU with no computation occurring at the OBU except data compression if selected. a_H utilizes full offloading, no compression, and uses an uncompressed ResNet-50 as the feature extractor so the only situations where this is possible is when the throughput is high ($>$ 50 Mbps) such as in 5G-2 (Table VIIc). Additionally, a greedy decision method was created that uses the channel conditions to make a decision about which of the four static method should be used at each time step.

Two ensemble learning methods are also employed with the Random Forest [84] and Gradient Boosted Tree [85]. These decision tree based machine learning methods provide a non-neural network comparison for our model that does not consume a massive amount of training time or computing resources; they do not perform as well as our proposed neural network based model, but they do provide competitive levels of performance across the majority of test configurations. In addition the these two off-the-shelf methods, we also implemented two methods from related works to examine how well these methods would preform when tested in our created testing scenarios. One of these is an adaptive task offloading method (ALTO) [54] and the other is an AI-enabled edge task partitioning and model compression algorithm (Edgent) [65]. As seen in the table, our proposed REFO methodology is able to outperform all comparison models in terms of the EmAP metric. The ALTO method is based on the multi-armed bandit algorithm which proved to be effective for the task partitioning and offloading decisions in vehicular edge networks, but does not perform so well in choosing what action to select in REFO; part of the reason for this is because the actions become more likely to be selected the longer they go without being selected in the multi-armed bandit, which is not a good model trait for this particular problem. Edgent performs well in the easy cases of high throughput, computing and latency threshold, but falls behind the machine learning based methods in other cases.

We provide a visual representation of the results from Table VII in Fig. 15. This figure shows the results of each method averaged over the 3 different network conditions providing a visual summary of the testing results over all 18 test cases. As is consistent with Table VII, the REFO-ADF line is the highest on the graph with a sizable margin over the next best method in all 6 computing/threshold configurations.

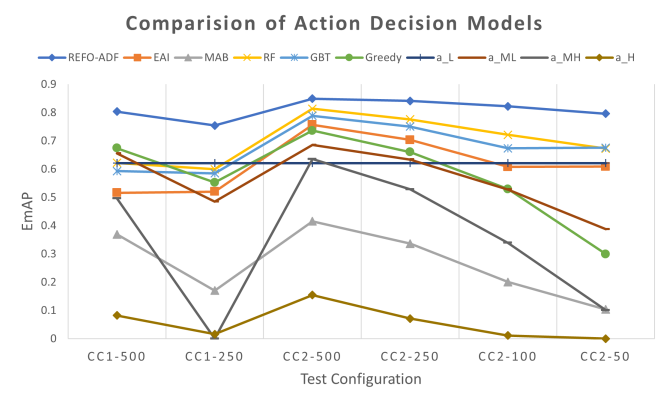


Fig. 15: Results of all of the different action decision models averaged over the three network traces.

VI. Conclusion and Future Work

In this work we proposed REFO, a method for achieving multi-source sensor fusion for collaborative perception in a connected vehicle environment even with highly varying wireless channel and limited computation capacities. We implemented a REFO action decision framework to determine the best action decision given information about the current state. We tested our method on both 5G and C-V2X network traces and show that our REFO action decision framework is able to outperform the best comparison methods by 9.6% on average.

This work is an important first step in exploring collaborative perception in connected vehicle environments that we plan on continuing as part of our smart transportation research[1]. While the fusion models explored in this work only use positional and RGB image data, we plan to incorporate other models that can utilize additional sensor modalities such lidar and/or radar data or even telematics/telemetry data from the vehicle and explore the challenges of heterogeneous sensor fusion. Additionally, we want to improve the intelligence of the REFO action decision

[1]http://cwc.ucsd.edu/research/cellular-vehicle-everything-c-v2x

framework to incorporate environmental context information such as the weather or driver state information that can further improve the performance of the system as well.

REFERENCES

[1] S. Thornton, B. Flowers, and S. Dey, "Multi-source feature fusion for object detection association in connected vehicle environments," *IEEE Access*, vol. 10, pp. 131 841–131 854, 2022.

[2] Y. Han, H. Zhang, H. Li, Y. Jin, C. Lang, and Y. Li, "Collaborative perception in autonomous driving: Methods, datasets and challenges," *arXiv preprint arXiv:2301.06262*, 2023.

[3] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 898–916, 2011.

[4] D. Gavrila and V. Philomin, "Real-time object detection for "smart" vehicles," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 1, 1999, pp. 87–93 vol.1.

[5] C. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection," in *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, 1998, pp. 555–562.

[6] O. Chapelle, P. Haffner, and V. Vapnik, "Support vector machines for histogram-based image classification," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1055–1064, 1999.

[7] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 3360–3367.

[8] K. E. A. van de Sande, J. R. R. Uijlings, T. Gevers, and A. W. M. Smeulders, "Segmentation as selective search for object recognition," in *2011 International Conference on Computer Vision*, 2011, pp. 1879–1886.

[9] Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, L. Cao, and T. Huang, "Large-scale image classification: Fast feature extraction and svm training," in *CVPR 2011*, 2011, pp. 1689–1696.

[10] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester, "Cascade object detection with deformable part models," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 2241–2248.

[11] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[12] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, 2006, pp. 2169–2178.

[13] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 1794–1801.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[17] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural Computation*, vol. 29, no. 9, pp. 2352–2449, 2017.

[18] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[19] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3523–3542, 2022.

[20] A. Saberironaghi, J. Ren, and M. El-Gindy, "Defect detection methods for industrial products using deep learning techniques: A review," *Algorithms*, vol. 16, no. 2, 2023. [Online]. Available: https://www.mdpi.com/1999-4893/16/2/95

[21] N. Zeng, P. Wu, Z. Wang, H. Li, W. Liu, and X. Liu, "A small-sized object detection oriented multi-scale feature fusion approach with application to defect detection," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–14, 2022.

[22] N. Davari, G. Akbarizadeh, and E. Mashhour, "Corona detection and power equipment classification based on googlenet-alexnet: An accurate and intelligent defect detection model based on deep learning for power distribution lines," *IEEE Transactions on Power Delivery*, vol. 37, no. 4, pp. 2766–2774, 2022.

[23] Q. Li, W. Cai, X. Wang, Y. Zhou, D. D. Feng, and M. Chen, "Medical image classification with convolutional neural network," in *2014 13th International Conference on Control Automation Robotics Vision (ICARCV)*, 2014, pp. 844–848.

[24] J. Ker, L. Wang, J. Rao, and T. Lim, "Deep learning applications in medical image analysis," *IEEE Access*, vol. 6, pp. 9375–9389, 2018.

[25] Z. Li, M. Dong, S. Wen, X. Hu, P. Zhou, and Z. Zeng, "Clu-cnns: Object detection for medical images," *Neurocomputing*, vol. 350, pp. 53–59, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231219305521

[26] M. Pritt and G. Chern, "Satellite image classification with deep learning," in *2017 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, 2017, pp. 1–7.

[27] G. A. Nastaran Aghaei and A. Kosarian, "Osdes_net: oil spill detection based on efficient_shuffle network using synthetic aperture radar imagery," *Geocarto International*, vol. 37, no. 26, pp. 13 539–13 560, 2022. [Online]. Available: https://doi.org/10.1080/10106049.2022.2082545

[28] F. Mahmoudi Ghara, S. B. Shokouhi, and G. Akbarizadeh, "A new technique for segmentation of the oil spills from synthetic-aperture radar images using convolutional neural network," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 15, pp. 8834–8844, 2022.

[29] X. Ding, A. Carlsen, J. Schaefer, M. Marple, D. Klotzbücher, W. Poiger, B. Brust, and F. Trompeter, "Theory and practice: A two-channel automotive radar for three-dimensional object detection," in *2015 European Radar Conference (EuRAD)*, 2015, pp. 265–268.

[30] M. Chiani, A. Giorgetti, and E. Paolini, "Sensor radar for object tracking," *Proceedings of the IEEE*, vol. 106, no. 6, pp. 1022–1041, 2018.

[31] R. Nabati and H. Qi, "Rrpn: Radar region proposal network for object detection in autonomous vehicles," in *2019 IEEE International Conference on Image Processing (ICIP)*, 2019, pp. 3093–3097.

[32] X. Dong, P. Wang, P. Zhang, and L. Liu, "Probabilistic oriented object detection in automotive radar," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[33] M. Meyer, G. Kuschk, and S. Tomforde, "Graph convolutional networks for 3d object detection on radar data," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 3060–3069.

[34] Y. Wang, Z. Jiang, Y. Li, J.-N. Hwang, G. Xing, and H. Liu, "Rodnet: A real-time radar object detection network cross-supervised by camera-radar fused object 3d localization," *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no. 4, pp. 954–967, 2021.

[35] Y. Wu, Y. Wang, S. Zhang, and H. Ogai, "Deep 3d object detection networks using lidar data: A review," *IEEE Sensors Journal*, vol. 21, no. 2, pp. 1152–1171, 2021.

[36] G. Zamanakos, L. Tsochatzidis, A. Amanatiadis, and I. Pratikakis, "A comprehensive survey of lidar-based 3d object detection methods with deep learning for autonomous driving," *Computers Graphics*, vol. 99, pp. 153–181, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0097849321001321

[37] S. Y. Alaba and J. E. Ball, "A survey on deep-learning-based lidar 3d object detection for autonomous driving," *Sensors*, vol. 22, no. 24, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/24/9577

[38] D. A. Lelewer and D. S. Hirschberg, "Data compression," *ACM Comput. Surv.*, vol. 19, no. 3, p. 261–296, sep 1987. [Online]. Available: https://doi.org/10.1145/45072.45074

[39] K. Dhibi, R. Fezai, M. Mansouri, M. Trabelsi, K. Bouzrara, H. Nounou, and M. Nounou, "A hybrid fault detection and diagnosis of grid-tied pv systems: Enhanced random forest classifier using data reduction and interval-valued representation," *IEEE Access*, vol. 9, pp. 64 267–64 277, 2021.

[40] Y. Cao, H. Zhang, Y.-B. Choi, H. Wang, and S. Xiao, "Hybrid deep learning model assisted data compression and classification for efficient data delivery in mobile health applications," *IEEE Access*, vol. 8, pp. 94 757–94 766, 2020.

[41] D. F. Mahmoud, S. M. Moussa, and N. L. Badr, "The spatiotemporal data reduction (stdr): An adaptive iot-based data reduction approach,"

in *2021 Tenth International Conference on Intelligent Computing and Information Systems (ICICIS)*, 2021, pp. 355–360.

[42] M. Younan, M. Elhoseny, A. E.-M. A. Ali, and E. H. Houssein, "Data reduction model for balancing indexing and securing resources in the internet-of-things applications," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5953–5972, 2021.

[43] H. Qiu, P.-H. Huang, N. Asavisanu, X. Liu, K. Psounis, and R. Govindan, "Autocast: Scalable infrastructure-less cooperative perception for distributed collaborative driving," in *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services*, ser. MobiSys '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 128–141. [Online]. Available: https://doi.org/10.1145/3498361.3538925

[44] J. Cui, H. Qiu, D. Chen, P. Stone, and Y. Zhu, "Coopernaut: End-to-end driving with cooperative perception for networked vehicles," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 17 231–17 241.

[45] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 16 259–16 268.

[46] T.-H. Wang, S. Manivasagam, M. Liang, B. Yang, W. Zeng, and R. Urtasun, "V2vnet: Vehicle-to-vehicle communication for joint perception and prediction," in *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II*. Berlin, Heidelberg: Springer-Verlag, 2020, p. 605–621. [Online]. Available: https://doi.org/10.1007/978-3-030-58536-5_36

[47] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," 2018.

[48] U. D. of Transportation. Successful communication between roadside unit and standard traffic controller. [Online]. Available: https://www.its.dot.gov/pilots/roadside_unit.htm

[49] X. Zhang, A. Zhang, J. Sun, X. Zhu, Y. E. Guo, F. Qian, and Z. M. Mao, "Emp: Edge-assisted multi-vehicle perception," in *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 545–558. [Online]. Available: https://doi.org/10.1145/3447993.3483242

[50] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, "Vehicular edge computing and networking: A survey," 2019.

[51] Y. Liu, S. Wang, J. Huang, and F. Yang, "A computation offloading algorithm based on game theory for vehicular edge networks," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–6.

[52] C. Yang, Y. Liu, X. Chen, W. Zhong, and S. Xie, "Efficient mobility-aware task offloading for vehicular edge computing networks," *IEEE Access*, vol. 7, pp. 26 652–26 664, 2019.

[53] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task offloading in vehicular edge computing networks: A load-balancing solution," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2092–2104, 2020.

[54] Y. Sun, X. Guo, J. Song, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, "Adaptive learning-based task offloading for vehicular edge computing systems," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3061–3074, 2019.

[55] Y.-J. Ku, S. Baidya, and S. Dey, "Adaptive computation partitioning and offloading in real-time sustainable vehicular edge computing," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 12, pp. 13 221–13 237, 2021.

[56] G. Raja, A. Ganapathisubramaniyan, S. Anbalagan, S. B. M. Baskaran, K. Raja, and A. K. Bashir, "Intelligent reward-based data offloading in next-generation vehicular networks," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 3747–3758, 2020.

[57] 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.

[58] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, "Model compression and hardware acceleration for neural networks: A comprehensive survey," *Proceedings of the IEEE*, vol. 108, no. 4, pp. 485–532, 2020.

[59] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "Model compression and acceleration for deep neural networks: The principles, progress, and challenges," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 126–136, 2018.

[60] J. Yu, L. Yang, N. Xu, J. Yang, and T. Huang, "Slimmable neural networks," 2018.

[61] S. Teerapittayanon, B. McDanel, and H. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, 2016, pp. 2464–2469.

[62] Y. Choi, M. El-Khamy, and J. Lee, "Towards the limit of network quantization," 2017.

[63] C. Tai, T. Xiao, Y. Zhang, X. Wang, and W. E, "Convolutional neural networks with low-rank regularization," 2016.

[64] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015.

[65] 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.

[66] L. Wang, L. Xiang, J. Xu, J. Chen, X. Zhao, D. Yao, X. Wang, and B. Li, "Context-aware deep model compression for edge cloud computing," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, 2020, pp. 787–797.

[67] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[68] S. Chen, J. Hu, Y. Shi, L. Zhao, and W. Li, "A vision of c-v2x: Technologies, field testing, and challenges with chinese development," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 3872–3881, 2020.

[69] Commsignia Inc., *Powerful V2X Onboard Unit*, (Accessed: Feb. 18, 2022). [Online]. Available: https://www.commsignia.com/products/obu/

[70] Commsignia Inc. , *High performance V2X enabled roadside unit with edge computing*, (Accessed: Feb. 18, 2022). [Online]. Available: https://www.commsignia.com/products/rsu/

[71] Qualcomm Inc., *C-V2X 9150*, (Accessed: Feb. 18, 2022). [Online]. Available: https://www.qualcomm.com/products/qualcomm-c-v2x-9150

[72] The 3rd Generation Partnership Project, *3GPP Release 14*, (Accessed: Feb. 18, 2022). [Online]. Available: https://www.3gpp.org/release-14

[73] Y.-J. Ku, B. Flowers, S. Thornton, S. Baidya, and S. Dey, "Adaptive c-v2x sidelink communications for vehicular applications beyond safety messages," *2022 IEEE 95th Vehicular Technology Conference (VTC-Spring)*, pp. 1–7, Jun 2022.

[74] D. Raca, D. Leahy, C. J. Sreenan, and J. J. Quinlan, "Beyond throughput, the next generation: A 5g dataset with channel and context metrics," in *Proceedings of the 11th ACM Multimedia Systems Conference*, ser. MMSys '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 303–308. [Online]. Available: https://doi.org/10.1145/3339825.3394938

[75] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[76] e. a. Glenn Jocher, "ultralytics/yolov5: v7.0 - yolov5 sota realtime instance segmentation," Nov. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.7347926

[77] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for mobilenetv3," 2019. [Online]. Available: https://arxiv.org/abs/1905.02244

[78] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Compt. Vis. (IJCV)*, vol. 60, no. 2, pp. 91–110, 2004.

[79] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417.

[80] G. Akbarizadeh, "A new statistical-based kurtosis wavelet energy feature for texture recognition of sar images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 50, no. 11, pp. 4358–4368, 2012.

[81] G. Akbarizadeh and G. A. Rezai-Rad, "A new cumulant-based active contour model with wavelet energy for segmentation of sar images," in *2010 6th Iranian Conference on Machine Vision and Image Processing*, 2010, pp. 1–4.

[82] NVIDIA. Resnet-50 v1.5 for mxnet. [Online]. Available: https://github.com/NVIDIA/DeepLearningExamples/tree/master/MxNet/Classification/RN50v1.5#inference-performance-benchmark

[83] Ultralytics. yolov5. [Online]. Available: https://github.com/ultralytics/yolov5

[84] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[85] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf