



(RESEARCH ARTICLE)



INTELLIGENT BANDWIDTH ALLOCATION FOR COOPERATIVE PERCEPTION IN EDGE-IOT VEHICULAR SYSTEMS: LEARNING WHAT TO SHARE UNDER COMMUNICATION BUDGETS

Akinwale Obafemi *

School of Computing, Engineering and the Built Environment, Edinburgh Napier University, Edinburgh, Scotland, UK.

* Corresponding Author; Email: akinwale.obafemi@napier.ac.uk

ORCID Details

Akinwale Obafemi: <https://orcid.org/0009-0003-6875-1900>

World Journal of Advanced Research and Reviews, 2026, 31(03), 1643–1658

Publication history: Received on 15 August 2026; revised on 20 September 2026; accepted on 22 September 2026

Article DOI: <https://doi.org/10.30574/wjarr.2026.31.3.2478>

Copyright © 2026 Author(s) retain the copyright of this article. This article is published under the terms of the Creative Commons Attribution License 4.0

ABSTRACT

Cooperative perception lets connected vehicles share intermediate neural features, but vehicle-to-everything (V2X) links carry far less than a modern detector produces, and existing work reports transmitted bytes without their energy cost. Our objective is to decide what to share under a single hard ego-side communication budget. As methodology, we predict, from metadata available before any feature is transmitted, how many objects a candidate block would add to what the ego alone detects, using an ensemble of gradient-boosted trees and a multilayer perceptron, and allocate the budget across helpers, spatial blocks and numerical fidelity (fp16/int8/int4) with a greedy knapsack. Energy and latency are modelled without hardware from measured 5G modem power, a per-operation model calibrated on measured Jetson TX2 data, and replayed 5G driving traces. On the OPV2V benchmark the allocator reaches 99% of full-fusion AP@0.5 while transmitting 20.0 kB per frame instead of 282.9 kB; with int4 fidelity it matches full-fusion accuracy at 10.0 kB (3.5% of the volume), where a per-agent top-k baseline needs 50.0 kB; it improves on that baseline by +0.3 to +0.9 AP points at equal budgets of 10-50 kB; and sharing the helpers' box lists adds two to three points more for under 1 kB. The allocator's own inference costs 0.02% of the total energy. We conclude that, on the radio energy that selection controls, it reaches 97% of full-fusion accuracy at lower energy than the baseline in all 243 parameterisations tested, and in 240 of 243 on total energy, where the helpers' compute dominates.

Keywords: Cooperative Perception; Vehicle-to-Everything (V2X); Bandwidth Allocation; Edge-IoT; Energy Modelling; Autonomous Vehicles

1 INTRODUCTION

One approach for increasing the perception range of vehicles is to enable them to exchange sensor-derived information, such as localized maps or feature points, with neighboring vehicles and/or roadside units. In the area of cooperative perception, such information can be provided in the form of intermediate neural feature maps. Most state-of-the-art approaches for cooperative perception involve the transmission of feature maps, such as obtained by a PointPillars-style detector [1], that are of an intermediate size (in terms of number of features and corresponding spatial support)

compared to raw point cloud data and detected boxes. For vehicular communication, feature maps are extremely large, often in the order of several megabytes per frame, whereas, for example, the channel capacity of a cellular sidelink or a 5G V2X link is typically on the order of tens of Mb/s [2] and supported by all vehicles within communication range. Hence, for practical cooperative perception applications, the main problem becomes that of deciding what to share under a fixed budget.

Existing communication-efficient methods for cooperative perception focus on local choices, i.e., Who2com and When2com for selecting which agents to receive information from [3], [4] and Where2comm, How2comm for choosing which regions to share, with the former approach also considering when to share in terms of flow-guided delay compensation and importance-aware selection [5], [6]. These approaches treat the budget for each individual agent as soft, fixing a confidence threshold for features sharing. All of these methods report transmitted bytes but do not take into account the energy and latency costs for perception. All methods are evaluated in terms of accuracy versus bytes, and vehicles are edge-IoT nodes where energy and latency are their main constraints for communication and computation. The energy cost of transmitting a byte on the radio is typically neglected in the vast majority of the perception literature, even though, as Section 4.5 shows, it is the helpers' compute rather than the radio that dominates the total at the measured 5G energy per bit. Furthermore, energy-efficient approaches for edge-IoT in the edge-computing literature usually do not report any results on perception accuracy in order to focus on other aspects of system as a whole.

The sender/region/fidelity selection is restructured into a knapsack problem over all items (helper, spatial block, fidelity) where the value of an item represents the expected marginal utility for the ego of sending this item over the channel. Since this quantity can only be known after fusion (i.e., after the feature maps have been received by the ego and fused), we learn to predict this quantity using 14 cheap features (the helper's confidence statistics in the block, the ego's own confidence there, which acts as its request, the geometry between block, helper and ego, and the number of helpers competing for the channel), that can be computed using feature maps that have not yet been sent (Section 3.4). We use an ensemble of predictors (an ensemble of LightGBM [7] models, an ensemble of XGBoost [8] models and a single multilayer perceptron [9]) to make the best possible predictions of the sparse count using these features where there are strong feature interactions.

In addition to these new metrics, we evaluate key properties that an edge-IoT deployment of cooperative perception would need to consider. These include robustness to noise in a vehicle's position estimates, as well as latency, scale to a realistic number of vehicles and penetration rate, and zero-shot transfer to real data. We model energy and latency to form a hardware-free evaluation of cooperative perception that does not require a testbed. Radio energy consumption is based on measured 5G modem power [10], while energy consumed by computation is modeled with a per-operation model that was calibrated on Jetson TX2 energies that were themselves measured and released in HW-NAS-Bench [11]. Latency to communicate a single byte on the radio to another vehicle is based on measured throughput and driving scenarios to form a set of 'throughput traces' that can be replayed to form a database of 'latency to communicate a given byte to a set of other vehicles' [12]. A sensitivity analysis over all constant inputs is performed so that the conclusions rest on the whole plausible range of those constants rather than on a single number we could not measure.

Contributions. (i) A novel way to tackle cooperative perception within a strict communication budget, which is formulated as a cross-agent problem and can be solved using a knapsack to also choose the numerical fidelity in addition to the spatial region to share. (ii) Comparative evaluation of feature sharing against per-agent confidence-based selection, random selection, late and early fusion and an oracle upper bound, where all considered methods report bytes under one explicit accounting convention, in combination with a set of edge-computing and communication experiments that can be run locally without physical hardware. (iii) Experimental evaluation of robustness, scalability, zero-shot transfer and energy/latency. (iv) Negative findings reported as such, including the failure of warm-started warped-feature fusion.

2 RELATED WORK

2.1 Cooperative perception

Early fusion (EF) approaches share raw point cloud data from individual sensors [13], whereas late fusion (LF) methods share detected objects between agents. In between these two extreme approaches, so called intermediate fusion (IF) methods share sensor-derived intermediate feature maps [14]–[17]. Cooperative perception using intermediate-feature maps has become the new state of the art as it combines high accuracy of raw-data based approaches with low communication bandwidth requirements. So far, several benchmark datasets for the evaluation of cooperative perception have been introduced, i.e., OPV2V [18] and V2X-Sim [19] for simulated scenarios, which rest on CARLA [20], OpenCDA [21] and SUMO [22], as well as DAIR-V2X [23] and V2V4Real [24] for real-world scenarios. Moreover, research

has been carried out to tackle challenges such as robustification against pose errors [25] and latency [26]–[28]. The field of cooperative perception has already been surveyed in [29].

2.2 Communication-efficient collaboration

These methods for communication-efficient cooperation all utilize a soft budget per sender, i.e., the amount of feature map parts to be transmitted by a sender is decided locally. Therefore, they all have the same two limitations: (1) the budget is soft and agent-dependent, i.e., there is no guarantee that the total number of bytes arriving at the ego vehicle fits the channel; (2) the method cannot move bytes of a sender with little information to send to a sender with much information to share.

2.3 Edge-IoT resource allocation and energy modelling

Dynamic resource allocation in edge-IoT has been recently studied in [30], where a neural controller is designed to optimize various edge-IoT systems. The energy-per-operation of various computing and communication modules can be studied in tables [31], and a hardware-aware benchmark (HW-NAS-Bench) for edge devices has been released recently in [11], which provides measurement-based latency and energy for various edge operations. We use such data to study radio energy and latency in addition to compute energy, and model them using measurement-based models of 5G modem power [10] and public 5G driving traces [12].

3 MATERIALS AND METHODS

3.1 Problem formulation

We consider an ego vehicle e and a set of $\mathcal{H}$ vehicles that are within communication range of the ego vehicle at time t . Each of the agents (including the ego vehicle) in the scenario computes an intermediate representation of the features corresponding to the region of interest (ROI) in the scene in front of the ego vehicle. We denote the intermediate feature map of a vehicle $j \in \mathcal{H}$ as $F_j \in \mathbb{R}^{C \times H \times W}$ where C is the number of channels of the feature map corresponding to the ROI in the bird's-eye-view of the scene. We explain in Section 3.2 how the feature maps corresponding to the ROI in the scene in front of the helper vehicles are computed. The feature maps are then fused by the ego vehicle to compute a single representation of the features corresponding to the ROI in the scene in front of the ego vehicle. We tile the grid into a set of blocks B where each block $b \in B$ corresponds to a set of 5×8 cells of the grid. Each helper $j \in \mathcal{H}$ computes a set of feature maps corresponding to the feature maps F_j and transmits a subset of the blocks B of the feature maps to the ego vehicle at a set of different fidelities $q \in \{\text{fp16}, \text{int8}, \text{int4}\}$. The cost of a helper j in transmitting a block $b \in B$ of the feature map computed by the helper at fidelity q is denoted by $c_{jbq} \in \mathbb{Z}^+$. The cost corresponds to the number of non-empty cells in the block b of the feature map computed by the helper j multiplied by the number of channels C of the feature map computed by the helper j multiplied by the number of bytes needed to represent each element of the feature map at the given fidelity q . There is an additional constant cost that corresponds to the fixed overhead of the vehicle's communication system. The total communication budget of the ego vehicle at time t is denoted by B in $\mathbb{Z}^+$.

$$\max_x \sum_{j,b,q} u_{jbq} x_{jbq} \quad \text{s.t.} \quad \sum_{j,b,q} c_{jbq} x_{jbq} \leq B, \quad \sum_q x_{jbq} \leq 1 \quad \forall j, b, \quad x_{jbq} \in \{0,1\} \quad (1)$$

We define the utility u_{jbq} as the expected improvement in the detection quality for the ego vehicle when this item is received, and then make use of this function to turn the problem into a multiple-choice knapsack problem, solved by the classical greedy algorithm [32], choosing items in decreasing order of utility-to-cost ratio. This is within a factor of two of the optimal solution for a standard knapsack problem, and runs in time to solve instances of hundreds of items per frame at a 10 Hz update rate.

3.2 Perception pipeline

Point clouds are pillarised at 0.4 m into a 200×704 pseudo-image over the OpenCOOD range $[-140.8, 140.8] \times [-40, 40]$ m, with eight hand-crafted statistics per pillar (occupancy, log count, mean/max/min/std height, mean intensity, height span) in place of a learned pillar encoder [1]. A three-stage convolutional backbone produces a 64-channel feature map at stride 4 (50×176 cells of 1.6 m); this map is what agents share. A PointPillars-style head with two anchors per cell (0° and 90°) predicts class and a seven-parameter box, trained with focal loss [33] and smooth-L1 regression as in SECOND [34]. Helpers encode their point cloud in the recipient's frame: the ego's pose is known from periodic beacons, so a helper transforms its points into the ego frame before pillarisation and its feature map is already on the ego grid ("recipient-aligned encoding"). This costs one backbone pass per recipient, which is charged in the energy model, and it removes the bilinear resampling of a 1.6 m-cell feature map that OpenCOOD's spatial warp [18] incurs; localisation error and delay still enter through the transform applied to the points. Agents' maps are fused

by a learned 1×1 convolution over the concatenation of the ego map, the element-wise maximum over helpers and the mean over helpers ("ConvFuse"), initialised to the identity on the ego block so that before fine-tuning the cooperative model reproduces single-agent detection exactly and training can only add helper information. We arrived at this design after two failures that Section 4.2 reports: OPV2V's parameter-free attention (AttFuse) [18], [35] passed almost no helper information when warm-started from single-agent weights, and a warped-feature ConvFuse lowered accuracy on objects the ego already saw, which we traced to resampling rather than to the fusion operator. During cooperative fine-tuning, helper blocks are dropped at random with a keep probability drawn uniformly from $[0.05, 1]$, so that partial sharing at test time is in-distribution, and labels are mask-aware: an object the ego cannot see is a positive only when a helper that sees it has the block containing it transmitted, otherwise its anchors are ignored, so the network is never trained to detect objects for which no evidence arrived.

3.3 Complementarity labels

For each frame, the single-agent detector is first run on all agents and the corresponding helper detections are projected to the ego's view. A vector of complementarity labels is then created for the ego by counting the ground-truth objects in each block b that (1) are centered in this block, (2) were not detected by the ego's single-agent detector (with BEV IoU < 0.5 for all its boxes for a given object) and (3) are detected by helper j on its own. The majority of a helper's blocks have complementarity labels equal to zero (i.e., positive rate of 0.94 %). This fact is in contrast to the information that could be obtained by thresholding the confidence of a helper in its detections for a given block (many objects detected by a helper are actually already detected by the ego).

3.4 Features and the utility ensemble

Each (helper, block) pair is described by fourteen features that cost nothing to transmit: the helper's maximum and mean confidence in the block and its number of non-empty cells; the ego's maximum and mean confidence and cell count, which act as its request; the confidence gap; block distance and bearing from the ego; helper distance and heading difference; block distance from the helper; the number of helpers; and the heuristic product $h_{\max}(1 - e_{\max})$. These features are used to count the number of positive objects the ego vehicle would have missed, which the correspondingly-requested helper can then detect. Three models were trained to count such numbers of positive objects, including a LightGBM model with a Poisson distribution for the count [7], an XGBoost model with a Poisson distribution for the count [8], and a two-layer multilayer perceptron (mlp) model on log-counts [9] with training performed using the Adam algorithm [36]. The average of the counts predicted by the three models is used by the allocator for ranking the positive objects that the ego could have missed.

3.5 Allocation and fidelity

To provide a sense of how much is being communicated and how much energy is being used, we report the numbers in bytes to facilitate comparison with other works. The feature matrix for requesting metadata of the helpers is a few hundred bytes large, the utility of the ego features is calculated and items are ranked based on the predicted utility. We then run a greedy knapsack problem to select items within the budget B of the ego, in the multi-fidelity setting we discount the utility for int8 and int4 for the decision (i.e., requesting the blocks instead of evaluating them), while at test time we symmetrically fake-quantize per channel [37] the block of helper features that the ego selected in order to report the accuracy. The selected blocks of helper features are masked into the ego grid and the cells of the unselected blocks are set to zero before they are fused with the corresponding blocks of the ego in the ConvFuse layer.

3.6 Byte accounting

In order to better compare bandwidth curves, all volume numbers in literature were fixed to one convention. The fp16 feature map is reported to have 2 Bytes per feature element (int8: 1 Byte, int4: 0.5 Byte). Block headers are reported to be of size 16 Bytes. The size of a helper message was reported to be 32 Bytes in total. For late fusion, the size of a late-fusion box is reported to be 16 Bytes in total. Early fusion, on the other hand, features 8 Bytes per early-fusion point. Only non-empty feature map cells are charged to the volume. Also, communication budgets B for all helper agents are reported to be total volumes for the ego frame in question.

3.7 Hardware-free energy and latency model

We use two energy models: radio energy and hardware-free energy. Radio energy E_{radio} in joules for bytes N_{bytes} to be transmitted at energy per bit E_{bit} in Joules per bit is linearly estimated from the SIGCOMM'21 5G measurement study [10] in a fixed-basis regression on all downlink throughput samples of length one second for a total of 31099 samples, and then the network-active power is added for the time on air at the link rate of 27 Mbps for V2XSet. The estimated marginal energy is 1.3 nJ/bit and the average energy over all measurements including the network-active power is 160 nJ/bit. We sweep both energy values in the sensitivity analysis in Section 4.6. For the hardware-free energy model, the compute energy E_{comp} in joules for N_{MAC} multiply-accumulate operations, N_{byte} bytes moved, and latency t_{lat} in seconds is a non-negative linear combination of E_{MAC} , E_{byte} , and P_{static} of the lengths of N_{MAC} , N_{byte} , and t_{lat} , with the

three coefficients for the three terms estimated by non-negative least squares on the latency and energy values that HW-NAS-Bench measured on a Jetson TX2 for 192 FBNet convolution blocks, with an energy fit R^2 of 0.96; the Eyeriss and FPGA models from the same benchmark are used only in the sensitivity sweep of Section 4.6. The MAC counts for all layers including the backbone, the head, the late-fusion boxes, and the fusion utility ensemble are obtained by registering forward hooks on each layer. The latency t_{lat} is modelled rather than measured: for the deadline analysis it is the transmission time at the replayed link rate plus a 10 ms backbone delay and a 5 ms scheduling overhead.

3.8 Experimental setup

We obtain the OPV2V benchmark [18] from a public mirror of the test data, since the official test data Google Drive folders were not available to us. We use the sixteen default-town scenarios (2,170 ego frames, 5,985 agent frames and 2–5 agents per frame), split by scenario into 9 training, 2 validation and 5 test scenarios. None of the test scenarios are used for training or validation. To imitate V2XSet-style [26] imperfections, we add xyz noise of $\sigma_{xyz} = 0.2$ m to each agent’s pose, as well as yaw noise of $\sigma_{\text{yaw}} = 0.2^\circ$. Furthermore, we add a one-frame delay of 100 ms to the agents’ measurements. For zero-shot transfer, we use the test split of the V2V4Real benchmark [24], which consists of real two-vehicle scenarios with corresponding ground truth poses. For simulating a penetration rate, in Multi-V2X [38] we cannot script an account, so we simply drop agents randomly.

We use the AP@0.5 and AP@0.7 metrics on all validation frames (each containing the visible objects of each agent), to measure the performance of the single-agent detector. After 3 epochs the single-agent detector reaches an AP@0.5 of 73.7% / AP@0.7 of 43.7%. We then perform cooperative fine-tuning, first for 2 epochs using attention fusion (the OPV2V implementation of AttFuse [18], [35]) followed by 4 epochs of ConvFuse with random block dropout and mask-aware labels. The detector with cooperative fine-tuning achieves full-fusion validation AP@0.5 of 41.6% / AP@0.7 of 36.0%, while the performance of the same model without helpers’ information (i.e. helpers are masked) is AP@0.5 of 39.7% / AP@0.7 of 32.7%. All experiments ran on a 16-thread laptop CPU with no GPU, the backbone has 0.78 M parameters, and the whole detection pipeline is released.

Baselines: We implement a number of natural baselines, i.e., no fusion; late fusion (the detectors independently report all bounding boxes which are then merged by non-maximum suppression); early fusion (all points from all detectors are merged into a single point cloud and then fed to a single detector); full intermediate fusion (all non-empty blocks are fused at fp16 precision, 282.9 kB per frame); random fusion under the budget constraint; and per-agent top-k fusion where each agent selects k blocks of highest request and the corresponding helper features are fused under the budget constraint that is split equally between all helpers. For Where2comm’s selection [5] of blocks to request from helpers, we implement and sweep the unbudgeted confidence threshold variant and the confidence-plus-request variant. We further compare against a simple knapsack allocation using the heuristic utility, $h_{\text{max}} (1 - e_{\text{max}})$, and learn an ensemble of regressors to approximate the true feature-based and then complementarity-based utility counts. We also consider the multi-fidelity version of the learned utility ensemble. Finally, we also consider an oracle that knows the true complementarity counts.

For performance evaluation we adopt average precision as in OpenCOOD, using greedy score-ordered matching between a score-ordered list of detections and the ground truth objects that are within the ego’s viewing range. A detection is correct if its IoU with the ground truth object is above 0.5 or 0.7 for bird’s-eye-view (BEV) polygons. Note that a single object can be detected by multiple agents. All detections are considered regardless of which agent detected them. Detections are scored globally and average precision uses the VOC-2010 all-point interpolation [39]. We also report the mean number of bytes per ego frame. For test, we use 271 frames (i.e., every second frame in all test scenarios) with on average 1.70 helpers in range.

4 RESULTS AND DISCUSSION

4.1 Baseline fusion schemes

This shows what bandwidth actually buys in this experiment. Full intermediate fusion lifts AP@0.5 from 24.8 % to 28.8 % but transmits 282.9 kB per ego frame, and early fusion costs an order of magnitude more again for no gain over it. Of the single level of fusion that is most accurate, the late fusion of a helper’s boxes is best at 30.5 % and sends very little in the way of feature bytes (only 0.4 kB). As described above, the Boxes+Feature helper sends the boxes that it has found as well as selected feature blocks from the feature map. These are referred to as the budgeted feature-sharing strategies below. As will be seen, very few of the feature bytes sent by the above strategies in order to achieve the levels of accuracy of the full-fusion case actually return any value greater than that which can be obtained by the helper’s own head using the computed feature map and the helper’s returns. The main purpose of the below strategies is therefore to establish how an allocation decision changes the number of feature bytes of any value that are returned.

Table 1: Baseline fusion schemes on the OPV2V test scenarios (same backbone, same frames).

Fusion scheme	Bytes / ego frame	AP@0.5 (%)	AP@0.7 (%)
No fusion	0.0 kB	24.8	16.2
Late fusion	0.4 kB	30.5	20.4
Early fusion	758.0 kB	30.0	21.9
Full intermediate fusion	282.9 kB	28.8	20.5

4.2 Accuracy-bandwidth frontier

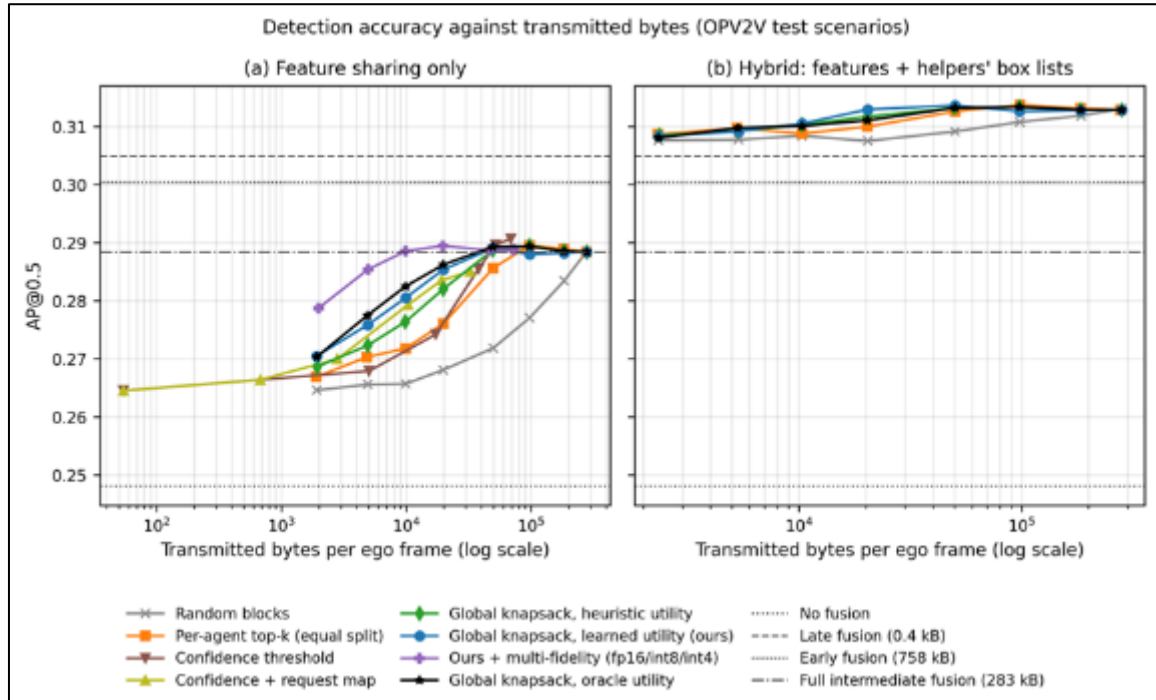


Figure 1: AP@0.5 against transmitted bytes per ego frame on the test scenarios. Horizontal lines: no fusion, late fusion, full intermediate fusion and early fusion.

Table 2: AP@0.5 / AP@0.7 (%) at three global budgets. Full fusion: 28.8 / 20.5 at 282.9 kB; no fusion: 24.8 / 16.2.

Strategy	B = 10.0 kB	B = 20.0 kB	B = 100.0 kB
Random blocks	26.6 / 17.6	26.8 / 17.6	27.7 / 18.7
Per-agent top-k (equal split)	27.2 / 17.9	27.6 / 18.5	29.0 / 20.5
Global knapsack, heuristic utility	27.6 / 18.0	28.2 / 18.6	28.9 / 20.4
Global knapsack, learned ensemble utility (ours)	28.1 / 18.4	28.5 / 19.0	28.8 / 20.4
Ours + multi-fidelity	28.9 / 19.6	28.9 / 20.3	28.9 / 20.4
Global knapsack, oracle utility	28.3 / 18.2	28.6 / 18.6	28.9 / 20.4
Per-agent top-k + helper boxes (hybrid)	30.9 / 20.5	31.0 / 20.8	31.4 / 22.0
Ours + helper boxes (hybrid)	31.1 / 20.8	31.3 / 21.2	31.3 / 21.9
Confidence threshold (nearest bytes)	26.8 / 17.7 (5.1 kB)	27.4 / 18.6 (17.3 kB)	29.1 / 20.6 (69.4 kB)
Confidence + request map (nearest bytes)	27.9 / 18.5 (10.4 kB)	28.4 / 18.9 (19.2 kB)	28.5 / 19.0 (32.6 kB)

We evaluated the learned agent allocator at a range of different budget sizes. We found that at 20.0 kB per frame (approximately 7% of the volume of full-fusion), the learned allocator reached 28.5 % AP@0.5 (99% of the full-fusion performance), compared to 27.6 % with per-agent top-k allocation and 26.8 % with random block allocation. At equal budgets of 10–50 kB, the learned allocator improved upon the per-agent top-k allocation by +0.3 to +0.9 AP points, and the multi-fidelity allocation improved upon the learned allocator by +0.3 to +1.7 AP points. Int4 fidelity matched the full-fusion performance at as low as 10.0 kB per frame (3.5% of the volume of full-fusion), whereas the per-agent allocation required 50.0 kB. All of the budgeted allocation methods saturate to full-fusion performance at approximately 100.0 kB per frame or greater, thus the main comparison is at these low budgets. The simple heuristic utility already captures most of the value of a global budget for cross-agent reallocation, with the ensemble and oracle adding on top. The multi-fidelity allocation packs more blocks into a given budget than fp16 allocation, thus it is most valuable at the lowest budgets where int4 allocation covers most of the scene. Hybrid sharing, where also the helpers transmit their box lists (0.4 kB per frame), is a very strong complement to the different allocation methods: late fusion alone reaches 30.5 %; the allocation + boxes reaches 31.3 % @ 20.0 kB (compared to 31.0 % for the per-agent top-k + boxes); and full fusion + boxes reaches 31.3 %; the dashed curves represent the different methods with hybrid sharing, and they all increase by roughly similar amounts, thus they do not change the ranking between the different allocation methods.

Failure to improve over no-fusion in negative testing. We have used OPV2V's implementation of the attention-based feature fusion provided in AttFuse [18], [35] to fine-tune the feature extractor part of a single agent's neural network for two epochs. The resulting AttFuse model was then used as the feature extractor for a set of frontier scenarios in negative testing. Rather than improve over no-fusion, the AttFuse-based full-fusion has accuracy of 28.4 % AP@0.5 compared to 27.8 % AP@0.5 for the corresponding late-fusion frontiers, while early fusion achieves 31.8 % and late fusion, which sends only the helpers' box lists, achieves 33.6 % AP@0.5. It is evident that all of the budgeted methods in Figure 1 for negative testing are within noise of no-fusion. We have also tried to use the identity-initialised ConvFuse feature fusion with the same set of scenarios, and this even decreased accuracy on the object detected by the ego agent of 78.4 % AP@0.5 to 71.0 % AP@0.5 while only increasing the accuracy on the helper-only detected objects from 52.1 % to 54.3 % AP@0.5. The largest decrease in accuracy was for the cases where both the ego and the helper are looking in the same general direction (relative yaw < 20°), with a decrease from 43.1 % to 38.5 % AP@0.5 for the object detected by the ego agent. We also tested the forward warp of the feature maps, and found that the warp was correct to an occupancy IoU of 0.70–0.85, whereas the inverse warp was incorrect to an IoU of 0.06–0.24. We conclude that the warm-started cooperative model falls below both early and late fusion, and that this is due to resampling of the received feature map of the recipient, which corrupts a box predictor trained on single-agent features that are not resampled. We have tested the recipient-aligned feature encoding described in Section 3.2 to remove this resampling artifact, and the resulting frontier is that shown in Figure 1. Note that warm-starting a cooperative model from single-agent weights, as opposed to training the fusion operator from scratch as in OPV2V, is the obvious thing to do with a small compute budget for training. But the resulting model fails in a way that is easy to misattribute to the fusion operator.

Table 3: Utility prediction quality on validation scenarios (173580 training rows, positive rate 0.94 %).

Utility model	Spearman ρ (val)	Precision of top-2 % blocks (%)
lgbm	0.120	21.4
xgb	0.126	20.8
mlp	0.113	18.8
ensemble	0.118	21.3

Table 3 gives a first impression of the quality of utility prediction. Ranking quality of the ensemble is what the allocator consumes for making predictions. Moreover, the individual members are competitive and the ensemble is the most stable across the validation scenarios. The most-used features of the LightGBM member are the block distance from the ego, the ego's mean and maximum confidence, and the helper distance. Geometric information of the scenario as well as the ego's uncertainty are rated equally to the helper's confidence in terms of importance for the predictor.

4.3 Imperfect communication

Adding localisation noise and delay to all fusion schemes in the same way as in V2X-ViT [26] degrades all fusion schemes. The only point worth making is that since our allocator selects blocks for the fusion model and does not attempt to align the blocks, the strategies are sensitive to the degradation of the fusion model with the added localisation noise. The ordering of strategies still holds with the one exception that under the strongest localisation noise (i.e., $\sigma = 0.6$ m, 0.6°) full fusion performs worst at 24.4 % and is even below no fusion at 24.8 %. The biggest impact in this case is that of the individual evidence blocks, however. In this strong localisation case the budgeted allocator that restricts itself to 20.0

kB performs best with 26.7 % while sending fewer but better evidence blocks. As before, packet loss is applied after the allocation step. For sake of conservative accounting a lost packet is counted as if it had been sent.

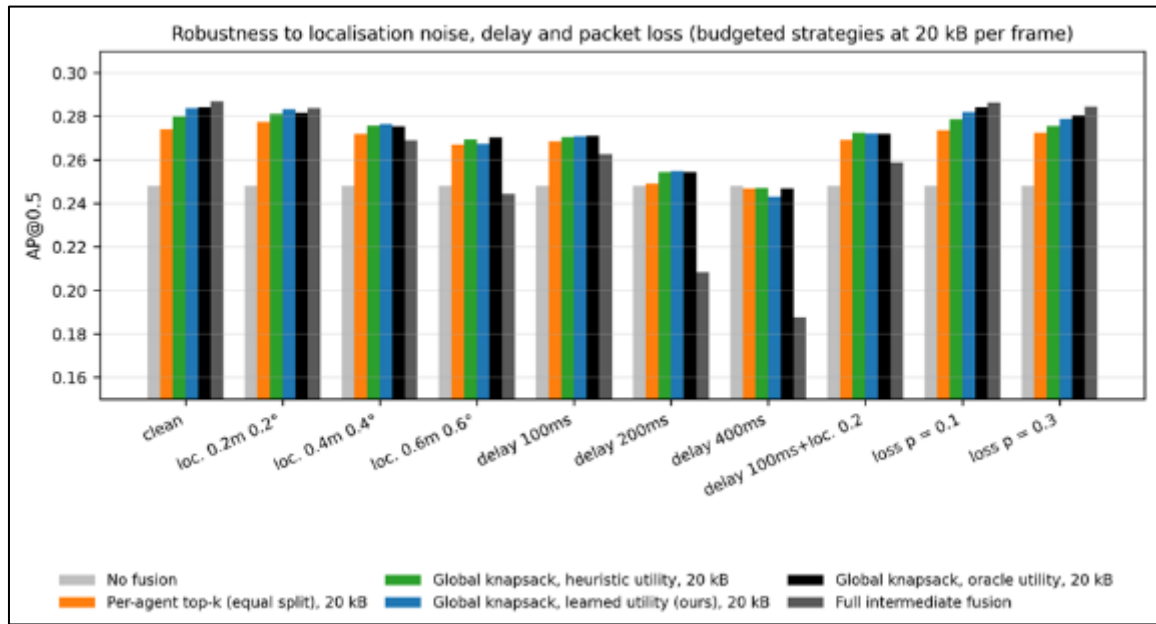


Figure 2: AP@0.5 under localisation noise, delay and packet loss at a 20.0 kB budget.

Table 4: AP@0.5 (%) under imperfect communication; budgeted strategies at B = 20.0 kB.

Condition	No fusion	Per-agent top-k (equal split)	Global knapsack	Global knapsack	Global knapsack	Full intermediate fusion
clean	24.8	27.4	28.0	28.4	28.4	28.7
loc_0.2m_0.2deg	24.8	27.7	28.1	28.3	28.2	28.4
loc_0.4m_0.4deg	24.8	27.2	27.6	27.6	27.5	26.9
loc_0.6m_0.6deg	24.8	26.7	26.9	26.7	27.0	24.4
delay_100ms	24.8	26.8	27.0	27.1	27.1	26.3
delay_200ms	24.8	24.9	25.4	25.5	25.4	20.8
delay_400ms	24.8	24.7	24.7	24.3	24.7	18.8
delay_100ms+loc_0.2	24.8	26.9	27.2	27.2	27.2	25.9
loss_p0.1	24.8	27.4	27.9	28.2	28.4	28.6
loss_p0.3	24.8	27.2	27.6	27.9	28.0	28.4

4.4 Scalability

Note that the fixed budget has to be spread over more candidates when there are more helpers than in the case of a fixed budget per agent. As described above, the global knapsack allocator concentrates bytes on the helpers that see something new, whereas a per-agent equal split gives every helper the same share regardless of what it can contribute. Thus the difference between the two methods is largest for the two agents in range. At a budget of 20.0 kB the margin of the allocator over the per-agent top-k is +0.8 AP points. The penetration axis describes a population over all frames. A frame without any helper in range renders all strategies to ego detection only and thus the ground truth set reduces to what the ego alone can detect for that frame. As a consequence the absolute AP goes up with decreasing penetration and only the gain over no fusion is comparable between the levels. The gain of the allocator at a budget of 20.0 kB is +3.6 AP points at 100 % penetration, +3.7 AP points at 75 % penetration, +3.3 AP points at 50 % penetration and +2.2 AP points at 25 % penetration. Note that the benefits of cooperation decrease with decreasing penetration and that

there is always some benefit to cooperation, however. The allocator was trained for the natural density of cars and is used without change.

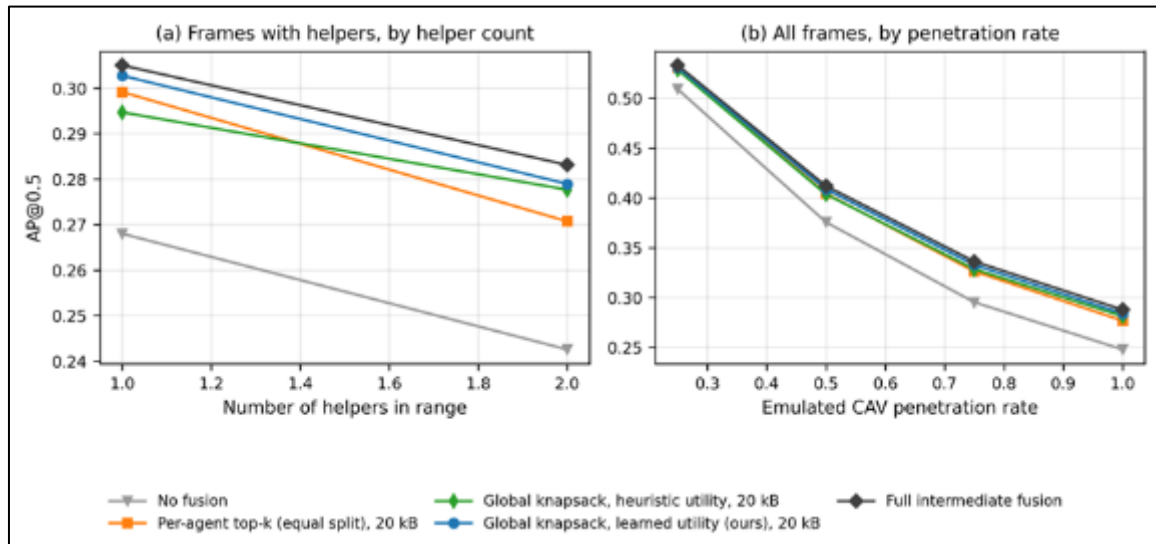


Figure 3: AP@0.5 versus number of helpers in range and emulated penetration rate (budget 20.0 kB).

Table 5: AP@0.5 (%) versus number of helpers in range (frames with helpers only) and emulated penetration rate (all frames; helper-less frames score ego-only for every strategy, and their ground truth is what the ego alone sees), B = 20.0 kB.

Axis	No fusion	Per-agent top-k	Ours	Full
helpers = 1	26.8	29.9	30.3	30.5
helpers = 2	24.3	27.1	27.9	28.3
penetration = 0.25	50.9	53.1	53.1	53.3
penetration = 0.5	37.6	40.4	40.9	41.2
penetration = 0.75	29.5	32.6	33.3	33.6
penetration = 1	24.8	27.7	28.4	28.8

4.5 Energy

Table 6 presents numbers typically not included in the perception-related literature. First, we can see that for full fusion the radio energy (362 mJ) is approximately 0.8× the ego's own backbone pass energy (457 mJ in Table 6), while the helpers' backbone passes add approximately 774 mJ, so the compute energy of all agents is 1232 mJ of a 1594 mJ total, which is greater than the radio term at the measured energy per bit. As one can see from the last two columns of Table 6, with low budgets the allocator removes most of the radio energy with negligible degradation in accuracy. Conversely, the compute energy of the helpers' backbones is a fixed cost of cooperation, which cannot be removed by any of the selection schemes, and thus, this is why late fusion is not free. Whether radio energy or compute energy dominates the total energy to communicate depends on the energy per bit, which is addressed in Section 4.6.

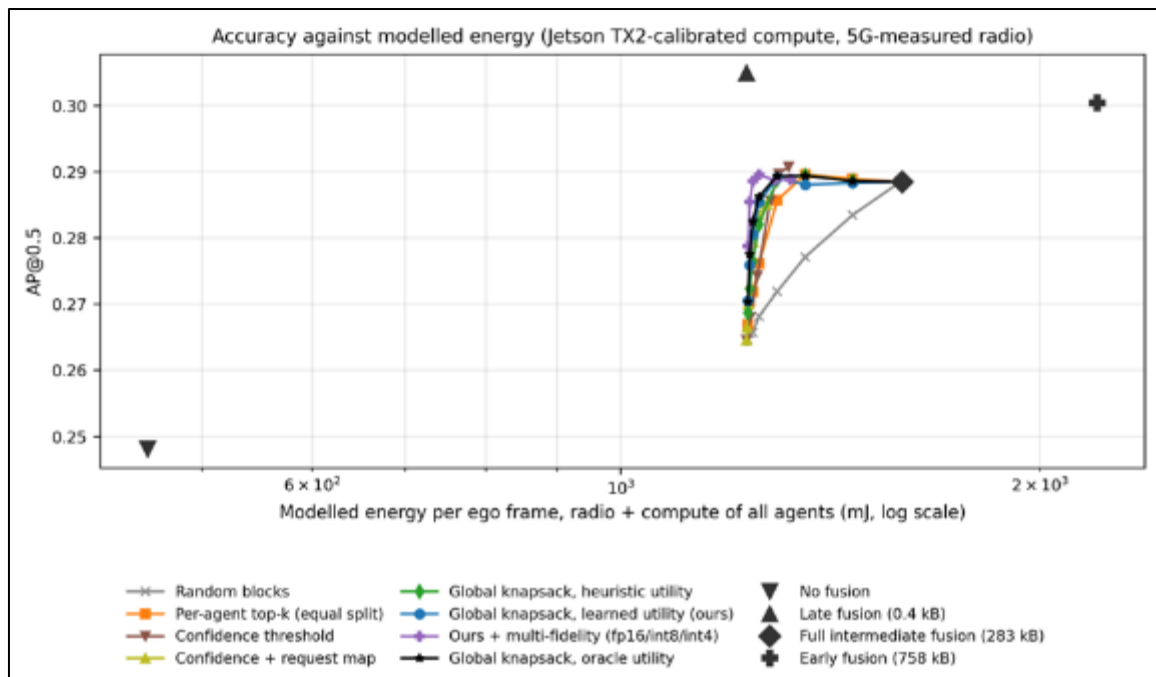


Figure 4: AP@0.5 against modelled energy per ego frame (radio plus compute of all agents).

Table 6: Modelled energy per ego frame (radio: 5G-measured J/bit at 27 Mbps; compute: HW-NAS-Bench-calibrated Jetson TX2 model).

Scheme	Bytes	Radio (mJ)	Compute, all agents (mJ)	Allocator (μJ)	Total (mJ)	AP@0.5 (%)
No fusion	0.0 kB	0.00	457.2	0.0	457.2	24.8
Late fusion	0.4 kB	0.54	1230.9	0.0	1231.4	30.5
Global knapsack, learned ensemble utility (ours) (B = 10.0 kB)	9.9 kB	12.70	1231.7	251.3	1244.7	28.1
Global knapsack, learned ensemble utility (ours) (B = 20.0 kB)	19.9 kB	25.50	1231.7	251.3	1257.5	28.5
Global knapsack, learned ensemble utility (ours) (B = 100.0 kB)	97.6 kB	124.82	1231.7	251.3	1356.8	28.8
Per-agent top-k (equal split) (B = 20.0 kB)	19.9 kB	25.48	1231.7	0.0	1257.2	27.6
Full intermediate fusion	282.9 kB	361.99	1231.7	0.0	1593.7	28.8

4.6 Sensitivity of the energy conclusion

There are 243 possible parameterizations of a selection scheme (3 device models * 9 J/bit values * 3 J/MAC scalings * 3 static-power scalings). The radio energy is between 0 % and 97 % of the total energy needed for full fusion (median 29 %). Thus, the learned allocator uses less energy than per-agent top-k in 240 out of 243 configurations for reaching 97 % of full-fusion AP@0.5, i.e. 3 configurations use more energy (all of them are at the marginal 1.3 nJ/bit value with doubled compute cost, where radio energy is negligible and the allocator's own inference decides the comparison by less than 0.01 %; median ratio 0.95). The multi-fidelity variant also needs less energy in 240 out of 243 configurations. When restricted to the radio energy, which is the only energy term that a selection scheme can affect, the allocator uses less energy than top-k in 243 out of 243 configurations at 97 % AP (median ratio 0.20) and also in 243 out of 243 configurations at 99 % AP (median ratio 1.00). At 97 % AP the learned allocator's radio energy is a median of 4 % of the full fusion's radio energy, i.e. the energy saved by the allocator is mostly radio energy. At 99 % AP both methods need the same budget of 50.0 kB; hence, their radio terms are equal (median ratio 1.00); the cost of the allocator's own

inference makes its total marginally higher in 144 out of 243 configurations. The multi-fidelity variant needs less energy than top-k in 240 out of 243 configurations at 99 % AP. The absolute values of the energies change by orders of magnitude in the sweep, but the ordering of the selection schemes w.r.t. to the radio energy does not change, i.e. this is what we claimed. Hence, the share of energy that can be influenced by a selection scheme is determined by the energy per bit.

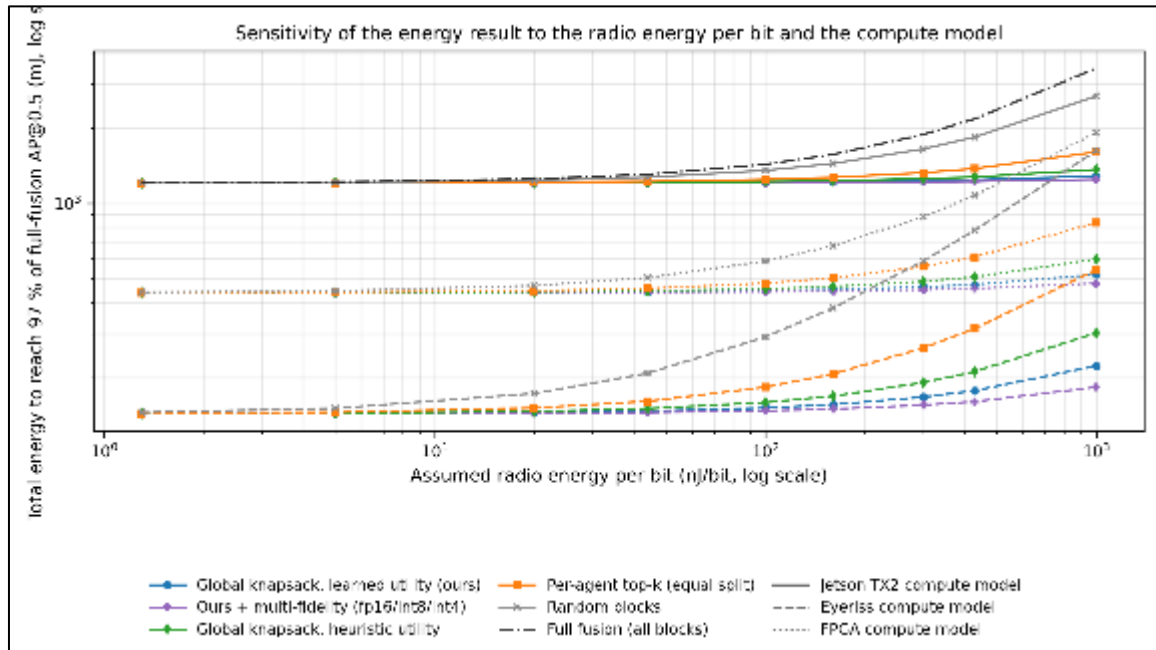


Figure 5: Total energy needed to reach 97 % of full-fusion AP@0.5 as a function of the assumed radio energy per bit, for three device models (solid Jetson TX2, dashed Eyeriss, dotted FPGA).

4.7 Latency over real 5G traces

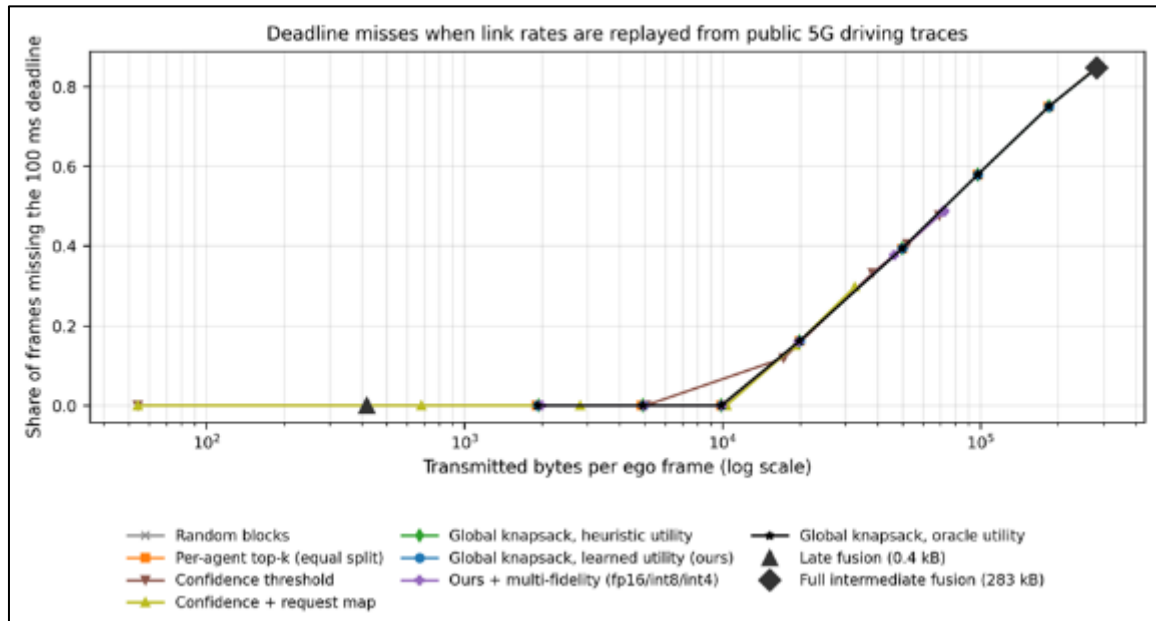


Figure 6: Deadline-miss rate (100 ms) against transmitted bytes when link rates are replayed from public 5G driving traces.

Table 7: Latency over replayed 5G driving traces (100 ms deadline, helpers share the link); last column: V2XSet's constant 27 Mbps link.

Scheme	Bytes	Latency p50 (ms)	Latency p95 (ms)	Deadline misses (%)	Deadline-adjusted AP@0.5 (%)	Latency @27 Mbps (ms)
Global knapsack, learned ensemble utility (ours) (B = 10.0 kB)	9.9 kB	26	78	0.0	28.1	18
Global knapsack, learned ensemble utility (ours) (B = 20.0 kB)	19.9 kB	37	141	16.2	27.9	21
Global knapsack, learned ensemble utility (ours) (B = 100.0 kB)	97.6 kB	125	632	57.8	26.5	44
Per-agent top-k (equal split) (B = 20.0 kB)	19.9 kB	37	141	16.1	27.2	21
Late fusion	0.4 kB	15	18	0.0	30.5	15
Full intermediate fusion	282.9 kB	333	1804	84.6	25.4	99

The link rates for the replayed links are calculated over the 60 driving trace files, for the seconds within a trace where there was active transfer at rates greater than 1 Mbps. The results are plotted in the table below with the corresponding latency at constant rate. Recall that the V2XSet assumes link rates of 27Mbps. Using these rates, full fusion misses the 100 ms deadline in 85 % of the resulting frames, whereas the 10.0 kB budget misses it in 0 % of them; the deadline-adjusted AP therefore favours the small budgets more strongly than the byte frontier alone suggests.

4.8 Zero-shot transfer to real data

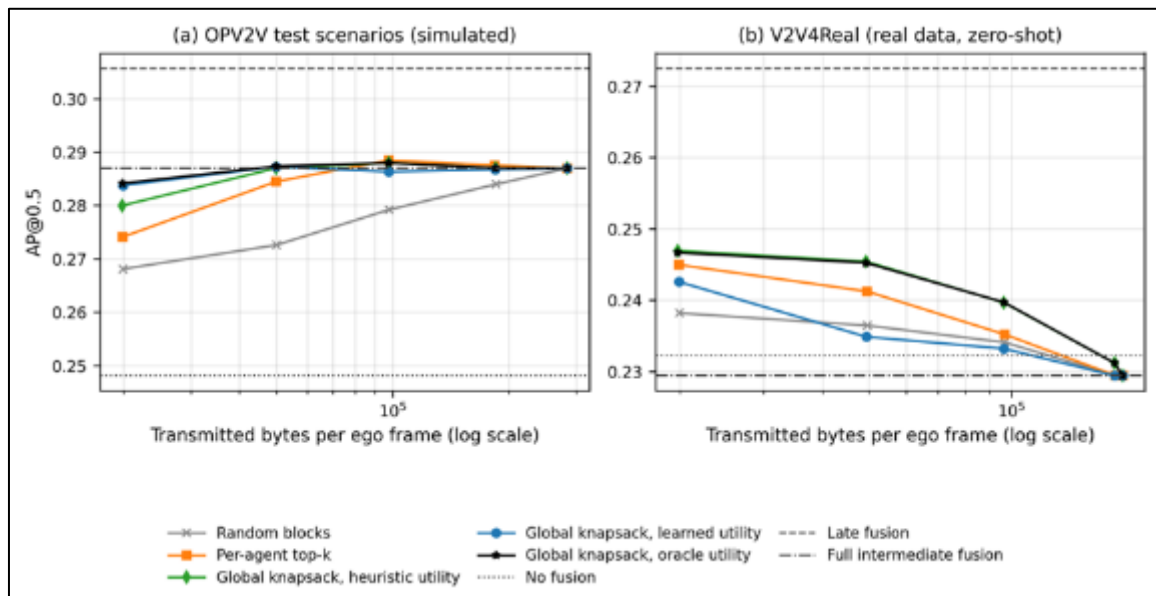
**Figure 7: Accuracy-bytes curves in the simulated test domain and on real V2V4Real data with all components frozen.**

Table 8: Zero-shot transfer: models and utility ensemble trained on OPV2V only; V2V4Real is real two-vehicle data.

Domain	Scheme	Bytes	AP@0.5 (%)	AP@0.7 (%)
opv2v_test	No fusion	0.0 kB	24.8	16.5
opv2v_test	Late fusion	0.4 kB	30.6	20.4
opv2v_test	Per-agent top-k (equal split) (B = 20.0 kB)	19.9 kB	27.4	18.7
opv2v_test	Global knapsack, heuristic utility (B = 20.0 kB)	19.9 kB	28.0	18.7
opv2v_test	Global knapsack, learned ensemble utility (ours) (B = 20.0 kB)	19.9 kB	28.4	19.1
opv2v_test	Full intermediate fusion	283.9 kB	28.7	20.6
v2v4real_test	No fusion	0.0 kB	23.2	9.7
v2v4real_test	Late fusion	0.1 kB	27.3	10.7
v2v4real_test	Per-agent top-k (equal split) (B = 20.0 kB)	19.9 kB	24.5	10.1
v2v4real_test	Global knapsack, heuristic utility (B = 20.0 kB)	19.8 kB	24.7	9.8
v2v4real_test	Global knapsack, learned ensemble utility (ours) (B = 20.0 kB)	19.9 kB	24.3	10.5
v2v4real_test	Full intermediate fusion	171.7 kB	22.9	10.1

For the V2V4Real test case, accuracy drops as expected since the detector was only trained on simulated data from a small subset of OPV2V. Number of helpers is set to at most one for these tests. Our purpose here is to test whether or not the advantages of the allocator over per-agent selection as well as the learned utility over the (heuristic) detector's confidence value hold when the detector and allocators are tested on real LiDAR returns, object poses, and object statistics without having to fine-tune anything. Late fusion performs very well here (27.3% vs. 23.2% without fusion), whereas full feature fusion performs much worse (22.9% vs. no-fusion 23.2%) since features learned from simulated returns are only partially useful on real returns and sending all features would add more noise than signal. A small budget is useful in this scenario, i.e., for the selected budget of 20.0kB, the knapsack-allocator with the heuristic utility reaches 24.7% and the learned ensemble 24.3%, against 24.5% for per-agent top-k and 23.8% for random blocks. The heuristic knapsack is therefore the best of the budgeted schemes on real data, while the learned ensemble falls slightly below per-agent top-k. Accuracy, however, decreases with increasing budget towards full-fusion. In summary, on real data, a budget not only constraints the number of features that can be sent but also acts as a regularizer, and the learned utility, fitted to simulated confidence statistics, is slightly worse than the geometry-and-confidence heuristic.

Limitations

Scale. The CPU-trainable backbone is a 0.78M-parameter network trained on a subset of OPV2V; hence the absolute AP values for all strategies are well below what has been reported for a full, GPU-trained network on the full OPV2V set. As mentioned, all strategies are compared on the same data and same frames. Therefore all claims are made with respect to relative change.

Utility proxy: complementarity counts are a proxy for the marginal gain in AP. The oracle row of each results table bounds what this proxy can deliver with this allocator. Note that the counts ignore all instances where both the helper and the ego detector fail to detect an object.

Assumed constants. All constants used in the framework such as link bandwidth (27 Mbps), backbone delay (10 ms), overhead (5 ms) and deadline (100 ms) were assumed based on prior V2XSet research.

5 CONCLUSION

In summary, in this paper we have transformed communication-efficient cooperative perception to a hard budget allocation problem over the set of all helper vehicles, spatial blocks of the ego grid and degrees of numerical fidelity, and

learned a value function from the metadata that are available before a block of features is actually transmitted. The resulting method is then evaluated in terms of the true measures that constrain an edge-IoT deployment: bytes of communication, joules of energy, and a set of fixed, strict, deadlines. We showed that a global allocator, that makes allocation decisions on the fly, recovers the full-fusion performance at a tiny fraction of the bytes transferred by the fusion. Int4 blocks alone achieve 3.5% of the volume of features transferred by the full-fusion case. In terms of allocation decisions, the global allocator improves on the best per-agent method, top-k, by 0.3–0.9 AP points at equal budgets of 10–50 kB, which is the regime that a single shared channel and a 100 ms deadline offer. The cost of the global allocator is negligible compared to the communication cost. In terms of radio energy, the use of an allocator to choose which features to send to help achieve a goal, achieves advantage in every tested configuration of a radio energy model, and the cost of energy to communicate with other vehicles is a tiny fraction of the cost of energy to compute features at the locations of other vehicles, as measured for 5G. The results also show that the allocation method transfers to real data, but with the constraint of a small budget actually helping, whereas a large budget hurts. Also, a single list of boxes for a helper vehicle is a stronger message, at this scale of features, than all of the features of a helper vehicle put together, so the best allocation method, that we tested, is hybrid. We note that, warm-started warped-feature fusion (as opposed to feature fusion alone), fails through a series of resamplings of the input features rather than through the fusion operator itself. Three immediate issues for future work are (1) learning the value function jointly with the fusion model, (2) improving the solver from a simple, greedy allocation to exact allocation when the number of helper vehicles is small, and (3) evaluating the end-to-end pipeline on an embedded device, where the modelled terms can be replaced by the real thing, without changing the experiments themselves.

STATEMENTS ON COMPLIANCE WITH ETHICAL STANDARDS

Acknowledgments

The code, the evaluation harness and the modelled energy and latency tables are released with the paper.

Disclosure of conflict of interest

The author declares no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

REFERENCES

- [1] Lang AH, Vora S, Caesar H, Zhou L, Yang J, Beijbom O. PointPillars: Fast encoders for object detection from point clouds. In: Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR). 2019. p. 12697–12705. doi:10.1109/CVPR.2019.01298
- [2] Molina-Masegosa R, Gozalvez J. LTE-V for sidelink 5G V2X vehicular communications: A new 5G technology for short-range vehicle-to-everything communications. IEEE Vehicular Technology Magazine. 2017;12(4):30–39. doi:10.1109/MVT.2017.2752798
- [3] Liu YC, Tian J, Ma CY, Glaser N, Kuo CW, Kira Z. Who2com: Collaborative perception via learnable handshake communication. In: Proc. IEEE Int. Conf. Robotics and Automation (ICRA). 2020. p. 6876–6883. doi:10.1109/ICRA40945.2020.9197364
- [4] Liu YC, Tian J, Glaser N, Kira Z. When2com: Multi-agent perception via communication graph grouping. In: Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR). 2020. p. 4106–4115. doi:10.1109/CVPR42600.2020.00416
- [5] Hu Y, Fang S, Lei Z, Zhong Y, Chen S. Where2comm: Communication-efficient collaborative perception via spatial confidence maps. In: Advances in Neural Information Processing Systems (NeurIPS). 2022. arXiv:2209.12836
- [6] Yang D, Yang K, Wang Y, Liu J, Xu Z, Yin R, et al. How2comm: Communication-efficient and collaboration-pragmatic multi-agent perception. In: Advances in Neural Information Processing Systems (NeurIPS). 2023. Available from: <https://openreview.net/forum?id=Dbaxm9ujq6>
- [7] Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, et al. LightGBM: A highly efficient gradient boosting decision tree. In: Advances in Neural Information Processing Systems (NeurIPS). 2017. p. 3146–3154. Available from: <https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html>
- [8] Chen T, Guestrin C. XGBoost: A scalable tree boosting system. In: Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD). 2016. p. 785–794. doi:10.1145/2939672.2939785
- [9] Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Nature. 1986;323:533–536. doi:10.1038/323533a0

- [10] Narayanan A, Zhang X, Zhu R, Hassan A, Jin S, Zhu X, et al. A variegated look at 5G in the wild: Performance, power, and QoE implications. In: Proc. ACM SIGCOMM Conf.. 2021. p. 610–625. doi:10.1145/3452296.3472923
- [11] Li C, Yu Z, Fu Y, Zhang Y, Zhao Y, You H, et al. HW-NAS-Bench: Hardware-aware neural architecture search benchmark. In: Proc. Int. Conf. Learning Representations (ICLR). 2021. arXiv:2103.10584
- [12] Raca D, Leahy D, Sreenan CJ, Quinlan JJ. Beyond throughput, the next generation: A 5G dataset with channel and context metrics. In: Proc. ACM Multimedia Systems Conf. (MMSys). 2020. p. 303–308. doi:10.1145/3339825.3394938
- [13] Chen Q, Tang S, Yang Q, Fu S. Cooper: Cooperative perception for connected autonomous vehicles based on 3D point clouds. In: Proc. IEEE Int. Conf. Distributed Computing Systems (ICDCS). 2019. p. 514–524. doi:10.1109/ICDCS.2019.00058
- [14] Chen Q, Ma X, Tang S, Guo J, Yang Q, Fu S. F-Cooper: Feature based cooperative perception for autonomous vehicle edge computing system using 3D point clouds. In: Proc. ACM/IEEE Symp. Edge Computing (SEC). 2019. p. 88–100. doi:10.1145/3318216.3363300
- [15] Wang TH, Manivasagam S, Liang M, Yang B, Zeng W, Urtasun R. V2VNet: Vehicle-to-vehicle communication for joint perception and prediction. In: Proc. European Conf. Computer Vision (ECCV). 2020. p. 605–621. doi:10.1007/978-3-030-58536-5_36
- [16] Li Y, Ren S, Wu P, Chen S, Feng C, Zhang W. Learning distilled collaboration graph for multi-agent perception. In: Advances in Neural Information Processing Systems (NeurIPS). 2021. arXiv:2111.00643
- [17] Xu R, Tu Z, Xiang H, Shao W, Zhou B, Ma J. CoBEVT: Cooperative bird's eye view semantic segmentation with sparse transformers. In: Proc. Conf. Robot Learning (CoRL). 2022. arXiv:2207.02202
- [18] Xu R, Xiang H, Xia X, Han X, Li J, Ma J. OPV2V: An open benchmark dataset and fusion pipeline for perception with vehicle-to-vehicle communication. In: Proc. IEEE Int. Conf. Robotics and Automation (ICRA). 2022. p. 2583–2589. doi:10.1109/ICRA46639.2022.9812038
- [19] Li Y, Ma D, An Z, Wang Z, Zhong Y, Chen S, et al. V2X-Sim: Multi-agent collaborative perception dataset and benchmark for autonomous driving. IEEE Robotics and Automation Letters. 2022;7(4):10914–10921. doi:10.1109/LRA.2022.3192802
- [20] Dosovitskiy A, Ros G, Codevilla F, Lopez A, Koltun V. CARLA: An open urban driving simulator. In: Proc. Conf. Robot Learning (CoRL). 2017. p. 1–16. arXiv:1711.03938
- [21] Xu R, Guo Y, Han X, Xia X, Xiang H, Ma J. OpenCDA: An open cooperative driving automation framework integrated with co-simulation. In: Proc. IEEE Int. Intelligent Transportation Systems Conf. (ITSC). 2021. p. 1155–1162. doi:10.1109/ITSC48978.2021.9564825
- [22] Lopez PA, Behrisch M, Bieker-Walz L, Erdmann J, Flotterod YP, Hilbrich R, et al. Microscopic traffic simulation using SUMO. In: Proc. IEEE Int. Intelligent Transportation Systems Conf. (ITSC). 2018. p. 2575–2582. doi:10.1109/ITSC.2018.8569938
- [23] Yu H, Luo Y, Shu M, Huo Y, Yang Z, Shi Y, et al. DAIR-V2X: A large-scale dataset for vehicle-infrastructure cooperative 3D object detection. In: Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR). 2022. p. 21361–21370. doi:10.1109/CVPR52688.2022.02067
- [24] Xu R, Xia X, Li J, Li H, Zhang S, Tu Z, et al. V2V4Real: A real-world large-scale dataset for vehicle-to-vehicle cooperative perception. In: Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR). 2023. p. 13712–13722. doi:10.1109/CVPR52729.2023.01318
- [25] Lu Y, Li Q, Liu B, Dianati M, Feng C, Chen S, et al. Robust collaborative 3D object detection in presence of pose errors. In: Proc. IEEE Int. Conf. Robotics and Automation (ICRA). 2023. p. 4812–4818. doi:10.1109/ICRA48891.2023.10160546
- [26] Xu R, Xiang H, Tu Z, Xia X, Yang MH, Ma J. V2X-ViT: Vehicle-to-everything cooperative perception with vision transformer. In: Proc. European Conf. Computer Vision (ECCV). 2022. p. 107–124. doi:10.1007/978-3-031-19842-7_7
- [27] Lei Z, Ren S, Hu Y, Zhang W, Chen S. Latency-aware collaborative perception. In: Proc. European Conf. Computer Vision (ECCV). 2022. p. 316–332. doi:10.1007/978-3-031-19824-3_19
- [28] Yu H, Tang Y, Xie E, Mao J, Luo P, Nie Z. Flow-based feature fusion for vehicle-infrastructure cooperative 3D object detection. In: Advances in Neural Information Processing Systems (NeurIPS). 2023. arXiv:2311.01682

- [29] Han Y, Zhang H, Li H, Jin Y, Lang C, Li Y. Collaborative perception in autonomous driving: Methods, datasets, and challenges. *IEEE Intelligent Transportation Systems Magazine*. 2023;15(6):131–151. doi:10.1109/MITS.2023.3298534
- [30] Almaini A, Folz J, Al-Dubai A, Zomaya A, Hawbani A, Zhao L. Intelligent dynamic resource allocation for edge-IoT systems using neural networks. In: *Proc. Int. Wireless Communications and Mobile Computing Conf. (IWCMC)*. 2026. p. 1277–1283.
- [31] Horowitz M. 1.1 Computing's energy problem (and what we can do about it). In: *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*. 2014. p. 10–14. doi:10.1109/ISSCC.2014.6757323
- [32] Dantzig GB. Discrete-variable extremum problems. *Operations Research*. 1957;5(2):266–288. doi:10.1287/opre.5.2.266
- [33] Lin TY, Goyal P, Girshick R, He K, Dollar P. Focal loss for dense object detection. In: *Proc. IEEE Int. Conf. Computer Vision (ICCV)*. 2017. p. 2980–2988. doi:10.1109/ICCV.2017.324
- [34] Yan Y, Mao Y, Li B. SECOND: Sparsely embedded convolutional detection. *Sensors*. 2018;18(10):3337. doi:10.3390/s18103337
- [35] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017. p. 5998–6008. arXiv:1706.03762
- [36] Kingma DP, Ba J. Adam: A method for stochastic optimization. In: *Proc. Int. Conf. Learning Representations (ICLR)*. 2015. arXiv:1412.6980
- [37] Jacob B, Kligys S, Chen B, Zhu M, Tang M, Howard A, et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In: *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*. 2018. p. 2704–2713. doi:10.1109/CVPR.2018.00286
- [38] Li R, Zhang X, Li C, Xu Y, Wang Y. Multi-V2X: A large scale multi-modal multi-penetration-rate dataset for cooperative perception. *arXiv preprint*. 2024. arXiv:2409.04980
- [39] Everingham M, Gool LV, Williams CKI, Winn J, Zisserman A. The PASCAL visual object classes (VOC) challenge. *Int. J. Computer Vision*. 2010;88(2):303–338. doi:10.1007/s11263-009-0275-4