Real-World Modeling of Computation Offloading for Neural Networks with Early Exits and Splits

Jan Danek, Zdenek Becvar, and Adam Janes
Faculty of Electrical Engineering, Czech Technical University in Prague, Prague, Czech Republic
danekja5@fel.cvut.cz, zdenek.becvar@fel.cvut.cz, janesada@fel.cvut.cz

***Abstract*—In this paper, we target offloading of computer vision-like applications based on convolutional neural network (CNN) from moving devices, such as mobile robots or autonomous vehicles, to Multi-Access Edge Computing (MEC) servers via a mobile network. In order to reduce overall CNN inference time, we design and implement CNN with early exits and splits, allowing partial or full offloading of CNN inference. Through real-world experiments, we analyze the impact of the CNN inference offloading on the total processing delay of CNN, energy consumption, and classification accuracy in a practical road sign recognition task. The results confirm that early exits and splits for CNN offloading can significantly reduce both total processing delay and energy consumption compared to full local processing while not impairing classification accuracy. Based on the results of the real-world experiments, we derive realistic models for energy consumption and total processing delay of CNN with early exits and splits. The derived models along with our open-source implementation of the offloading provide a foundation for future optimization of the computation offloading of computer vision-like applications with practical real-world aspects taken into account.**

***Index Terms*—Multi-access edge computing, mobile network, autonomous systems, vehicles, robots, convolutional neural networks, early exit, split, testbed, modeling.**

I. Introduction

Autonomous systems, such as autonomous vehicles (AVs) or robots, rely on real-time data processing for safe and effective operation in dynamic environment [1]. However, onboard computing resources of autonomous systems are often limited and may be insufficient for demanding data processing tasks, such as, computer vision. Besides, autonomous systems are constrained by a strict energy budget due to battery powering. The energy constraint limits practical use of powerful equipment, such as advanced graphical processing units (GPUs), for on board computation. To address these limitations, multi-access edge computing (MEC) has emerged as a promising approach. The MEC enables an offloading of computationally intensive tasks from the autonomous systems to nearby servers located close to base stations (BSs) in mobile networks, thereby reducing the total task processing delay encompassing both communication and computing delays [2].

Many recent works optimize utilization of the MEC resources via allocation of communication and computing resources to maximize the number of processed tasks [2], while the number of processed tasks is maximized, the total processing delay is not considered in such works, despite the delay is essential metric to ensure real-time operation of the autonomous systems. In addition, existing works, such as [3] typically focus on uplink transmission during CNN offloading to MEC server and neglect downlink delivery of the computing results. While in simplified theoretical communication models such assumption may be applicable, in practical application, the downlink delivery of results back to the autonomous system is not negligible, even though a low amount of data due to overhead imposed by frame alignment, scheduling, processing latency, and buffering delays, as shown in [4].

Furthermore, resource allocation for computation offloading depends on type of applications being processed. Due to the explosion of applications related to computer vision in both theory and practice, many recent works deal with offloading of machine learning-based computing tasks. An example of such task is convolutional neural network (CNN) for image processing (detection of objects in an image). Such type of applications, widely used in the AVs, can benefit from offloading from the AV to the MEC server due to high computational and energy demands of the CNN inference [5]. To reduce computational load related to CNN inference, novel CNN architectures support early exits [6]. The CNN architecture with early exits allow to classify inputs (e.g., objects in images) at earlier stages of the CNN provided that a required prediction confidence is reached [7], thus reducing the overall CNN processing delay.

Another option to reduce the delay of the CNN task processing is to split the CNN processing between the AV and the MEC server so that a part of layers is processed locally by the AV and a part is offloaded to the MEC server [8], [9]. The offloading of a part of CNN inference to the MEC server takes place at predefined split points. The split of CNN allows to accelerate the CNN inference in scenarios with the AVs that have limited computing power.

Another step beyond stand-alone early exits and stand-alone split is a combination of both into a single CNN. The papers [10] and [11] explore the combination of early exits and split computing to optimize CNN inference with focus on reducing total processing delay. While all the works on early exits, splits, and their combination demonstrate notable improvement in various metrics of CNN-based task processing in the mobile networks with MEC, the performance is evaluated only by simulations under over-optimistic assumptions. As a result,

This work was supported by Ministry of Education, Youth and Sport of the Czech Republic under grants no. LUASK22064 and LUABA24067, and by Czech Technical University in Prague under Grant no. SGS26/072/OHK3/1T/13.

the existing research does not capture key aspects of practical deployment, such as hardware constraints or practical aspects of mobile networks.

Motivated by the lack of the real-world evaluation and simplifying assumptions in the existing research works, we implement and evaluate offloading of CNN with split points and early exits on a real hardware of the AV wirelessly connected to the MEC server via mobile network. The major contributions of this paper are as follows:

1) We implement complete offloading process for CNN with early exits and split points on a real AV to demonstrate feasibility of the offloading in a practical real-world deployment. Unlike prior works, we account for *both uplink and downlink communication* and for *full processing pipeline* including commonly neglected local data preprocessing and transmission preparation.
2) As a key outcome, we derive *mathematical models* for the total processing delay and energy consumption of CNN inference with early exits and splits, *based on real-world measurements* to enable future research founded on realistic models not neglecting any practical aspect.
3) We *demonstrate the impact* of CNN's *early exits and splits* on total computing task processing delay, energy consumption of the AV, and accuracy of CNN decisions.
4) We make all developed codes and software for the AV and CNN as well as experimental data publicly available on GitLab[1] contributing to transparency and reproducibility of the work.

The rest of the paper is organized as follows. First, we outline a model of the system considered in this paper. Then, in Section III, we describe implementation of CNN with early exits and split points and integration of CNN to the testbed composed of MEC server, AV, and mobile network for offloading. Afterwards, the evaluation scenario for experiments is outlined in Section IV. Section V presents and discusses results of real-world experiments. In Section VI, we introduce the analytical models of the total processing delay and total energy consumption. Last, Section VII concludes the paper.

II. System model

In this section, we introduce modeling of the computation offloading of CNN from the AV to the MEC server via mobile network. We also define classification accuracy, the total processing delay and total energy consumption related to both offloading and local computing.

We consider a system composed of the AV and the BS, as illustrated in Fig. 1. The BS is enhanced with the MEC server for the processing of computing tasks. Such scenario allows us to carry out controlled experiments that are required to derive practical models for key performance indicators. Note that extension of experiments to multi-AV or multi-BS/MEC scenario is straightforward, but we leave it for future investigation, since such scenario introduces mutual relations (interference, handovers, etc.) that cannot be captured and analyzed within the page limit of this paper.

We focus on the tasks based on machine learning, such as computer vision tasks including road sign classification, based on CNNs. The CNN model used in our system, see Fig. 2, consists of N_L layers grouped into N_B blocks, where each block comprises a set of layers designed to extract specific features. The CNN includes N_E early exits [6] and N_S split points [8]. The split points are placed between blocks to reflect the CNN's structure [12], and early exits are co-located with split points to simplify design by sharing intermediate outputs.

The CNN can be processed fully on the AV, fully offloaded to the MEC server, or partially offloaded at any selected split point so that the layers before the split point are processed by the AV and remaining layers are processed by the MEC server. We define the variable S, which determines position of the split point within the CNN model, see Fig. 2. All layers before this split point are processed locally by the AV while all layers after split point are processed by the MEC server. If $S = 0$, all layers are processed by the MEC server. Conversely, if $S = N_S$, all layers are processed locally by the AV.

We also define E as the selected exit point of the CNN. If $E = N_E$, the main exit is used while $E < N_E$ represents the used early exit. The selection of the early exits influences the trade-off between computing cost and classification accuracy $A(E, S)$, which is defined as:

$$A(E,S) = \frac{N_{true}(E,S)}{N}, \tag{1}$$

where $N_{true}(E, S)$ represents the number of correctly classified objects, and N is the total number of inputs to the CNN.

To optimize data transmission efficiency between the AV and the MEC server, an autoencoder is deployed at each split point S to compress the output of CNN layers before transmission [13]. Without compression, the data size can even exceed volume of the original input image, increasing uplink communication delay and total energy consumption [14]. The encoder (on the AV) and the decoder (on the MEC) are implemented as convolutional layers, allowing continued CNN inference on reconstructed data [11].

The compression ratio R_S quantifies data reduction at the split point S and is defined as the ratio of the original data volume D^S_{orig} to the compressed data volume D^S_{comp}:

$$R_S = \frac{D^S_{orig}}{D^S_{comp}}. \tag{2}$$

To enable partial offloading, CNN inference is divided into subtasks between consecutive split points [11]. Each subtask is characterized by:

- Volume of data D^S_{ul} transferred from the AV to the MEC server. The volume depends on the selected split point S and equals $D^S_{ul} = 0$ in the case of local processing.
- Volume of computing results D^S_{dl} sent back from the MEC server to the AV. This volume remains the same for all split points except local processing of the entire CNN, in this case $D^S_{dl} = 0$.

[1] https://gitlab.fel.cvut.cz/mobile-and-wireless/codes/multi-exit-neural-network

- Computing demand $C_D = (C_D^1, \ldots, C_D^{N_S})$, where each C_D^i represents the required computing demand C_D of the CNN layers between two consecutive split points.

The computing power of the AV and the MEC server is denoted as C_{AV} and C_{MEC}, respectively. Computation at the AV is done by central processing unit (CPU) with a computing power C_{AV}. The MEC server is equipped with the CPU with a computing power C_{CPU} and with a GPU with a computing power C_{GPU}. The total computing power of the MEC server is $C_{MEC} = C_{CPU} + C_{GPU}$.

The communication between the AV and the BS is characterized by an uplink bitrate b_{ul} and a downlink bitrate b_{dl}, both depending on modulation, coding rate, and number of allocated resource blocks [15]. Thus, the uplink bitrate b_{ul} is defined as:

$$b_{ul} = N_{ul}^{RB} \times N_{ul}^{sub} \times N_{ul}^{bits} \times N_{ul}^{sym} \times CR_{ul}, \quad (3)$$

where N_{ul}^{RB} is the number of used resource blocks, N_{ul}^{sub} is the number of subcarriers per resource block, N_{ul}^{bits} is the number of bits per symbol, N_{ul}^{sym} is the number of symbols per subcarrier, and CR_{ul} is the code rate. The downlink bitrate b_{dl} is defined similarly with corresponding downlink parameters.

The overall communication delay $t_n^{comm,S}$ for the n-th CNN inference includes the uplink communication delay $t_n^{ul,S} = d_{ul} + \frac{D_{ul}^S}{b_{ul}}$ for offloading the computing task to the MEC server and the downlink communication delay $t_n^{dl,S} = d_{dl} + \frac{D_{dl}^S}{b_{dl}}$ for transferring the result back to the AV. The uplink delay d_{ul} and the downlink delay d_{dl} denote constant processing delays caused by mobile network processing overhead at the UE and BS, respectively. The processing delay arises from factors such as frame alignment, scheduling, processing latency, and buffering delays [4].

The overall communication delay of the n-th CNN inference is formulated as:

$$t_n^{\text{comm},S} = \begin{cases} t_n^{ul,S} + t_n^{dl,S} = \\ \left(d_{ul} + \frac{D_{ul}^S}{b_{ul}}\right) + \left(d_{dl} + \frac{D_{dl}^S}{b_{dl}}\right), & \text{if split } S \text{ is used,} \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

The computing delay $t_n^{comp}(E,S)$ of the n-th CNN inference is determined as:

$$\begin{aligned} t_n^{comp}(E,S) &= t_n^{AV}(E,S) + t_n^{MEC}(E,S) \\ &= \left(t_n^{prep,S} + \sum_{i=1}^{\min(S,E)} \frac{C_D^i}{C_{AV}}\right) + \left(\sum_{i=S+1}^{E} \frac{C_D^i}{C_{MEC}}\right), \end{aligned} \quad (5)$$

where the preprocessing delay $t_n^{prep,S}$ is the time required to prepare the data for transmission to the MEC server, $t_n^{AV}(E,S) = t_n^{prep,S} + t_n^{proc}(E,S) = t_n^{prep,S} + \sum_{i=1}^{\min(S,E)} \frac{C_D^i}{C_{AV}}$ represents the local computing delay on the AV including the preparation of data for transfer to the MEC server and all CNN layers before the split point S or the early exit E (whichever occurs first), $t_n^{MEC}(E,S) = \sum_{i=S+1}^{E} \frac{C_D^i}{C_{MEC}}$ denotes the MEC computing delay on the MEC server, which accounts for the computing of all CNN layers that follow the selected split point S up to the chosen exit E.

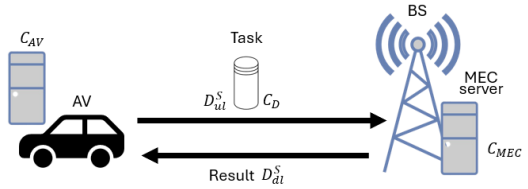

FIG. 1: System model with AV either computing/processing CNN inference locally or offloading the computation to MEC server via mobile network.

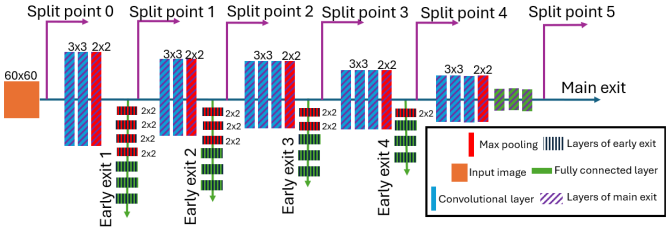

FIG. 2: Model of early exit CNN with split points.

The total processing delay $t_n^{total}(E,S)$ is defined as the total time required to process the n-th CNN inference locally or offloaded to the MEC server and includes both the computing delay $t_n^{comp}(E,S)$ and the communication delay $t_n^{comm,S}$, i.e.,

$$t_n^{total}(E,S) = t_n^{comp}(E,S) + t_n^{comm,S}. \quad (6)$$

Next, we define the total energy consumption $E_n^{total}(E,S)$ of the AV for processing the n-th CNN inference as:

$$\begin{aligned} &E_n^{total}(E,S) = E_n^{idle}(E,S) + E_n^{prep}(E,S) + \\ &E_n^{comp}(E,S) + E_n^{comm}(E,S) = \\ &= t_n^{total}(E,S) \times P_{idle} + t_n^{prep,S} \times P_{prep} \\ &+ t_n^{proc}(E,S) \times P_{comp} + t_n^{comm,S} \times P_{comm}. \end{aligned} \quad (7)$$

where $E_n^{idle}(E,S)$ is the idle energy consumed by the AV's CPU while waiting for results from the MEC server, $E_n^{prep}(E,S)$, $E_n^{comp}(E,S)$, and $E_n^{comm}(E,S)$ represent the energy consumed for data preprocessing, local CNN computing, and communication, respectively. Each energy component is proportional to the duration of corresponding phase and associated power consumption P_{idle}, P_{prep}, P_{proc}, and P_{comm}.

III. Implementation of CNN to AV and Mobile Network with MEC

This section describes the implementation of CNN with early exits and split points, enabling (partial) offloading to the MEC server. Details of the physical AV architecture and MEC integration are available in [4].

A. Architecture of CNN with early exit and split points

The implemented CNN, designed for road sign classification, is based on VGG-16 [16], leveraging its deep hierarchical structure and 3x3 convolutional kernels for efficient feature extraction in image classification tasks. However, its computational complexity limits real-time deployment in resource-constrained AVs. The implemented CNN consists of five blocks ($N_B = 5$), each followed by an exit of CNN, resulting in four early exits and one main exit (Fig. 2).

Furthermore, six split points are introduced between the blocks to enable partial offloading. One additional split point before the first convolution allows full offloading. At each split point, computation on the AV can be terminated, and the extracted feature representations are transmitted to the MEC server for further processing.

Since the total data volume at split points exceeds the input image size leading to higher communication delays in case of selection of later split points. To mitigate this, a compression mechanism based on autoencoders [13] is applied.

The CNN is trained and validated on the German Traffic Sign Recognition Benchmark (GTSRB) dataset [19], comprising 39,209 training and 12,630 validation images across 43 road sign classes.

B. Integration of AV and MEC server

The CNN can either be processed locally or (partially) offloaded to the MEC server over a software-defined 5G mobile network, implemented by means of OpenAirInterface[2] (OAI) and USRP N310 software-defined radio. Communication is handled via Internet protocol (IP) and transmission control protocol (TCP). The same CNN model is deployed on both the AV and the MEC server using Python and its library PyTorch. Data flow of CNN processing is shown in Fig. 3. For a detailed description of the AV's architecture, hardware components, and software integration, please refer to [4] or our GitLab[3].

IV. Experimental setup and scenario

This section outlines the experimental setup and scenario used to evaluate early exits and splits for CNN offloading implemented on real hardware, and defines performance metrics.

A. Experimental setup

Since this is the first experimental work on offloading of CNN early exits and split points from the AV to the MEC server, we target experiments in a controlled environment to be able to provide reliable results and findings without the interference caused by other AVs or handover effects. Thus, we deploy single AV and MEC server.

The MEC server uses CPU with C_{CPU} of up to 20.4 GFLOPS and GPU with C_{GPU} of 6500 GFLOPS corresponding to Intel Core i7-9700K CPU and NVIDIA RTX 2060 GPU, respectively. The AV is equipped with computing power C_{AV} of 7.36 GFLOPS provided by Intel Atom x7-E3950 CPU. Note that C_{GPU}, C_{CPU}, and C_{AV} represent theoretical maxima and may not be reached in practice. Communication between the AV and the MEC server is established via 5G network at 3.5 GHz (n78) with 20 MHz bandwidth and 106 resource blocks, using modulation and coding scheme 24 (64QAM, code rate 0.754) [15]. This configuration is one of the commonly used setups in OAI with using USRP N310.

[2]https://openairinterface.org

[3]https://gitlab.fel.cvut.cz/mobile-and-wireless/autonomous-driving/autonomous-driving-ROS1

TABLE I: Volume of data transferred in each split point D^S_{ul} and D^S_{dl} (including overhead) and computing demand for layers between split points C^S_D. Note that autoencoders are applied for partial offloading (i.e., $S = 1$–4).

Split S	**Uplink communication**				**Downlink communication**		**Computing demand**
	D^S_{orig} [kb]	D^S_{comp} [kb]	R_S [-]	D^S_{ul} [kb]	$D^S_{without}$ [kb]	D^S_{dl} [kb]	C^S_D [GFLOPS]
0	10.10	10.10	1	1749.8	1.4	1.6	0.145
1	56.25	7.03	8	1206.4	1.4	1.6	0.226
2	3.52	0.44	8	625.1	1.4	1.6	0.358
3	1.53	0.19	8	279.4	1.4	1.6	0.311
4	0.56	0.07	8	100.6	1.4	1.6	0.080
5	0	0	NaN	0	0	0	0

Table I presents parameters for each split point S including original data volume before compression D^S_{orig}, compressed data volume D^S_{comp}, transmitted data volume with TCP overhead and Python coding overhead D^S_{ul}, compression ratio R_S, computing result volume with TCP overhead D^S_{dl} and without TCP overhead $D^S_{without}$, and computing demand C^S_D.

B. Performance metrics

We consider the following performance metrics (see Section II): i) total energy consumption defined in (7), ii) total processing delay, defined in (6), and iii) classification accuracy, defined in (1).

The communication energy $E^{comm}_n(E, S)$, measured by USB power meter with 1 mWh resolution and ±1.41 % accuracy, captures 5G modem consumption during uplink and downlink communication and during computation on MEC server. The computing $E^{comp}_n(E, S)$, idle $E^{idle}_n(E, S)$, and preprocessing $E^{prep}_n(E, S)$ energies are measured via Intel RAPL [20], which measures the energy consumed by the AV's processor in different phases of offloading.

To ensure precise measurement of communication delays, the time bases of the AV and the MEC server are synchronized via the Network Time Protocol (NTP) [21].

The classification accuracy $A(E, S)$ is obtained by processing all images through specific early exit E and split point S.

C. Scenario of experiment

To ensure comparability across scenarios, the bitrate is limited to 25 Mbps in both uplink and downlink. Nevertheless, as the experiments are conducted over a real mobile network, unavoidable variations of the wireless environment (e.g., interference, fading, and load dynamics) are inherently present. Such effects cannot be fully eliminated, but these are also naturally a part of real-world conditions, thus, do not compromise the representativeness of the results.

Road sign images are randomly selected from the GTSRB dataset. In each experiment, a single road sign image is processed by CNN either locally of (partially) offloaded. Each experiment is repeated 250 times and results are averaged out.

V. Performance evaluation

In this section, we evaluate offloading of the CNN with splits and early exits. We present the total processing delay $t_n^{total}(E,S)$, total energy consumption $E_n^{total}(E,S)$, and classification accuracy $A(E,S)$ as functions of the split point S and early exit E. Note that the full offloading corresponds to $S=0$ while full local processing to $S=5$. If $E \leq S$, the CNN is processed entirely on the AV; otherwise, computation is split between the AV and MEC server.

Fig. 4 shows the classification accuracy $A(E,S)$ depending on early exit E and split point S. The $A(E,S)$ rises from 32% (exit 1) to 93% (main exit), confirming that later CNN layers improve $A(E,S)$. The most notable gain (from about 30% to over 80%) occurs between the first three exits, and further increase beyond exit 3 is marginal. For each exit, classification accuracy $A(E,S)$ stays nearly constant across all split points, indicating minimal impact of autoencoder compression.

Fig. 5 shows the total processing delay $t_n^{total}(E,S)$ for various exits E and split points S. Earlier exits reduce $t_n^{total}(E,S)$ significantly by limiting the number of computed layers. Conversely, later splits increase the local computing delay $t_n^{AV}(E,S)$ due to AV's limited computing power C_{AV}. Overall, early exits reduce $t_n^{total}(E,S)$ by up to 4.2 times compared to the main exit, with later splits increasing $t_n^{total}(E,S)$ as more layers are computed on the less computationally powerful AV before offloading, as expected according to (5). Full offloading reduces the $t_n^{total}(E,S)$ by up to 2.5 times compared to full local processing on the AV.

Fig. 6 shows that uplink communication delay $t_n^{ul,S}$ decreases with higher split points, as the transmitted data volume D_{ul}^S shrinks between splits, see Table I, in line with (4). Downlink communication delay $t_n^{dl,S}$ remains constant, except at S=5, where no offloading occurs. As the split moves toward later stages of the CNN, local computing delay $t_n^{AV}(E,S)$ increases due to more layers being computed at the AV, while MEC computing delay $t_n^{MEC}(E,S)$ decreases. However, MEC computing delay is up to 81.5 times lower than the local computing delay $t_n^{AV}(E,S)$, reflecting higher computing power C_{MEC} of MEC server. The preprocessing delay $t_n^{prep,S}$ is considered only for the uplink, as $t_n^{prep,S}$ varies with the volume of data transmitted from the AV to the MEC server D_{ul}^S, while the preparation of single classification result on the MEC server is negligible and not measurable.

Fig. 7 shows the total energy consumption $E_n^{total}(E,S)$ across exits and split points, highlighting the trade-off between local computing and offloading. Early exits reduce energy consumption by up to 4.4 times compared to the main exit. Full offloading keeps $E_n^{total}(E,S)$ low, as the AV remains mostly idle while the MEC server handles CNN inference. Fast MEC computation also shortens $t_n^{total}(E,S)$, further reducing energy. In contrast, higher splits shift more computation to the AV, increasing energy consumption.

Fig. 8 breaks down the energy consumption for the main exit into individual components. With full local processing, communication energy $E_n^{comm}(E,S)$ is zero, but total energy consumption $E_n^{total}(E,S)$ is 2.6 times higher than with full offloading due to increased $E_n^{comp}(E,S)$. In contrast, full offloading results in zero computing energy at the AV, with total energy consumption $E_n^{total}(E,S)$ dominated by idle energy $E_n^{idle}(E,S)$, as the AV waits for the MEC server to complete the CNN's processing. As the split increases, more CNN layers are computed locally, raising $E_n^{comp}(E,S)$.

TABLE II: Values of parameters for estimating total processing delay $t_n^{total}(E,S)$ and total energy consumption $E_n^{total}(E,S)$.

Parameter	Value	Parameter	Value	Parameter	Value
d_{ul} [ms]	22.81	b_{ul} [Mbps]	12.36	d_{dl} [ms]	7.19
b_{dl} [Mbps]	9.81	d_{AS} [ms]	43.69	d_{MEC} [ms]	1.12
d_{prep} [ms]	12.18	k_{prep} [ms/kb]	2.33	C_{MEC} [GFLOPS]	365.94
C_{AV} [GFLOPS]	3.62	P_{idle} [W]	4.62	P_{prep} [W]	4.92
P_{proc} [W]	5.17	P_{comm} [W]	0.79		

VI. Modeling of total processing time and total energy consumption

The experimental results presented in Section V, allow us to create realistic models of the total processing time $t_n^{total}(E,S)$ and the total energy consumption $E_n^{total}(E,S)$. The total processing delay is theoretically defined in (6), however, the following expressions for $t_n^{proc}(E,S)$, $t_n^{MEC}(E,S)$, and $t_n^{prep,S}$ provide refined formulations reflecting the real-world conditions observed during the experiments:

$$t_n^{proc}(E,S) = \sum_{i=1}^{\min(S,E)} \left(d_{AV} + \frac{C_D^i}{C_{AV}} \right), \tag{8}$$

$$t_n^{MEC}(E,S) = \sum_{i=S+1}^{E} \left(d_{MEC} + \frac{C_D^i}{C_{MEC}} \right), \tag{9}$$

$$t_n^{prep,S} = d_{prep} + k_{prep} D_{comp}^S, \tag{10}$$

where d_{AV} and d_{MEC} are the constant processing delays on the AV and MEC server, d_{prep} is the fixed delay at the start of preprocessing, while k_{prep} captures the delay that increases proportionally with the volume of transmitted data D_{comp}^S. Estimated parameter values are summarized in Table II.

In contrast to reformulated expressions above, the expression for $t_n^{comm,S}$ in (4) matches real-world experiment.

The total energy consumption $E_n^{total}(E,S)$ is modeled based on (7) and the estimated power consumption values P_{idle}, P_{prep}, P_{proc}, and P_{comm} are provided in Table II.

We present the estimated values for all parametrs in Table II. The higher computing power of the MEC server relative to the AV explains the reduction in inference time upon offloading. The communication delay parameters match the 5G network configuration described in section IV.

VII. Conclusion

We have demonstrated the feasibility of computation offloading from the autonomous systems to the MEC server using the AV and a software-defined mobile network, all deployed on real hardware in realistic environment. Focusing

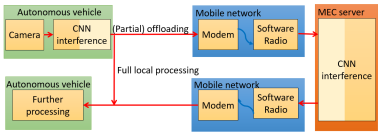

FIG. 3: Data flows from AV's camera to either (partial) offloading or full local processing on the AV. For offloading, data is transmitted via mobile network to MEC server for further computation.

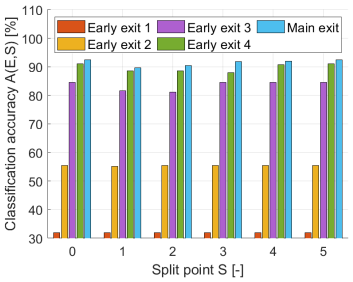

FIG. 4: Impact of split point S and early exit E on classification accuracy $A(E,S)$.

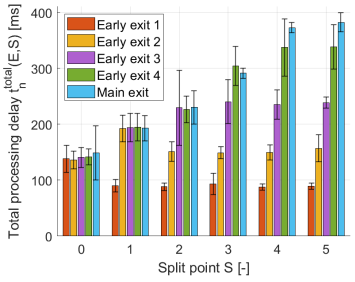

FIG. 5: Impact of split point S and early exit E on total processing delay $t_n^{total}(E,S)$. Whiskers represent a standard deviation (68% confidence).

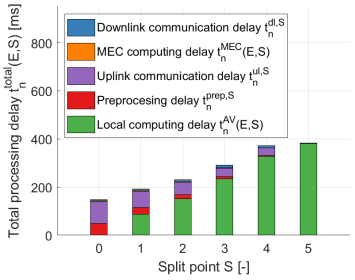

FIG. 6: Impact of split S on communication, computing, and preprocessing delays for main exit ($E = 5$).

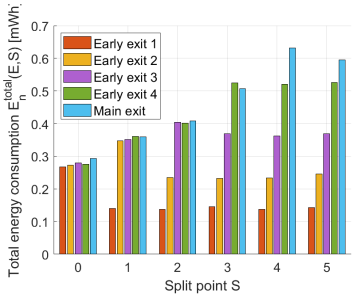

FIG. 7: Impact of splits point S and early exits E on total energy consumption $E_n^{total}(E,S)$.

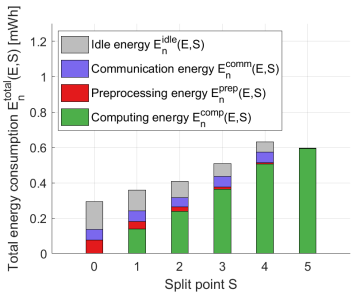

FIG. 8: Impact of split S on communication, computing, and preprocessing energy consumption for main exit ($E = 5$).

on road sign classification, we have evaluated CNN enhanced with early exits and split points to support partial or full offloading. Our results show that computation offloading reduces the total processing delay by up 2.5 times and the total energy consumption by up to 2.6 times compared to the local processing by the AV.

Classification accuracy depends on choice of the split point and the early exit. Exits located after more CNN layers improve classification accuracy at the cost of higher computing delay. We have also derived analytical models for the total processing delay and the total energy consumption based on the results of real-world experiments.

Future work should expand the system to multiple MEC servers and AVs and explore new offloading decision algorithm for choosing a suitable split and early exit dynamically according to actual channel conditions and resource availability.

REFERENCES

[1] J. Karangwa, J. Liu and Z. Zeng, "Vehicle Detection for Autonomous Driving: A Review of Algorithms and Datasets," IEEE T-ITS, 2023.

[2] L. Liu, et al. , "Deep Reinforcement Learning-based Dynamic SFC Deployment in IoT-MEC Networks," IEEE ICCC, 2022.

[3] H. Hu, et al., "Energy Efficiency and Delay Tradeoff in an MEC-Enabled Mobile IoT Network," IEEE Internet of Things Journal, 2022.

[4] J. Danek, Z. Becvar and A. Janes, "Computational Offloading for Autonomous Systems: Real-World Experiments and Modeling," IEEE Vehicular Technology Conference (IEEE VTC2025-Spring), 2025.

[5] N. Li, et al., Graph Reinforcement Learning-based CNN Inference Offloading in Dynamic Edge Computing. IEEE GLOBECOM, 2022.

[6] G. Casale and M. Roveri, "Scheduling Inputs in Early Exit Neural Networks," in IEEE Transactions on Computers, vol. 73, no. 2, 2024.

[7] S. Teerapittayanon, et al., "Branchynet: Fast inference via early exiting from deep neural networks," IEEE ICPR, 2016.

[8] A. Bakhtiarnia, et al., "Dynamic Split Computing for Efficient Deep EDGE Intelligence," IEEE ICASSP, pp. 1-5, 2024.

[9] H. Zhou, et al., "Accelerating Deep Learning Inference via Model Parallelism and Partial Computation Offloading," IEEE TPDS, 2023.

[10] R. Narmeen, P. Mach, Z. Becvar and I. Ahmad, "Joint Exit Selection and Offloading Decision for Applications Based on Deep Neural Networks," IEEE Internet of Things Journal, vol. 11, no. 23, pp. 38098-38112, 2024.

[11] R. Rauch, Z. Becvar, P. Mach and J. Gazda, "Cooperative Multi-Agent Deep Reinforcement Learning for Dynamic Task Execution and Resource Allocation in Vehicular Edge Computing," IEEE TVT, 2025.

[12] Y. Matsubara, et al., "Head Network Distillation: Splitting Distilled Deep Neural Networks for Resource-Constrained Edge Computing Systems," IEEE Access, 2020.

[13] Y. Matsubar, et al., "SC2 benchmark: Supervised compression for split computing," Transactions on machine learning research, 2023.

[14] R. Xu, et al., "Impact of data compression on energy consumption of wireless-networked handheld devices," ICDCS, 2003.

[15] 3GPP, "5G Mobile System Architecture, TS 23.501, version 17.5.0," PP Technical Specification, 2022.

[16] X. Gu, Q. Lang, F. Lin and P. Wang, "Attention-aware CNN model for Traffic Signs Classification," CBASE, pp. 83-87, 2022.

[17] F. Nikbakhtsarvestani, S. Rahnamayan and M. Ebrahimi, "Training Deep Neural Networks with Multi-Objective Adam Optimizer for Medical Image Classification," IEEE CIHM, 2025.

[18] S. Li, H.-T. Nguyen and C. C. Cheah, "A Theoretical Framework for End-to-End Learning of Deep Neural Networks With Applications to Robotics," IEEE Access, vol. 11, pp. 21992-22006, 2023.

[19] J. Stallkamp, M. Schlipsing, J. Salmen and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," Neural Networks, 2012.

[20] Intel Corporation, "Running Average Power Limit (RAPL) Energy Reporting," 2023. [Online]. Available: https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/advisory-guidance/running-average-power-limit-energy-reporting.htm. [Accessed 22 01 2026].

[21] N. Tripathi and N. Hubballi, "Preventing time synchronization in NTP broadcast mode," Computers & Security, vol. 102, 2021.