Edge Computing with Early Exiting for Adaptive Inference in Mobile Autonomous Systems

Edge Computing with Early Exiting for Adaptive Inference in Mobile Autonomous Systems

Simone Angelucci*, Roberto Valentini*, Marco Levorato†, Fortunato Santucci*, Carla Fabiana Chiasserini‡
* Dept. of Information Engineering, Computer Science and Mathematics, and Centre Ex-EMERGE, University of L'Aquila, IT
† The Donald Bren School of Information and Computer Science, UC Irvine, CA, US
‡ Dept. of Electronics and Telecommunications, Politecnico di Torino, Torino, IT

***Abstract*—Early Exiting (EE) is an emerging computing paradigm where Deep Neural Networks (DNNs) are equipped with earlier classifiers, enabling trading-off accuracy with inference latency. EE can be effectively combined with edge computing, a paradigm that allows mobile nodes to offload complex tasks, such as the execution of DNNs, to servers at the edge of the network, thus reducing computing times and energy consumption at the mobile devices. The integration of such technologies is particularly attractive for the support of applications for connected and automated driving. In this paper, we consider a system that jointly leverages the benefits of EE and edge computing, and we model their complex interactions by means of a Markov Decision Process (MDP). We then formulate an optimization problem to select the inference strategy that maximizes the average task accuracy. Importantly, such an optimization problem has low complexity, as the optimal policy can be derived by mapping the MDP into a linear program. Our numerical results focus on a use case centered on automated vehicles connected with an edge server under varying channel and network conditions, and show that our solution achieves up to 11% higher accuracy compared to the optimal policy with no EE.**

***Index Terms*—Early exiting, Edge computing, Mobile edge applications, Markov decision process, Connected and automated vehicles**

I. Introduction

Connected and automated vehicles (CAV) are envisioned to play a key role in future Intelligent Transportation Systems (ITS), as demonstrated by the many use cases identified by major organizations such as the Third Generation Partnership Project (3GPP) and the 5G Automotive Association (5GAA) [1]–[4]. Some of the prospected use cases heavily rely on Artificial Intelligence (AI). For instance, local dynamic maps [5] – databases that can be accessed by safety-related applications to retrieve meaningful information about a vehicle's surroundings – are populated by recognition of vulnerable road users, other vehicles, and obstacles surrounding the vehicle. AI tasks such as object detection/recognition and instance segmentation typically leverage resource-demanding and power-consuming Deep Neural Networks (DNNs), thus limiting their execution on mobile devices, including CAVs.

This research was partly funded by the project "Centre of Excellence on Connected, Geo-Localized and Cyber-secure Vehicles (EX-Emerge)" – Italian Government (Grant Number: CIPE Resolution 70/2017) and by the Horizon Europe project CENTRIC (Grant No. 101096379). Marco Levorato's work has been partially supported by the NSF, under grant CCF 2140154.

However, by connecting with edge servers positioned within the edge of wireless infrastructures, CAVs can offload compute-intense inference tasks. Existing work aims to identify effective offloading strategies that can satisfy application-dependent requirements. Among these studies, [6] proposes two offloading strategies, for independent and connected edge servers, highlighting that in the latter case task offloading between edge servers can effectively counteract users' mobility. A game-theoretic approach is used in [7] for the minimization of the overall tasks execution latency, while a deep reinforcement learning-based approach is proposed with the same goal in [8]. Optimal offloading strategies for latency minimization are derived in [9]–[11], within the framework of Markov Decision Processes (MDPs).

Interestingly, edge computing can be combined with trends that aim at making the execution of DNNs dynamic and adaptive with respect to input to reduce resource footprint and inference time. In this domain, an emerging paradigm is Early Exiting (EE) [12], according to which existing neural network architectures are modified to introduce early branches producing the same output as the full model with reduced complexity. Branches are executed sequentially, and the execution is terminated if the latest output has sufficient confidence. The use of EE may thus represent an effective way to trade off accuracy with inference times, especially when considering scenarios with severe latency constraints. Further, it enables tuning the inference performance to the system load, thus allowing for an efficient usage of computational resources.

Based on the above observations, in this paper we combine EE and edge computing, and envision a novel solution to the problem of DNN task offloading that is particularly suitable for resource-limited mobile systems. To the best of our knowledge, our work is the first one that proposes such an approach and designs an optimal, low-complexity, policy that can easily and effectively adapt to the changes in the application requirements and the system conditions.

More specifically, our main contributions are as follows.

1) We develop a MDP approach to find the best inference execution strategy in an edge computing scenario. Different from [9]–[11], we consider that while the CAV uses a simple DNN with no EE, the edge server can rely on EE, which leads to the design of a flexible and effective approach.

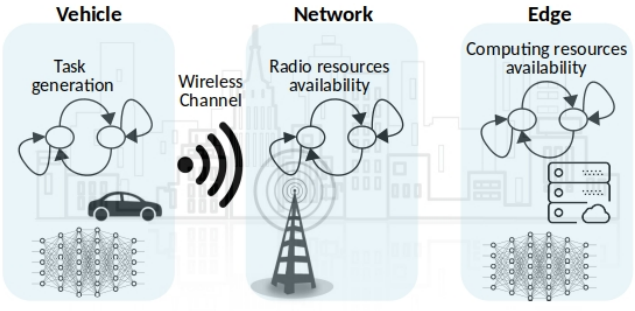

Fig. 1. Reference system scenario and main components of the system model.

2) We then formulate an optimization problem to identify the inference execution strategy that maximizes accuracy, while fulfilling the application-level requirements in terms of inference time and task dropping rate.
3) Finally, we evaluate the obtained optimal policy under varying system settings and operational conditions, including propagation channel and levels of network congestion and edge load. The performance of our policy is also compared to an optimal policy without EE, showing that EE achieves up to 11% performance gain.

The rest of the paper is organized as follows. Sec. II introduces the system model, while Sec. III gives its functional description. In Sec. IV, we formulate and solve the constrained accuracy maximization problem, presenting our optimal, low-complexity, offloading policy. Sec. V shows some numerical results, highlighting the benefits of combining early exiting with edge computing. Finally, Sec. VI draws our conclusions and sketches future research directions.

II. System Model

We consider the system scenario depicted in Fig. 1, which consists of a CAV that has to perform DNN-based inference tasks, and an edge server connected to a base station. The CAV can either perform the tasks locally or offload them to the edge server. We assume that the edge server uses a dynamic neural model equipped with EE, whereas the CAV uses simpler DNN models without branches due to its more limited computing capabilities. Also, time is assumed to be slotted; we denote by the time variable t the time interval $[tT, (t+1)T)$, where T is the slot duration. For compliance with the subframe duration in both 4G and 5G radio interfaces, we set T=1 ms.

All other system aspects, concerning task generation and handling as well as data transfer over the radio interface and the network model, are described below.

A. Task generation

The CAV stores tasks to be executed in a queue with finite capacity $Q_{\max}$; let $q(t)$ denote the state of the queue, i.e., the amount of backlogged tasks, at time slot t. The evolution of $q(t)$ is described as

$$q(t+1) = \min\{\max\{q(t) - e(t), 0\} + i(t), Q_{\max}\}, \quad (1)$$

where $e(t)$ is a binary flag representing the execution of a task at slot t and $i(t)$ is the number of incoming tasks at t. When the queue reaches its maximum capacity $Q_{\max}$, new incoming tasks are discarded.

Tasks are generated at the CAV depending on the running application state, $v(t)\in\{v_0, v_1\}$, where v_1 denotes the application active state, in which new tasks are generated, whereas v_0 represents the application idle state where no tasks are generated. For simplicity, we assume that only a single task can be generated in each time slot, then we define the task generation probabilities in each state as $p(i(t)=0|v(t)=v_0)=1$ and $p(i(t)=1|v(t)=v_1)=1$, where $p(\cdot)$ indicates the probability of an event.

The time evolution of $v(t)$ can be described through a two-state Markov chain with transition probability matrix $\mathbf{P_V} = \begin{pmatrix} \alpha & 1-\alpha \\ 1-\beta & \beta \end{pmatrix}$, where α=$p(v(t+1)=v_0|v(t)=v_0)$ and β=$p(v(t+1)=v_1|v(t)=v_1)$. In particular, the transition probabilities are expressed in terms of the burstiness b_v (i.e., mean sojourn time in state v_1) and the steady state probability ϕ_v of the application being active (i.e., in state v_1). Then,

$$1 - \alpha = \frac{\phi_v}{b_v(1-\phi_v)} \quad (2)$$

$$1 - \beta = \frac{1}{b_v}. \quad (3)$$

A task removal from the queue occurs when its execution is terminated. By denoting with $c_E(t)$ the remaining time slots required for executing a task, we can express $e(t)$ as

$$e(t) = \begin{cases} 1 & \text{if } c_E(t) = 0 \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

B. Wireless Channel model

The wireless link between the CAV and the edge server is assumed to be affected by Rayleigh fading. To obtain a finite-state representation of the channel gain envelope $r(t)$, we rely on a Markov chain model of the fading process, where the chain states are obtained by quantizing $r(t)$ through a finite set of intervals.

The intervals are identified by a set of thresholds that can be obtained by imposing the steady state probabilities π_i, i=$1, \ldots, N_r$, of the channel states, being N_r the desired number of states. Following [13], we assume a uniform steady state distribution, that is π_i=$1/N_r$, $\forall i$, and we determine the thresholds Γ_i and Γ_{i+1} by solving

$$\int_{\Gamma_i}^{\Gamma_{i+1}} f_r(r)dr = 1/N_r, \quad (5)$$

where $f_r(r)$ is the Rayleigh probability density function (pdf). Solving Eq. (5) yields

$$\Gamma_i = \sqrt{-2\sigma^2 \ln\left(1 - \frac{i}{N_r}\right)}. \quad (6)$$

Given the partitioning scheme in (6), we say that the chain is in state r_i if $r(t) \in [\Gamma_i, \Gamma_{i+1})$ and the channel output is given as the midpoint of the interval.

To define the transition probability matrix $\mathbf{P_R}=(p_{R,ij})_{i,j=1}^{N_r}$, we follow the approach in [13]. In particular, the transition probabilities between each pair of states are obtained through a series expansion of the bivariate Rayleigh cumulative distribution function (CDF), which allows capturing the second-order properties of the fading process. In particular, for the autocorrelation function of the underlying complex Gaussian fading process we adopt the well known Clarke's model, where the correlation between two consecutive fading samples is given as $J_0(2\pi f_D T)$, with f_D being the maximum Doppler shift and T the observation time between two samples, which is assumed to be equal to the time slot duration.

Note that the channel state affects the opportunity of performing tasks offloading. Indeed, assuming for simplicity a fixed rate R_c supported by a specific Modulation and Coding Scheme (MCS), to guarantee reliable data transfer, the experienced Signal-to-Noise Ratio (SNR) should be above a given minimum level. If not, task offloading cannot take place.

C. Radio resources allocation

To offload a task, the CAV has to request the base station for the necessary radio resources. Let $n(t)\in\{n_0, n_1\}$ be a binary flag indicating the availability of radio resources at slot t: if $n(t)=n_0$, no radio resources are readily available; otherwise, if $n(t)=n_1$, a radio resource can be allocated. Assuming the CAV demands radio resources and the system is in n_0, a slot counter $c_N(t)$ is initialized to T_g. When $c_N(t)$ reaches 0, then $n(t)$ moves to state n_1 with probability p_g, corresponding to a successful resource assignment in that specific time slot. Conversely, when $n(t)$ stays in n_0 (which happens with probability $1-p_g$), no offload can take place due to the lack of available resources. Then, the above procedure may repeat. Notice that, when in n_1, the user can offload the task to the server only if the SNR is above the minimum threshold for the previously selected MCS, otherwise the offload fails.

D. Inference time and accuracy

The execution time of an inference task depends on whether computation occurs at the CAV or at the edge. A local task execution has a duration T_l, due solely to the computation at the CAV; on the contrary, remote execution also involves offloading the task, which increases the overall latency.

Furthermore, since the edge server relies on EE, it can choose among several exits of the DNN, each exit e ($e=1,\dots,M$) yielding a different level of accuracy, A_e, and execution time, T_e. Notice that we have: $T_1<T_2<\dots<T_M$ and $A_1<A_2<\dots<A_M$. Additionally, the execution time at the edge is affected by the current computational load the edge server is experiencing. To account for the case where the edge server might not have ready-to-use computing capabilities, we define the server state $s(t)\in\{s_0, s_1\}$, with s_0 and s_1 representing idle and busy computational resources, respectively. An additive random delay $d(t)$ is then associated with each state and characterized by the probabilities $p(d(t)=0|s(t)=s_0)=1$ and $p(d(t)=\tau|s(t)=s_1)=p(\tau)$, where $\tau\in\{0,1,\dots,\tau_{\max}\}$ and $p(\tau)$ is assumed to be a truncated decreasing geometric probability mass function (pmf).

The dynamic of $s(t)$ is modelled through a Markov chain with transition probability matrix $\mathbf{P_S}=\left(\begin{smallmatrix}\gamma & 1-\gamma\\ 1-\phi & \phi\end{smallmatrix}\right)$, where $\gamma=p(s(t+1)=s_0|s(t)=s_0)$ and $\phi=p(s(t+1)=s_1|s(t)=s_1)$. Similarly to the task generation model, transition probabilities can be expressed as a function of the burstiness b_s and the steady state distribution ϕ_s of state s_1.

Depending on the selected exit and the server state, the remote execution time is thus given by:

$$T_{\text{edge}}(t) = T_e + d(t). \tag{7}$$

III. Actions Set and System Behaviour

The overall system state $x(t)$ at time slot t is defined as

$$x(t) = (q(t), v(t), s(t), r(t), n(t), c_E(t), c_N(t)). \tag{8}$$

The time evolution of $x(t)$ is driven by uncontrollable system dynamics (such as the channel state, task generation, computational resource availability, etc.) as well as the undertaken action a_e. We denote with $\mathcal{A}=\{a_e\}_{e=0}^{M+2}$ the set of possible actions. In particular, action a_e, with $e=1,\dots,M$, represents the execution up to exit e deferred to the edge; $a_{M+1}=a_l$ indicates a local execution of the task; $a_{M+2}=a_g$ defines the action of making a request for a radio resource; finally, $a_0=a_w$ is the action corresponding to the CAV doing nothing. According to the defined action set, we can distinguish the following relevant cases:

a) $c_E(t)=0$, $c_N(t)=0$, $n(t)=n_0$*:* in this case the system is in idle state and a task waiting in the queue can be executed. However, since $n(t)=n_0$, there are no radio resources available and only local computing, resource request or waiting are admissible actions. By selecting action a_l, the task is executed locally and $c_E(t)$ is initialized to T_l. Denoting with $p(x'|x,a)=p(x(t+1)=x'|x(t)=x, a(t)=a)$ the probability of moving into state x' from state x by taking action a, we can write the possible state transitions as follows:

$$p((q+1, v_1, s_1, r_j, n_0, T_l, 0)|(q, v_0, s_1, r_i, n_0, 0, 0), a_l) = (1-\alpha)\phi p_{R,ij}, \tag{9}$$

with similar transition probabilities that can be written in both the above cases for different values of states $v(t)$ and $s(t)$.

Instead, a radio resource can be requested by selecting action a_g, which allows transitions as

$$p((q, v_0, s_1, r_j, n_0, 0, T_g)|(q, v_0, s_1, r_i, n_0, 0, 0), a_g) = \alpha\phi p_{R,ij}. \tag{10}$$

b) $c_E(t)=0$, $c_N(t)=0$, $n(t)=n_1$*:* in this case also remote computing is possible if the SNR is high enough for the selected MCS. Then an action a_e, with $e=1,\dots,M$, can be selected, thus yielding transition probabilities of the type

$$p((q, v_0, s_0, r_j, n_0, T_e, 0)|(q, v_0, s_0, r_i, n_1, 0, 0), a_e) = \alpha\gamma p_{R,ij}. \tag{11}$$

Moreover, if the server is busy (i.e., $s(t)=s_1$), an additional random delay is required for obtaining computing resources, which yields the following state transitions:

$$p((q, v_0, s_1, r_j, n_0, T_e+\tau, 0)|(q, v_0, s_1, r_i, n_1, 0, 0), a_e) = \alpha\phi p(\tau) p_{R,ij}. \quad (12)$$

c) $c_E>0$ or $c_N>0$: in such situations, the system is either executing a task, locally or remotely, or counting the time spent for requesting a radio resource; it follows that the only possible action is a_w. Additionally, at any time slot either $c_E(t)$ or $c_N(t)$ decrease by 1, thus yielding the transition probabilities

$$p((q, v_0, s_0, r_j, n_0, N-1, 0)|(q, v_0, s_0, r_i, n_0, N, 0), a_w) = \alpha\gamma p_{R,ij}. \quad (13)$$

When $c_E(t)=1$, according to the behavior of the queue modeled in Eq. (1), we have

$$p((q-1, v_0, s_0, r_j, n_0, 0, 0)|(q, v_0, s_0, r_i, n_0, 1, 0), a_w) = \alpha\gamma p_{R,ij}. \quad (14)$$

Instead, when $c_N(t)=1$, a radio resource could not be assigned in the next time slot with probability $1-p_g$, giving the following transitions:

$$p((q, v_0, s_1, r_j, n_0, 0, 0)|(q, v_0, s_1, r_i, n_0, 0, 1), a_w) = \alpha\phi(1-p_g) p_{R,ij}, \quad (15)$$

or a radio resource can be assigned with probability p_g allowing the following transitions:

$$p((q, v_0, s_1, r_j, n_1, 0, 0)|(q, v_0, s_1, r_i, n_0, 0, 1), a_w) = \alpha\phi p_g p_{R,ij}. \quad (16)$$

IV. Optimal Execution Policy

We now derive an optimal action selection strategy that maximizes the average inference accuracy while meeting the constraints on the average execution time and the task discarding rate. To do so, we formulate an optimization problem by exploiting the above MDP framework, and solve it through Linear Program (LP) mapping – a well-established and effective solution approach [14].

Without loss of optimality we limit our search within the class of past-independent randomized policies $\mu\colon \mathcal{X}\times\mathcal{A}\to[0,1]$, where $\mathcal{X}$ is the set of possible system states and $\mu(a|x)=p(a(t)=a|x(t)=x)$ is the probability of taking action $a\in\mathcal{A}$ when in state $x\in\mathcal{X}$. Note that the distribution of the action is independent of the past history of the system, so that the state-action sample paths are Markovian with transition probabilities $p(x'|x,a)$. If the Markov process is ergodic (i.e., recurrent aperiodic for any control policy [15]), then time averages converge to state-space averages. Thus, by denoting with $A(x,a)$ the accuracy generated by taking action a when being in state x, we can express the expected accuracy as

$$\overline{A} = \sum_{x\in\mathcal{X}}\sum_{a\in\mathcal{A}} A(x,a)\pi_\mu(x,a), \quad (17)$$

where $\pi_\mu(x,a)$ is the joint steady state distribution of the states and the actions, and $A(x,a)=0$ for actions that do not correspond to the execution of an inference (i.e., a_w and a_g).

Next, following [10], we define the average execution time $\overline{T}_{EX}$ as the average total queuing time, which encompasses the execution time along with idleness epochs and the time required for radio resource assignment. Formally, $\overline{T}_{EX}$ is given as the ratio between the fraction of time a task remains in the queue and the fraction of executed tasks, that is

$$\overline{T}_{EX} = \frac{\sum_{x\in\mathcal{X}}\sum_{a\in\mathcal{A}} r_1(x,a)\pi_\mu(x,a)}{\sum_{x\in\mathcal{X}}\sum_{a\in\mathcal{A}} r_2(x,a)\pi_\mu(x,a)}, \quad (18)$$

where

$$r_1(x,a) = \begin{cases} 1 & \text{if } a = a_w, a_g, \\ 0 & \text{otherwise,} \end{cases} \quad (19)$$

is an indicator function for the count of time slots in which execution-related actions are not chosen. By recalling that waiting (i.e., a_w) is the only admissible choice during both running executions (i.e., $c_E(t)>0$) and radio resource negotiation (i.e., $c_N(t)>0$), then $r_1(x,a)$ inherently accounts for the time spent in executing a task (either locally or remotely) and the time spent in obtaining radio resources. Similarly,

$$r_2(x,a) = \begin{cases} 1 & \text{if } a \neq a_w, a_g, \\ 0 & \text{otherwise,} \end{cases} \quad (20)$$

is a flag indicating whether a task is executed or not.

The optimal randomized policy for maximizing the average accuracy can be obtained by solving the following LP:

$$\underset{\mu}{\text{argmax}} \quad \sum_{x\in\mathcal{X}}\sum_{a\in\mathcal{A}} A(x,a)\pi_\mu(x,a) \quad (21)$$

$$\text{s.t.} \quad \frac{\sum_{x\in\mathcal{X}}\sum_{a\in\mathcal{A}} r_1(x,a)\pi_\mu(x,a)}{\sum_{x\in\mathcal{X}}\sum_{a\in\mathcal{A}} r_2(x,a)\pi_\mu(x,a)} \leq t_c \quad (22)$$

$$1-\frac{\sum_{x\in\mathcal{X}}\sum_{a\in\mathcal{A}} r_2(x,a)\pi_\mu(x,a)}{\phi_v} \leq d_r \quad (23)$$

$$\sum_{a\in\mathcal{A}} \pi_\mu(x',a) - \sum_{x\in\mathcal{X}}\sum_{a\in\mathcal{A}} \pi_\mu(x,a)p(x'|x,a)=0, \forall x' \quad (24)$$

$$\sum_{x\in\mathcal{X}}\sum_{a\in\mathcal{A}} \pi_\mu(x,a)=1 \quad (25)$$

that yields the optimal steady state distribution $\pi^\star_\mu(a,x)$. Constraint (22) states that the average execution time cannot exceed a maximum tolerable value t_c. Constraint (23) imposes that the task dropping probability is lower than the target value d_r. Indeed, the ratio between the probability of executing a task and the probability of generating a task ϕ_v represents the fraction of tasks generated and effectively executed, while the left-hand side term in (23) represents the probability of

TABLE I
ACCURACY AND EXECUTION TIMES ASSOCIATED WITH THE DIFFERENT ACTIONS

	a_1	a_2	a_3	a_l
A_e	88.9	92.9	93.84	72.4
T_e [ms]	2	3	4	10

TABLE II
SIMULATION PARAMETERS

Parameter	Value	Parameter	Value
$Q_{\max}$	10	T_g [ms]	5
b_s	5	t_c [ms]	30
b_v	5	d_r	0.1
$\tau_{\max}$ [ms]	15	N_r	4

not executing a generated task. Finally, constraints (24)–(25) come from the mapping of the MDP into the LP [14], where (25) represents a flow-balance constraint.

Finally, the optimal policy $\mu^\star(a|x)$ can be obtained as

$$\mu^\star(a|x) = \frac{\pi^\star_\mu(a,x)}{\sum_{a\in\mathcal{A}} \pi^\star_\mu(a,x)}, \tag{26}$$

that is, as the optimal probability of choosing action a, conditioned on the system being in state x.

V. NUMERICAL RESULTS

In this section, we evaluate the system performance when our optimal executing strategy is applied. The values of inference execution times and accuracy are presented in Tab. I: for the early exiting model, they refer to a modified ResNet56 with 3 exits trained on the CIFAR-10 dataset [16], while, for the local inference, they refer to the MobileNetV2 model executed on a mobile device [17]. All relevant system parameters considered for results derivation are listed in Tab. II and are kept fixed, unless otherwise stated.

To find the set of channel states limiting the offloading procedure for the different MCS, we considered the SNR values reported in [18]. In particular, we fixed the transmission rate of the CAV, the transmit power, and the noise power. Then, depending on the value of the channel envelope power associated with each channel state, we identified a set of SNRs and identified which of the channel states allow for a sufficient MCS that can honor the rate constraint R_c.

We also remark that, since a non zero value of accuracy is obtained only if the task is executed, we normalize the maximum accuracy obtained under the optimal policy with respect to the fraction of tasks for which an inference is performed. We do so by defining

$$\overline{A} = \frac{\sum_{x\in\mathcal{X}}\sum_{a\in\mathcal{A}} A(x,a)\pi^\star_\mu(x,a)}{\sum_{x\in\mathcal{X}}\sum_{a\in\mathcal{A}} r_2(x,a)\pi^\star_\mu(x,a)}, \tag{27}$$

which is considered as the reference metric for performance assessment.

Fig. 2 shows the average accuracy as a function of the steady state probability of the application being in active state (ϕ_v) and for different values of the busy probability of the server (ϕ_s). Moreover, a comparison between the derived optimal policy and a benchmark scenario where the edge can only execute the full DNN (i.e., no EE is possible) is reported. In the latter case, we set the execution time and accuracy as in the third column of Tab. I. Referring to the EE-based policy, we can observe how the obtained accuracy matches the DNN highest performance for low values of ϕ_v. However, when the task generation rate increases (higher values of ϕ_v), the accuracy starts slightly decreasing, since the action policy favors earlier exits of the DNN model to avoid queue overflow and meet the constraint on the task dropping rate. Then, as ϕ_v further increases, the system starts avoiding radio resource requests by prioritizing local inference (hence, avoiding the time to offload the task), which causes additional accuracy drop. These effects are exacerbated when the server availability is exiguous (high values of ϕ_s), since remote task execution is affected by delayed assignment of the computing resources. Importantly, when the task generation rate at the CAV grows (high values of ϕ_v), the traditional edge computing approach with no EE favors local inference, while our approach exploits the earlier classifiers at the edge server, thus reaching higher accuracy. This effect is substantial when the edge availability rate is high (i.e., low values of ϕ_s), with our policy producing an accuracy gain of 11% if compared to the traditional edge computing scheme.

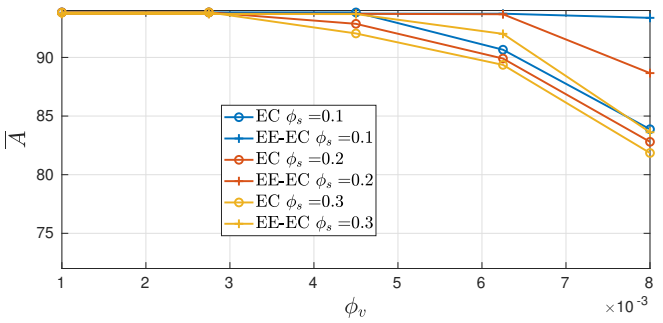

Fig. 2. Average accuracy as a function of ϕ_v, as obtained with Edge Computing with and without Early Exiting, and for various settings of ϕ_s. It is assumed $p_g = 0.8$, $f_D T = 0.01$ and $R_c = 15$ Mbps.

Next, Fig. 3 shows how the running application requirements affect the average accuracy. In particular, the plot reports the average accuracy as a function of the constraint on the average execution time t_c and for different values of the target maximum dropping rate d_r. As expected, by relaxing the application constraints, performance improves, since the optimal policy favours remote execution to maximize accuracy. Furthermore, the results highlight the flexibility of the early exiting paradigm in satisfying the application constraints: under stringent constraints, remote execution is still viable by exploiting the earlier classifiers.

Finally, Fig. 4 illustrates how the inference executing strategy reacts to different channel dynamics and network conditions. The plot reports the average accuracy as a function of the probability of obtaining a radio resource p_g, and for different regimes of channel correlation $f_D T$. Two different MCSs

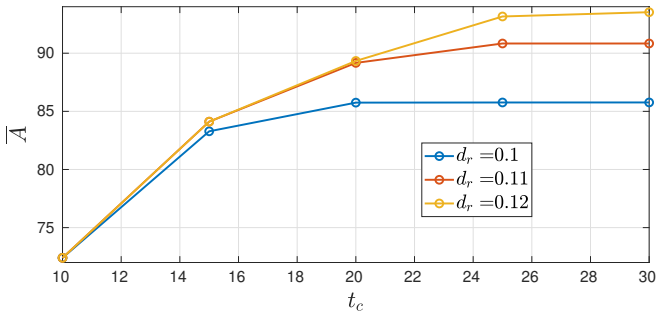

Fig. 3. Average accuracy as a function of t_c and for various settings of d_r. It is assumed $\phi_v = 5\text{e}-3$, $\phi_s = 0.8$, $p_g = 0.6$, $f_D T = 0.01$ and $R_c = 30$ Mbps.

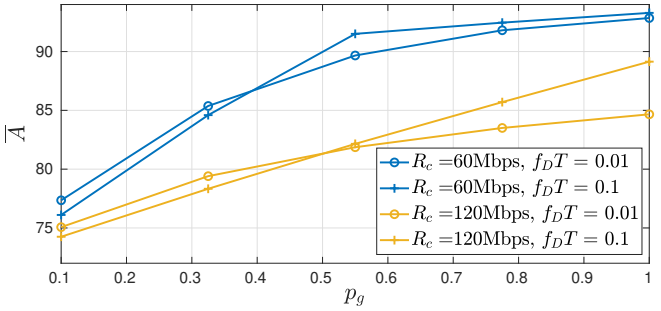

Fig. 4. Average accuracy as a function of p_g and for various settings of R_c, and $f_D T$. It is assumed ϕ_v=0.001 and ϕ_s=0.5.

(i.e., different rates R_c) are considered for the same channel partitioning scheme. As expected, when the probability of obtaining a radio resource is low, local execution is predominant: this allows the system to honor the target task dropping rate, at the cost of a lower accuracy value. Instead, as p_g increases, the CAV offloads the task more frequently, so as to maximize the accuracy. We can also observe that transmitting at higher rate requires a higher minimum SNR and this may prevent successful offloading, hence leading to a longer waiting time before a task can be actually dispatched to the server. In this case, the action policy is prone to opt for local computing thus reducing the attained accuracy. This highlights the importance of carefully adapting the MCS according to the wireless link conditions, since channel impairments may induce severe degradation of the attained accuracy if the transmission rate is selected regardless of the propagation phenomena. Finally, looking at the impact of the channel dynamics, one can observe how a higher accuracy level can be achieved when the channel exhibits low correlation, since in this case the channel tends to remain in unfavorable states for a shorter time. This effect is even more evident with higher rate constraints.

VI. Conclusions

We considered mobile devices that have to execute DNN tasks, either locally or by offloading them to the network edge. By leveraging both the Early Exiting and the Edge Computing paradigms, and modeling their interaction through a Markov Decision Process, we derived a low-complexity offloading policy that maximizes the tasks accuracy. We investigated the benefits of our policy in a scenario where automated vehicles are connected with an edge server. Results show that early exiting provides high flexibility in adapting the performance to different operational conditions and application-specific requirements. In particular, the obtained accuracy level can be tailored to the task generation rate at the vehicles, to the computing capabilities available at the server, and to the channel conditions and network load, thus demonstrating that Early Exiting can be a powerful tool to optimally orchestrate DNN tasks in dynamic scenarios.

As future development, we plan to extend the proposed model to capture complex multi-CAV dynamics impacting the server and network load, as well as the interference experienced by the CAV users. Also, we will account for CAVs generating tasks with different service requirements and levels of priority.

References

[1] 3GPP, "Service requirements for V2X services ," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 22.185, 2016, v.14.0.0.

[2] 5GAA, "C-V2X Use Cases: Methodology, Examples and Service Level Requirements," 5G Automotive Association, White Paper, 2019.

[3] ——, "C-V2X Use Cases Volume II: Examples and Service Level Requirements," 5G Automotive Association, White Paper, 2020.

[4] E. Cinque, F. Valentini, A. Persia, S. Chiocchio, F. Santucci, and M. Pratesi, "V2X Communication Technologies and Service Requirements for Connected and Autonomous Driving," in *2020 AEIT International Conference of Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE)*, 2020, pp. 1–6.

[5] L. Andreone, R. Brignolo, S. Damiani, G. Sommariva, G. Vivo, and M. Stefano, "SAFESPOT Final Report," , Tech. Rep., 2010, v.1.0.

[6] C. Yang, Y. Liu, X. Chen, W. Zhong, and S. Xie, "Efficient Mobility-Aware Task Offloading for Vehicular Edge Computing Networks," *IEEE Access*, 2019.

[7] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task Offloading in Vehicular Edge Computing Networks: A Load-Balancing Solution," *IEEE Trans. on Veh. Tech.*, 2020.

[8] B. Lv, C. Yang, X. Chen, Z. Yao, and J. Yang, "Task Offloading and Serving Handover of Vehicular Edge Computing Networks Based on Trajectory Prediction," *IEEE Access*, 2021.

[9] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *IEEE ISIT*, 2016.

[10] D. Callegaro, Y. Matsubara, and M. Levorato, "Optimal Task Allocation for Time-Varying Edge Computing Systems with Split DNNs," in *IEEE GLOBECOM*, 2020.

[11] D. Callegaro and M. Levorato, "Optimal Edge Computing for Infrastructure-Assisted UAV Systems," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 2, pp. 1782–1792, 2021.

[12] Y. Matsubara, M. Levorato, and F. Restuccia, "Split Computing and Early Exiting for Deep Learning Applications: Survey and Research Challenges," 2022.

[13] C. Tan and N. Beaulieu, "On first-order Markov modeling for the Rayleigh fading channel," *IEEE Trans. on Comm.*, 2000.

[14] K. W. Ross, "Randomized and Past-Dependent Policies for Markov Decision Processes with Multiple Constraints," *Oper. Res.*, 1989. [Online]. Available: https://doi.org/10.1287/opre.37.3.474

[15] S. Meyn, R. L. Tweedie, and P. W. Glynn, *Markov Chains and Stochastic Stability*, 2nd ed., ser. Cambridge Mathematical Library. Cambridge University Press, 2009.

[16] F. Ilhan, L. Liu, K.-H. Chow, W. Wei, Y. Wu, M. Lee, R. Kompella, H. Latapie, and G. Liu, "EENet: Learning to Early Exit for Adaptive Inference," 2023.

[17] Q. Zhang, X. Che, Y. Chen, X. Ma, M. Xu, S. Dustdar, X. Liu, and S. Wang, "A Comprehensive Deep Learning Library Benchmark and Optimal Library Selection," *IEEE Trans. on Mob. Comp.*, 2023.

[18] J. Fan, Q. Yin, G. Y. Li, B. Peng, and X. Zhu, "MCS Selection for Throughput Improvement in Downlink LTE Systems," in *ICCCN*, 2011.