Edge-Cloud Switched Low-Carbon Image Segmentation for Autonomous Vehicles

Siyuan Zhou, Duc Van Le, *Senior Member, IEEE*, and Rui Tan, *Senior Member, IEEE*

Abstract—Existing autonomous vehicles (AVs) utilize neither cloud computing for execution of their deep learning-based driving tasks due to the long vehicle-to-cloud communication latency, nor solar energy to offset the energy usage of car-borne computing. They are in general equipped with the resource-constrained edge computing devices which may be unable to execute the compute-intensive deep learning models in real time. The increasing data transmission speed of the commercial mobile networks sheds light upon the feasibility of using the cloud computing for autonomous driving, which is demonstrated by our city-scale real-world measurements over the fifth generation (5G) mobile networks. Moroever, the cost and form factor declines of solar harvesting systems make the quest of integrating them with AVs for improving carbon efficiency more promising. In this paper, we present the design and implementation of ECSeg, an edge-cloud switched low-carbon image segmentation system for AVs equipped with roof-mounted solar panels. ECSeg dynamically switches between edge and cloud processing to execute semantic segmentation models in real time while aiming to decarbonize AV computing by maximizing the use of harvested solar energy. The switching decision is challenging due to the complex interdependencies among various factors, including dynamic wireless channel conditions, vehicle motion, visual scene changes, and available renewable energy. To tackle this, ECSeg employs deep reinforcement learning to learn an optimal switching policy. Extensive evaluations based on real-world experiments and trace-driven simulations demonstrate that ECSeg outperforms six baseline approaches, achieving 98.8% reduction in computing's carbon emission, while maintaining high image segmentation accuracy, compared with our earlier design without integrating the solar panel.

Index Terms—Image segmentation, cloud-assisted system, autonomous driving

1 Introduction

Autonomous vehicles (AVs) have substantial potential to mitigate traffic congestion, enhance road safety, and curtail carbon emissions. Deep learning (DL) has been increasingly employed for various driving tasks of the AVs. For example, the DL models [1]–[3] can be used for the vehicles to understand their visual driving scenes correctly, facilitating the safe driving navigation and accurate collision avoidance. To avoid the long latency and privacy issues of data transmission, the commercial AV platforms (e.g., Apollo [4]) are often equipped with the resource-limited edge computing devices to directly execute the DL-based autonomous driving tasks on the vehicles. Meanwhile, the execution of deep models often imposes high demand on computing resources. Thus, current AV design strategies adopt customized lightweight, on-board deep models [5] which can be executed by the edge devices in real time to achieve autonomous driving. This design choice compromises the accuracy of the deep models. A possible approach to address this problem is to increase the computing capabilities of the edge devices which allow implementation of complex deep models with high accuracy. However, the powerful computing devices are energy-intensive, which reduces the vehicle's battery system lifetime. It is also challenging for the vehicle's heat dissipation system to handle a high amount of heat dissipated by the energy-intensive computing devices [6].

In this paper, we investigate the feasibility of using the cloud computing for executing the DL-based autonomous driving tasks. In particular, the cloud servers can provide the AVs with sufficient computing capabilities without power limitation and heat dissipation issues. To address the privacy concerns, the data encryption approaches [7] can be implemented to secure the AV's sensitive information before transmitting the data to the cloud server. Meanwhile, the increasing data transmission speed of the commercial mobile networks sheds light upon the opportunity of significantly reducing the vehicle-to-cloud communication latency. Our city-scale real-world measurements show that the fifth generation (5G) mobile network can provide an average round-trip time (RTT) latency of 100 ms for transmitting the 500-KB data packets from the moving vehicles to the remote cloud servers. Thus, we conjecture that the clouds connected via a mobile network can bring benefits to the AVs. Semantic segmentation is more computationally intensive than other vision-based tasks in autonomous driving. For example, kernelized correlation filters (KCF) for object tracking can reach approximately 170 FPS on a CPU [8], and object detection models such as YOLOv4 can achieve 45.5 FPS on an edge GPU [9]. These numbers suggest that detection and tracking tasks can generally be executed efficiently on the edge device with satisfactory accuracy. By contrast, semantic segmentation imposes substantially higher computational demands, which makes it a natural focus for studying the benefits of cloud computing.

The AVs generally require an additional amount of energy for powering the sensing, communication, and computing devices in their autonomous driving subsystem. A study in [10] reported that the autonomous driving subsystem can increase the vehicle's greenhouse gas emissions by 3%-20% due to the increases in power use, drag, weight, and data transmission. Thus, a low-carbon AV design is highly

Siyuan Zhou, Duc Van Le, and Rui Tan are with College of Computing and Data Science, Nanyang Technological University, Singapore.

desirable and important to the global transition to carbon neutrality. In this paper, we also investigate the feasibility of a low-carbon AV design in which the solar panels are mounted on the AV's roof. The solar panels convert sunlight irradiance into electric energy for powering the computation of AV.

To achieve the above two research objectives, in this paper, we design an edge-cloud switched low-carbon image segmentation system, called *ECSeg* which aims to provide high-accuracy segmentation results of the vehicle's visual driving scenes in real time while minimizing the computation carbon emissions. Specifically, ECSeg switches between the following two processing options to obtain the segmentation result of each image frame captured by the AV's camera before a certain deadline (e.g., the time when the next image frame is captured). First, *edge processing option* executes a lightweight convolutional neural network (CNN) model locally on the AV's edge computing device to obtain the segmentation results of the image frames. Second, *cloud processing option* compresses the raw images and transmits them to a cloud server via the mobile network. Then, the cloud server executes an advanced CNN model to process the images and sends the results back to the vehicle. Moreover, ECSeg adopts a hybrid energy system in which the solar energy harvested by the solar panels is mainly used to power the edge and cloud processing options. When the solar energy is not available, it will use the energy drawn from the AV's main battery.

Among the above two processing options of ECSeg, the edge processing option can always provide the segmentation result of the captured image frame in real time. However, it has low segmentation accuracy due to the use of the lightweight CNN model. With the execution of the advanced CNN model, the cloud processing option achieve higher segmentation accuracy. However, due to the cloud data transmission latency, the vehicle may not always obtain the segmentation result of a transmitted image frame before the image's processing deadline. To deal with this latency issue, the cloud processing option allows the vehicle to only spend a certain time period to wait for the cloud result of the current image frame (i.e., current driving scene). We define the *source frame* by the previous frames whose cloud segmentation results have already arrived at the vehicle. If the vehicle does not receive the cloud result of the current frame after the waiting period, it uses the segmentation result of the source frame as the basis of an extrapolation process to estimate the result of the current frame. For the extrapolation, we develop an optical flow-based approach to propagate the segmentation result of the source frame to the current frame. We adopt a metric, called delay-mitigated mean intersection over union (mIoU) to assess the accuracy of the image segmentation results obtained by our propagation approach.

The delay-mitigated mIoU may be affected by changes in the visual content between the source and current frames. As the vehicle moves, the condition of the mobile network connection may vary due to base station switching and radio signal blockage [11]. Poor mobile network connection can cause long vehicle-to-cloud communication latency. As a result, the source frame can be far away by multiple image intervals from the current frame, leading to significant content changes between these two frames. Such changes reduce the delay-mitigated mIoU of the current frame. For example, when a new object that did not appear in the source frame enters the current frame, the propagation approach fails to obtain the segmentation result of this new object. Thus, the segmentation accuracy of the cloud processing option can be lower than that of the edge processing option.

Moreover, the edge and cloud processing options also have different energy patterns during their execution. Specifically, the edge processing option uses a low and stable amount of energy over time, while the cloud processing option uses a higher and dynamic amount of energy which is generally associated with increased communication latency. When the solar energy is available, the use of the cloud processing option should be maximized such that the high segmentation accuracy can be achieved without increasing the carbon emissions. Otherwise, the edge processing option should be used to minimize the use of non-renewable energy from the AV's main battery. Therefore, ECSeg needs to dynamically switch between the edge and cloud processing options so that image segmentation accuracy can be maximized, while minimizing the use of non-renewable energy to reduce carbon emissions.

However, switching decision-making is challenging due to the intricate interdependencies among various factors, including the dynamic mobile network channel condition, vehicle's movement, visual driving scene change, and the availability of the solar energy. To this end, we employ deep reinforcement learning (DRL) to learn a good long-term switching policy. However, the typical online training of the DRL agent requires time to converge. Before the convergence, the DRL agent may make low-quality decisions that undermine the performance and even safety of the vehicle. Moreover, there is a lack of image ground truth for determining the immediate rewards during the online training phase. To address these challenges, we adopt an offline training approach which utilizes real-world data traces to train the DRL agent. Finally, the trained agent is deployed for making the edge-cloud switching decisions at runtime.

We extensively evaluate ECSeg through real-world experiments and trace-driven simulations. We compare two variants of our proposed DRL-based switching approach with four baselines that incorporate efficient DL and knowledge distillation approaches from existing research. Without the use of solar energy, the first variant of our approach, called accuracy-priority ECSeg can improve the mIoU of the AV's image segmentation result up to 48.8%, compared with the baseline. Furthermore, the second variant, called low-carbon ECSeg can achieve a similar mIoU while being powered solely by the solar energy. Our main contributions can be summarized as follows:

- We conduct large-scale measurements to evaluate the benefits of using the cloud computing for AVs. Our design and experimental results may be useful for the development of other edge-cloud collaboration pipelines for AVs.
- We formulate the edge-cloud switching problem and adopt a DRL-based approach to learn a good switching policy for maximizing the AV's image segmentation ac-

curacy under two different AV designs in the presence and absence of the roof-mounted solar panels.
- We conduct extensive evaluations on real-world testbeds to assess the effectiveness of our proposed approach. Our results indicate that our approach outperforms the baseline methods.

The remainder of this paper is organized as follows. §2 reviews related work. §3 presents the measurement study. §4 overviews ECSeg's design. §5 describes the cloud processing option. §6 presents the DRL-based switching approach. §7 presents the evaluation results. §8 concludes this paper.

2 Related Work

In this section, we review the existing studies on the edge-based system, cloud-enabled sensing system, and RL-based control policy.

■ **Edge-based system:** Edge platforms have been adopted to execute the DL-based driving tasks in the AVs [2], [9], [12]. For instance, the study in [2] employs a CNN model with early downsampling and smaller convolutional filters to identify road lanes in real time on a vehicle platform. The authors in [9] adopt a sparse scaling factor algorithm to identify and prune less important channels and weights of CNN models for detecting vehicles and traffic signs, tailored for deployment on vehicle-mounted computing platforms. Additionally, the study in [12] introduces a usage-driven compression framework that identifies frequently activated subpaths based on user behavior, and selectively applies pruning, decomposition, and quantization through automated search to compress deep models in a demand-aware manner.

Regarding large language or vision-language models (LLMs/VLMs) such as SAM [13], InternVL [14], and DriveVLM [15], while these models demonstrate strong generalization across diverse tasks, they are not well-suited for real-time autonomous driving due to their high computational overhead and long latency. Furthermore, most VLMs are designed for high-level or interactive tasks, rather than for low-latency inference in mission-critical environments. The LLMs such as SAM are designed for interactive segmentation, where users provide manual pixel-level prompts to obtain instance masks. However, SAM does not generate semantic class labels and is not intended for class-specific segmentation. In contrast, our application performs segmentation that directly predicts class-specific results from the input image without any prompts. To incorporate SAM into the ECSeg framework, an additional prompt-generation module would be required to supply prior label information, which relies on strong prior knowledge. This fundamental difference makes SAM unsuitable for direct comparison or integration in our setting.

■ **Cloud-enabled sensing system:** Several existing cloud-enabled systems [16]–[18] have been employed to enhance the task performance on resource-limited devices. For example, the study by [16] introduces DCSB, an edge-cloud collaborative object detection framework that leverages a lightweight difficulty discriminator and regional sampling to adaptively offload hard cases to the cloud, thereby reducing bandwidth usage and latency while maintaining high detection accuracy. ACCUMO [18] uses a large CNN model on the cloud server and a lightweight local tracker to adjust cloud results for a real-time augmented reality task. Based on this, it employs model predictive control to determine which tasks to offload to the cloud for multiple tasks. It uses predicted accuracy to optimize overall accuracy. The above approaches always offload tasks to cloud servers, which can result in long latency when the wireless condition is poor. In this paper, we develop an edge-cloud switched method that shifts to edge processing when wireless condition is poor.

Similar to our work, DeepDecision [17] determines whether to process images on the cloud using a large object detection model or locally at the edge, based on network conditions and hardware limitations, with the main objective of maximizing the accuracy and frame rate. However, DeepDecision cannot be directly applied to our AV scenario because it does not consider the processing deadline for each frame. Additionally, it does not account for how changes in the driving scene affect the accuracy of cloud model. We compare ECSeg with ACCUMO and DeepDecision in §7.

■ **RL-based control policy:** Reinforcement learning (RL) has been applied to decide the task offloading in the edge/cloud computing systems [19]–[21]. The study [19] uses an online DRL agent to decide whether to offload tasks to the cloud server from multiple users, based on the users' queue and channel states, to optimize the age of information of the task. FEAT [20] uses DRL to decide task offloading to edge servers, based on task size, bandwidth, and hardware observations, reducing latency and mobile device energy consumption, while employing a steerer neural network to switch the DRL agent based on changes in observation distribution. EFCam [21] adopts the RL agent to adapt the wireless camera configuration to maintain the high performance of visual sensing. Our work shares a similar DRL-based control approach for a switching strategy to improve segmentation accuracy in AVs while considering the impacts of dynamic cloud latency and driving scene changes.

■ **Solar charging system:** Several studies have explored solar energy for system operation [22]–[24]. For instance, the study in [22] proposes a reinforcement learning-based system that adjusts duty-cycle parameters of sensors to optimize the use of solar energy by observing environmental information, ensuring the detection of critical events under uncertain energy availability. The study [23] proposes a path optimization algorithm that leverages street-view image segmentation and shadow modeling to estimate solar energy availability along roads, enabling the selection of energy-saving routes for solar-powered vehicles in urban environments. The study [24] proposes a solar vehicle design with low-cost flexible thin-film panels on all upward-facing surfaces, and models the effects of panel orientation and tilt to reduce grid load, transformer aging, and energy use. The above studies focus more on optimizing solar harvesting systems to maximize energy collection, while our work focuses more on adapting the system to reduce non-renewable energy usage under dynamic solar energy availability.

The green edge computing has been explored in several studies [25], [26]. For example, the study [25] proposes a joint power allocation, channel assignment, and offloading decision method to balance the trade-off between local

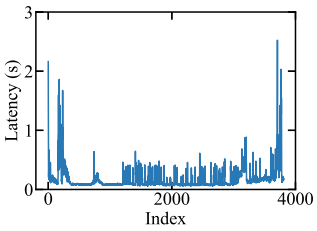


Fig. 1: Campus latency trace.

Tab. 1: Latency and accuracy of representative models.

Model	mIoU	Lat. (E/C)
ESPNet	0.505	30 / 139
PSP-R50	0.785	720 / 360
Intern-XL	0.836	1050 / 660

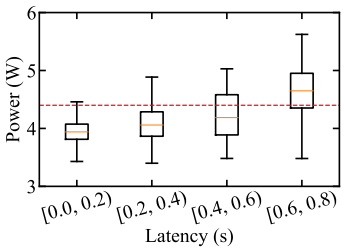


Fig. 2: Cloud option latency and power usage. The dashed line represents the power usage of the edge option.

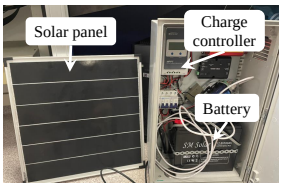


Fig. 3: Solar panel system.

processing delay and transmission power, which minimizes the total energy consumption of mobile devices in multi-cell multi-user mobile edge computing networks. In addition, [26] develops a Lyapunov optimization–based offloading framework that reduces long-term energy usage by balancing immediate task execution with resource preservation under uncertain energy harvesting and task arrivals. Differently, our work aims to minimize the reliance on non-renewable energy and maximize semantic segmentation performance, instead of optimizing for minimum energy consumption.

3 Measurement Study

3.1 Cloud Latency Profiling

We conduct a set of real-world experiments to measure the vehicle-to-cloud communication latency in various city areas. Specifically, we build a vehicular communication testbed which consists of a smartphone with a 5G communication service mounted on a vehicle. We use the Python socket library to implement a client-server application that allows a smartphone to transmit the data to a remote cloud server using the TCP/IP protocol via the 5G mobile network. In each experiment, the smartphone continuously transmits 500-KB application data packets to a Google's cloud server while traveling on a vehicle with the movement speed up to 50 km/h. Upon receiving each data packet, the cloud server sends a 1-byte acknowledgment packet back to the smartphone. We measure the round-trip time (RTT) of all transmitted data packets. The RTT of a data packet is defined as the time that it takes to receive the acknowledgment packet after the smartphone transmits the data packet. Fig. 1 presents the 1.5 hours RTT trace of the data packets in the university campus area. As the vehicle moves, the RTT fluctuates and reaches up to two seconds. This communication latency variability may be due to the switching of 5G base stations and signal blockage caused by buildings or other vehicles. More details are provided in Appendix A.1.

3.2 Segmentation Accuracy and Latency Profiling

We conduct profiling experiments to investigate the execution accuracy and latency of various deep learning-based image segmentation models on both the edge and cloud platforms. Specifically, we use an NVIDIA Jetson Orin [27] unit with a 32-GB DRAM as the AV's edge computing platform. A workstation equipped with an RTX 8000 GPU serves as the cloud platform. We implement eight image segmentation models with different sizes, including three lightweight models: ESPNet [5], as well as five large models: PSPNet-ResNet50 [28], and InternImage-XL [29]. We use the images sized $2048 \times 1024 \times 3$ pixels from the Cityscapes dataset [30] to evaluate the segmentation accuracy and latency of the implemented models.

Table 1 presents the mIoU (i.e., accuracy) and latency of the segmentation models executed on the edge and cloud platforms. The cloud latency is the end-to-end latency for image data transmission and model execution. We use the average RTT in the university campus area (c.f. §3.1) as the transmission latency of each image. The mIoU and latency numbers are the average values of the models over 1,000 testing images. From Table 1, the image segmentation accuracy and latency in general increase with the model complexity and size. Without the need of transmitting the images to the cloud, the edge platform can execute the lightweight models and provide the segmentation results with shorter latencies, compared with the cloud platform. On the other hand, the cloud platform can provide the segmentation results of large models with shorter end-to-end latency, even when including the cloud communication latency. More details are provided in Appendix A.2.

The above measurement results provide insights to guide the design of our edge-cloud switched image segmentation approach. Specifically, we can see that the cloud server can execute the large models to provide higher mIoU and shorter average latency, compared with the edge platform. However, the instantaneous cloud transmission latency fluctuates with vehicle movement, and its high value leads to reduction of mIoU at the AV. To address this issue, we propose a DRL-based approach for deciding the switching between edge and cloud processing options based on the dynamics of the real-time cloud latency and AV's driving scence. The detailed design of our proposed switching approach will be present in §6.

3.3 Power Consumption of Edge and Cloud Options

We conduct experiments to study the energy usage for executing the edge and cloud processing options of ECSeg. Then, we investigate the energy harvesting performance of our solar energy testbed. We use an NVIDIA Jetson Orin unit as the AV's edge platform in which the ESPNet is implemented as the image segmentation model at the edge. We conduct the experiment in which the edge platform uses its GPU to execute the ESPNet to obtain the segmentation results of 1,000 testing images at a frequency of 17 Hz over a period of 30 minutes. The total energy of the edge platform

during the experiment is measured by the Orin's built-in power meter. The measurement results indicate that the execution of the edge model uses an average power of 4.4 W which represents the average power use of the ECSeg's edge processing option.

The main energy consumers of cloud processing are result propagation and data transmission, with design details presented in §5. Specifically, the edge platform uses its CPU to execute the cloud result propagation process at a frequency of 17 Hz for 30 minutes. The measured average CPU power is 2.7 W over the 30-minute experiment period, which is considered as the average power usage for propagation. Moreover, to study the cloud data transmission energy usage, we conduct the experiment in which the edge platform uses the 5G dongle to continuously transmit the 500KB data packets to the cloud server over a period of 30 minutes while the vehicle is traveling at a speed of 50 km/h. We use the Monsoon Power Meter to measure the power usage traces of the 5G dongle as the data transmission power usage. The RTT of the transmitted packets is also measured as the cloud latency. At each time instant, the power usage of the cloud option is the sum of the 5G dongle's power usage and the average power usage of the cloud result propagation execution. Fig. 2 illustrates the distribution of the power usage of the cloud option under various cloud latency ranges. From Fig. 2, the cloud option power usage varies from 3.2 W to 5.8 W. Moreover, it generally increases with the cloud data transmission latency. Furthermore, from Fig. 2, we can see that the cloud option mostly uses less energy than the edge option when the cloud latency is less than 0.4 seconds. Thus, to save the energy, the AV's image segmentation should switch to the cloud option when the latency is low. Note that the cloud option also generates a higher segmentation accuracy due to the execution of the large image segmentation at the cloud server.

3.4 Solar Energy Harvesting Performance

We also build a solar energy harvesting testbed as shown in Fig. 3. Specifically, the testbed comprises monocrystalline solar panels and a maximum power point tracking charge controller that supports up to 10A of current to optimize power extraction and regulate the charging process. Additionally, a lithium iron phosphate battery is used to store the harvested solar energy. We test two types of solar panels with the sizes of 560x900 mm and 500x480 mm, which results in the peak harvesting powers of 100W and 50W, respectively.

The generation of solar energy is a carbon-free process, but the manufacturing and recycling of solar panels results in carbon emissions. Thus, a key concern of using the solar energy is that the embodied carbon emission of the solar panel production process may outweigh the carbon emission reduction achieved by the use of solar energy as replacement of the non-renewable energy. To investigate this concern, we analyze the carbon payback time (CPT) [31], which is the time required for a system's total carbon emissions C_{LCA} from manufacturing, transport, installation and recycling to be offset by its average annual carbon savings C_{ac}. The equation is given by CPT = $\frac{C_{\mathrm{LCA}}}{C_{\mathrm{ac}}}$. For example, in the U.S., the C_{LCA} of a monocrystalline silicon solar panel is measured at 127.3 kg CO_2 per square meter [32]. The C_{ac} for one square meter is 49.14 kg CO_2 [33], [34]. Using these values, the CPT is calculated as 2.59 years. A study in [35] reported that a solar panel can have a lifespan of over 25 years. With a CPT of 2.59 years, the use of a solar panel can contribute zero carbon emission over a period of about 22.41 years after an initial period of 2.59 years.

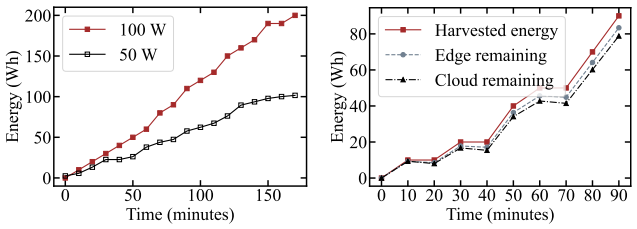
We conduct a set of experiments to investigate the energy harvesting rate of the solar panel in our testbed. Specifically, we deploy two solar panels on a sunny balcony with the direct sunlight at noon for 2.5 hours. During the experiment, the solar charge controller records the charged electricity level as the amount of harvested solar energy. Fig. 4(a) presents the cumulative energy harvested by the 100 W and 50 W solar panels over the experiment period. We can see that the 100 W panel can harvest an amount of about 200 Wh over 2.5 hours, which nearly double that of the 50 W panel. This result indicates the harvested energy proportionally increases with the solar panel size under strong sunlight conditions.

We also conduct experiments to investigate the energy harvesting performance of the solar panel mounted on the moving vehicle. Specifically, we mount the 100 W solar panel on the open trunk of a lorry which continuously travels through various environments such as tunnels, overpasses, and open streets at a movement speed up to 60 km/h for 90 minutes. Fig. 4(b) shows the cumulative energy harvested by the solar panel during the experiment and the remaining solar energy after executing the edge and cloud processing options over time. By the end of the journey, the total harvested energy reaches 85 Wh. The remaining solar energy for the edge and cloud options represents the available energy when AV tasks consistently execute either the edge or cloud processing option. Under a 100 W solar panel, the remaining solar energy for both edge and cloud options always stays above zero. This indicates that a 100 W solar panel, under good sunlight conditions, generates sufficient energy to support both edge and cloud processing options.

According to [36], the typical roof area of a passenger vehicle is about 2.5 m^2, which is sufficient to install five 100 W solar panels, yielding up to 500 W under bright sunlight. Our measurements show that a 100 W panel produces about 56 Wh per hour during vehicle operation. Thus, a 2.5 m^2 roof would produce roughly 280 Wh per hour.

3.5 Findings

Our measurement study provides three key observations. First, vehicle-to-cloud latency is highly variable, which can reach up to two seconds due to base station handover and environmental obstructions. Second, the profiling results of representative segmentation models show a trade-off between accuracy and latency: lightweight models can be executed efficiently on the edge with short latency at the cost of low accuracy, whereas large models provide significantly higher accuracy and achieve lower end-to-end latency when executed in the cloud despite transmission delay. Third, the cloud processing option incurs less power usage by the vehicle than the edge processing option when communication latency is low, and solar energy harvesting is



(a) The solar panel is deployed on a sunny balcony and harvest the energy for 2.5 hours at noon.

(b) The 100 W solar panel is mounted on a moving vehicle for 90 minutes.

Fig. 4: Solar panel performance.

sufficient to sustain both edge and cloud processing options in favorable conditions. These findings motivate our system design. The variability of communication latency and the accuracy–latency trade-off together show the need for an edge–cloud switching mechanism that adaptively chooses the processing venue between the edge and the cloud. Under dynamic solar energy harvesting conditions, the switching strategy also needs to consider energy availability to reduce power consumption and lower carbon emissions.

4 Design of ECSeg

In our work, we treat the vehicle as an edge device that both generates data and provides local computing capability [37], in contrast to a terminal, which mainly serves as a communication endpoint [38]. We consider an AV that uses a camera system to periodically capture its driving scene at every sampling interval, denoted by T. ECSeg is designed to execute deep learning-based semantic segmentation models to obtain the segmentation result of each image frame before the time when the next frame is captured. The image segmentation results provide pixel-level understanding of the AV's driving scenes which can be used as inputs for the autonomous driving pipelines [39], [40] to achieve accurate trajectory planning and safe navigation.

A solar panel is mounted on the AV's roof to continuously convert the sunlight irradiance into the electrical energy which is saved in a car-borne battery. The solar energy from the battery is used to power the edge and cloud processing options for executing the image segmentation tasks of the AV. As the amount of harvested solar energy varies over time and depends on the run-time weather condition, the AV will use the energy from its main battery for the segmentation task execution when the solar energy is not available. It is assumed that the main battery is pre-charged with the non-renewable energy sources. To reduce the carbon emission, our proposed low-carbon ECSeg employs DRL to dynamically switch between the edge and cloud processing options such that the usage of non-renewable energy is minimized while still maintaining high segmentation accuracy. In the ideal case, the AV's segmentation task is expected to achieve the zero carbon emission by operating with the solar energy only.

Fig. 5 overviews the design of low-carbon ECSeg which has two options: edge and cloud processing options for image segmentation as follows.

■ **Edge processing option:** This option executes a segmentation model locally to obtain the segmentation results

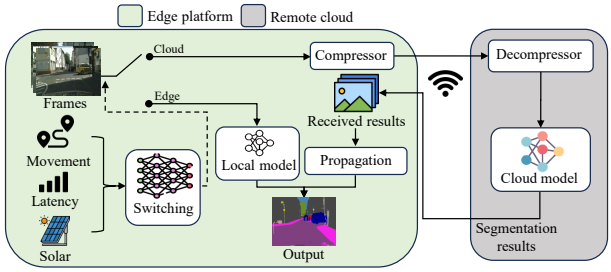


Fig. 5: Design overview of ECSeg.

of the captured image frames on the AV's edge platform. Given the limited computing resources of the edge platform, ECSeg employs a lightweight CNN-based image segmentation model as the local model such that the image segmentation result of each image can be always obtained before its deadline.

■ **Cloud processing option:** This option follows a streaming mode to continuously transmit the image frames from the vehicle to the cloud server via a mobile network. To reduce the communication overheads, we implement a JPEG approach to compress each image before transmitting it to the cloud server. Upon receiving the image data, the cloud server employs a JPEG decompressor to reconstruct the original image. Then, the cloud server executes the cloud model to process the reconstructed image. To achieve high image segmentation accuracy, an advanced CNN-based model with large size is implemented as the cloud model in the cloud server. Finally, the cloud image segmentation result is sent back to the vehicle.

Due to the long vehicle-to-cloud communication latency, the cloud segmentation result of an image frame may not arrive at the vehicle before the deadline. Thus, we also develop a propagation approach which uses the received cloud result of a previous frame as input to obtain the segmentation result for the current frame. The detailed design of our segmentation result propagation approach will be presented in §5.

■ **Switching:** The edge processing option can always provide the image segmentation result within the deadline. However, it suffers from low segmentation accuracy due to the use of the lightweight model. In contrast, the cloud processing option can execute an advanced model to achieve high accuracy, but has high latency uncertainty due to the dynamic vehicle-to-cloud communication latency. Due to the vehicle movement and poor wireless channel condition, the communication latency can be long, which causes the cloud segmentation results to become stale, decreasing the segmentation accuracy. To maximize the segmentation accuracy, at the edge platform, we implement a DRL-based controller which aims to dynamically switch between the edge and cloud processing options in response to changes of the communication latency, driving scene and solar energy. In §6, we formally formulate the switching problem and present our DRL-based solution.

5 Cloud Processing Option

5.1 Segmentation Result Propagation

Let $x_1, x_2, x_3, \ldots$ denote a sequence of images captured by the vehicle's camera system at every fixed interval of T. We

also denote t_i as the time when the image x_i is captured and then transmitted to the cloud server. The processing deadline of x_i is t_i+T (i.e., the arrival time of the next image x_{i+1}). Due to the dynamic vehicle-to-cloud communication latency, the vehicle may not receive the cloud segmentation result of x_i before the deadline $t_i + T$. To address this deadline missing issue, the vehicle only waits for the cloud result of x_i for a certain period, denoted by T_w, after t_i. If the vehicle does not receive the cloud result of x_i within the waiting period T_w, we employ a propagation approach to obtain the segmentation result of x_i based on the cloud segmentation result of the latest frame, denoted by x_k, among the previous frames of x_j $(j < i)$ whose cloud result has already arrived at the vehicle. Let T_p denote the fixed execution latency of the propagation. Then, the waiting period is calculated as $T_w = T - T_p$.

Now, we describe our approach to propagate the cloud segmentation result of x_k to the current frame x_i where $k < i$. The images x_k and x_i have the same size in terms of the number of pixels, denoted by M. Let $\mathcal{P}_k = \{P_{k,1}, \ldots, P_{k,M}\}$ and $\mathcal{P}_i = \{P_{i,1}, \ldots, P_{i,M}\}$ denote the sets of pixels of x_k and x_i, respectively. We adopt a computer vision technique, called optical flow which aims to determine the movement velocity of the pixels from x_k to x_i. Specifically, it derives the movement vectors, denoted by $F_{k\to i} = \{F_1, \ldots, F_M\}$ which are used to locate the pixels $\mathcal{P}_k$ of x_k within x_i. In this work, we adopt the dense inverse search-based approach proposed in [41] to derive $F_{k\to i}$ efficiently. Two pixels $P_{k,l} \in \mathcal{P}_k$ and $P_{i,h} \in \mathcal{P}_i$ share the same segmentation class label if $P_{i,h} = P_{k,l} + F_h$. Following this mapping method, the cloud segmentation class labels of M pixels in the previous image x_k are propagated to M pixels of the current image x_i. As a result, we can obtain the cloud segmentation result of x_i before its deadline.

5.2 Delay-mitigated mIoU

5.2.1 Definition

To assess the accuracy of the cloud segmentation results obtained via propagation, we introduce a segmentation accuracy metric, called delay-mitigated mean intersection over union (mIoU). The delay-mitigated mIoU is a variant mIoU used to evaluate the performance of image segmentation models. It measures the average overlap between the propagated predicted segmentation and the ground truth across all classes, given by

$$\text{delay-mitigated mIoU} = \frac{1}{C}\sum_{c=1}^{C}\frac{|\hat{A}_c \cap B_c|}{|\hat{A}_c \cup B_c|}, \tag{1}$$

where C is the total number of classes, $\hat{A}_c$ denotes the set of pixels predicted as class c after propagation, based on the original segmentation result A_c, and B_c denotes the set of ground-truth pixels belonging to class c. The numerator represents the intersection, and the denominator represents the union of the predicted and ground-truth regions for class c. A higher delay-mitigated mIoU indicates better segmentation accuracy.

Fig. 6 shows an example of delay-mitigated mIoU, where the i^{th} image is transmitted to the cloud and its cloud result arrives at the vehicle within the interval between the j^{th} and $(j+1)^{th}$ images. Due to the vehicle's movement, the scene captured in the j^{th} image may shift from the i^{th} image, causing the received cloud result to mismatch with the j^{th} image's ground truth. We adopt the propagation method, discussed in §5.1, to mitigate the mismatch. As a result, the delay-mitigated mIoU for the j^{th} image is the mIoU between the propagated i^{th} cloud result and the ground truth of the j^{th} image. When the pixel content of the i^{th} image differs significantly from that of the j^{th} image, the delay-mitigated mIoU may decrease.

5.2.2 Impact of image content changes

We conduct experiments to investigate the impact of content changes between the i^{th} and j^{th} images, referred to as image distance, on delay-mitigated mIoU. Specifically, we use the average Euclidean distance of each pixel in the movement vectors between two segmented images by InternImage [29] to indicate the image distance. The large image distance indicates fast-changing scenes between two images, such as more objects entering and exiting the scene, as well as rapid changes within the scene itself. The experimental results indicate that as the image distance increases, the mIoU decreases. This decline is attributed to larger distances, where changes in the captured scene may involve objects entering or exiting the field of view, leading to a mismatch that propagation cannot mitigate. More details are provided in Appendix B.1.

5.2.3 Impact of latency

We further conduct experiments to investigate the impact of end-to-end cloud latency on the delay-mitigated mIoU of received cloud results. Specifically, we select 267 images to transmit to the cloud server, where the InternImage serves as the cloud model to process these images. We denote end-to-end cloud latency as L, measured in the number of image intervals T, where $0 \le L \le 19$. $L = 0$ means that the cloud results can be received within the interval of the transmitted image. For each transmitted image, we compute the delay-mitigated mIoU after propagation for L values ranging from 0 to 19, resulting in 5,340 values. The delay-mitigated mIoU reflects the effectiveness of mismatch mitigation through propagation. Additionally, we also compute the mIoU between cloud results without propagation and the ground truth over the same L range (0 to 19), referred to as delay-affected mIoU. Fig. 7 shows the average delay-affected and delay-mitigated mIoU across different L values. The delay-mitigated mIoU is always higher than the delay-affected mIoU, ranging from 0.4 to 0.82. The results show that the propagation can mitigate the impacts of cloud latency. Both mIoU values decrease with L. More details are provided in Appendix B.2.

We also conduct an experiment with controlled end-to-end cloud latency to compare the performance of the cloud and edge processing options in terms of mIoU. Specifically, we use the Cityscapes image traces at 17 FPS and select 267 frames with available ground-truth labels for evaluation. For the cloud processing option, the whole trace is processed by the cloud model with an artificially introduced latency ranging from 60 ms to 960 ms. From the perspective of the vehicle, the latest received cloud result is propagated to the current frame for accuracy evaluation. For the edge processing option, all images can be processed within each frame

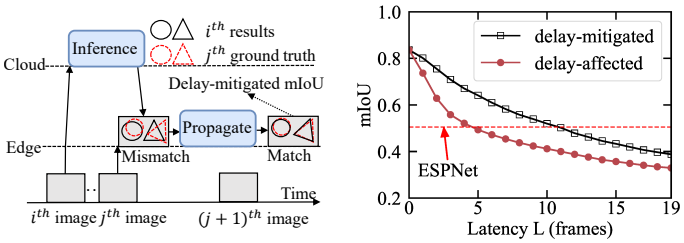


Fig. 6: Delay-mitigated mIoU.

Fig. 7: Accuracy vs. Latency.

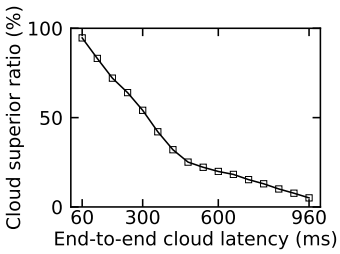
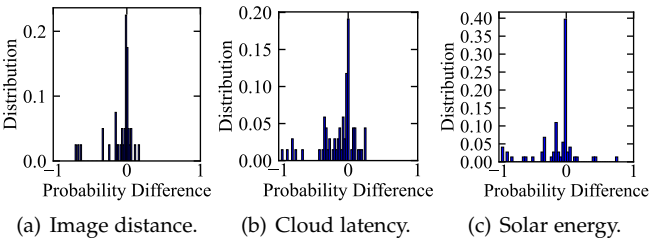


Fig. 8: Ratio of frames where cloud processing outperforms edge processing under different cloud latencies.



(a) Image distance. (b) Cloud latency. (c) Solar energy.

Fig. 9: Compliance of state transition with the Markov assumption.

interval. Fig. 8 reports the ratio of the frames where the cloud processing option achieves a higher mIoU than edge processing under different end-to-end latency settings. The results show that, when the cloud latency is small, the cloud processing option has a higher chance of outperforming edge processing. As the cloud latency increases, this chance gradually decreases because longer delays introduce greater discrepancies between frames, including object entries and exits, which reduce the accuracy of the propagated cloud results. The mechanism presented in §6 will learn a policy that jointly considers a number of factors including the predicted end-to-end cloud latency to identify the opportunities where the cloud processing option would outperform the edge processing option.

6 DRL-BASED SWITCHING

6.1 Problem Statement

6.1.1 Optimization formulation

Time is divided into intervals with identical duration of $\tau \geq T$ seconds, which is referred to as switching period. At the beginning of switching period, called time step, the ECSeg selects the edge or cloud processing options for image segmentation in response to the changes of three exogenous stochastic factors, including the time-varying cloud transmission latency, denoted by $\eta(t)$, the harvested solar energy, denoted by $\epsilon(t)$, and the vehicle's driving scene variation, denoted by $\xi(t)$. Denote $\eta_k = \eta(k\tau)$, $\epsilon_k = \epsilon(k\tau)$, and $\xi_k = \xi(k\tau)$, where $k \in \mathbb{Z}_{\geq 0}$. Denote by $\eta_{t=k\tau}^{(k+1)\tau}$, $\epsilon_{t=k\tau}^{(k+1)\tau}$, and $\xi_{t=k\tau}^{(k+1)\tau}$ the trace of $\eta(t)$, $\epsilon(t)$, and $\xi(t)$ when $t \in [k\tau, (k+1)\tau]$. At the k^{th} time step, we let $\pi(\xi_k, \epsilon_k, \eta_k, \ldots, \xi_0, \epsilon_0, \eta_0)$ denote the policy that determines a switching decision, denoted by ω_k, based on the historical measurements of $(\xi_k, \epsilon_k, \eta_k, \ldots, \xi_0, \epsilon_0, \eta_0)$. The ω_k represents the decisions, including the edge and cloud processing options, which jointly affect the delay-mitigated accuracy and non-renewable energy usage during the switching period. For a time horizon of K switching periods, the switching aims to solve the policy optimization problem:

$$\pi^* = \underset{\pi \in \Pi}{\operatorname{argmax}}\, \mathbb{E}_{\xi,\eta,\epsilon}\left[\frac{1}{K}\sum_{k=0}^{K-1}\left(A_k\left(\omega_k, \xi_{t=k\tau}^{(k+1)\tau}, \eta_{t=k\tau}^{(k+1)\tau}\right) - \lambda E_k\left(\omega_k, \eta_{t=k\tau}^{(k+1)\tau}, \epsilon_{t=k\tau}^{(k+1)\tau}\right)\right)\right] \quad (2)$$

where Π represents the policy space; $\omega_k = \pi(\xi_k, \epsilon_k, \eta_k, \ldots, \xi_0, \epsilon_0, \eta_0)$; $\mathbb{E}_{\xi,\epsilon,\eta}$ denotes the expectation over the three stochastic processes of $\eta(t)$, $\epsilon(t)$, and $\xi(t)$; the $A_k(\cdot)$ denotes the delay-mitigated mIoU in the k^{th} switching period; the $E_k(\cdot)$ denotes the non-renewable energy usage in the k^{th} switching period; the λ is a configurable weight to balance the trade-off between the delay-mitigated mIoU and the non-renewable energy usage. Note that the switching period may have multiple images, depending on the setting of the image frame rate. The objective is to find the optimal policy π^* that maximizes the average delay-mitigated mIoU while minimizes the non-renewable energy usage per switching period.

Solving the policy optimization problem in Eq. (2) faces a basic challenge that the $A_k(\cdot)$ cannot be measured during ECSeg's online operations due to the unavailability of the image segmentation ground truths. Thus, in this study, we develop a learning-based controller to find the near-optimal solution. Specifically, during the offline training phase, the controller learns the optimal switching policy based on the real data traces including the image frames labeled with ground truths, harvested solar energy from the testbed, and cloud processing latencies collected from the ECSeg. Then, the learned policy is applied to determine the processing options during the online inference phase.

6.1.2 Markov property

When applied to image segmentation in AVs, ECSeg aims to switch between edge and cloud processing options based on image distance values, cloud transmission latencies, and harvested solar energy. We conduct experiments to assess whether the above two stochastic processes satisfy the Markov assumption (MA), i.e., $\mathbb{P}[y_{k+1} \mid y_k] = \mathbb{P}[y_{k+1} \mid y_k, y_{k-1} \ldots y_0]$, where y_k represents the measurement state at the k^{th} time step. The MA suggests that the probability distribution of the state transition from y_k to y_{k+1} is independent of the past states $y_{k-1}, \ldots, y_0$. The MA is a basic property of the systems where RL is applicable [42]. When we assess the MA for the image segmentation applications, we use the probability difference, denoted by

$\Delta P = \mathbb{P}[y_{k+1} \mid y_k] - \mathbb{P}[y_{k+1} \mid y_k, \ldots, y_{k-N}]$, where $N \geq 0$, as a MA compliance metric. A lower absolute value of ΔP indicates better compliance. Fig. 9 shows the distributions of ΔP with $N = 1$ for the transitions of the image distance values, the end-to-end cloud latencies, and harvested solar energy in over 800 switching periods, where each switching period τ is one second. From Fig. 9, we can see that these three stochastic processes have good compliance with the MA because their values of ΔP concentrate at zero.

In our expectation, ΔP is close to zero because the key factors that determine future behavior are all recent and already captured in the current state. Specifically, the image distance depends on the most recent visual input, the cloud latency reflects the latest network condition, and the harvested solar energy is closely related to the most recent solar irradiance. Since these factors are already included in the current state, earlier history does not provide additional value for the prediction. Therefore, the Markov property is approximately satisfied, implying that $\Delta P \approx 0$. Although the current state reflects recent inputs, ΔP may still deviate from zero because sudden dynamics such as rapid changes in motion distance, bursty network congestion, or abrupt shifts in solar irradiance can introduce uncertainty in the state transition. In these cases, earlier history may contain residual information that improves the prediction of the next state, resulting in a non-zero ΔP.

6.2 MDP Formulation

In this section, we present two MDP formulations for the two variants of our proposed approach: accuracy-priority ECSeg and low-carbon ECSeg. The two ECSeg variants are designed for different hardware conditions. Specifically, the accuracy-priority variant is applicable when solar panels are not integrated, in which the system focuses on maximizing segmentation accuracy under latency constraints. In contrast, if the vehicle is equipped with the solar panels, the low-carbon variant can be used to leverage the harvested solar energy to reduce the non-renewable energy usage and lower carbon emissions. Specifically, the accuracy-priority ECSeg mainly focuses on switching between the edge and cloud processing options to maximize the AV's image segmentation accuracy without considering the solar energy. It only uses the non-renewable energy from the AV's main battery to power the AV's computation. Differently, the low-carbon ECSeg considers the hybrid energy system and aims to maximize the accuracy while also maximizing the solar energy utilization to reduce the carbon emission.

6.2.1 Accuracy-Priority ECSeg

System state: At the time step k, the system state, denoted by $\boldsymbol{x}_k$, is a vector $\boldsymbol{x}_k = [l_k, u_k]$, where l_k and u_k represent the average mobile network transmission latency and the image distance, respectively in the previous switching period. The u_k is computed via the average pixel Euclidean distance of dense optical flow between the palette-based segmented results of the first frame and the last frame of the previous period. Computing u_k via the segmented results can avoid environmental impacts such as lighting, low texture, and contrast of adopting the dense optical flow methods on the raw image. A large value of l_k can decrease the delay-mitigated mIoU of the received cloud results. The u_k indicates the image distance. A larger image distance can result in a decrease in the delay-mitigated mIoU of the received cloud results.

Moreover, vehicle parameters, such as the ego vehicle's speed, may affect the delay-mitigated mIoU of the received cloud results. Intuitively, higher speeds increase the image distance between consecutive frames, leading to greater mismatches between the received cloud results and the ground truth of the current frame. However, since image distance already reflects the cumulative effect of the vehicle's movement over time, including its speed, explicitly using the speed as an additional input becomes redundant and may introduce noise into the switching controller.

Switching action: The switching action, denoted by $a_k \in \{0, 1\}$, represents the choice between cloud and edge processing options to be executed in the current period.

Reward function: When an action a_k is performed at the system state $\boldsymbol{x}_k$, let $A_k(\boldsymbol{x}_k, a_k)$ denote the average delay-mitigated mIoU for image segmentation over the k^{th} switching period. The immediate reward r_k is defined based on the delay-mitigated mIoU as follows: $r_k = A_k(\boldsymbol{x}_k, a_k)$.

6.2.2 Low-Carbon ECSeg

System state: The system state $\boldsymbol{x}'_k$ is a vector $\boldsymbol{x}'_k = [l_k, u_k, \varepsilon_k, \beta_k]$, where ε_k denotes the amount of energy harvested by the solar panel during the previous period. A larger ε_k indicates efficient energy harvesting, while a smaller ε reflects limited energy harvesting. β_k denotes the remaining solar energy at the beginning of the current period.

Switching action: The switching action $a'_k \in \{0, 1\}$ represents the choice between cloud and edge processing options to be executed in the current period.

Reward function: When an action a'_k is performed at the system state $\boldsymbol{x}'_k$, let $E_k(\boldsymbol{x}'_k, a'_k)$ represent the non-renewable energy usage for executing the selected action a'_k during period k. The immediate reward r'_k is defined based on the non-renewable energy usage and the delay-mitigated mIoU as follows: $r'_k = A_k(\boldsymbol{x}'_k, a'_k) - \lambda \cdot \mathcal{N}(E_k(\boldsymbol{x}'_k, a'_k))$, where λ is configurable weight. $\mathcal{N}(x)$ is a normalization function, i.e., $\mathcal{N}(x) = \frac{x}{x_{\max}}$, where $x_{\max}$ is the maximum value of x. Recall that the AV only uses the non-renewable energy (i.e., $E_k(x'_k, a'_k) > 0$) when no solar energy is available. Thus, the design objective of r'_k is to reduce the carbon emission by minimizing the usage of non-renewable energy.

6.3 DRL Training

We adopt the learning framework of a DRL algorithm, called the proximal policy optimization (PPO) [43] to learn the optimal switching policy. PPO directly learns from a stochastic distribution, leading to more effective exploration, and adopts a clipping mechanism to improve efficiency compared with Deep Q-Network and Trust Region Policy Optimization. Under the typical setting, a PPO agent learns the optimal policy during the online interactions with the controlled system. However, for the formulated switching problem, the online DRL scheme faces the following two

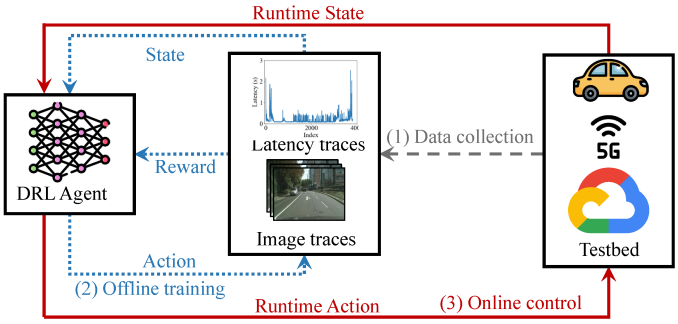


Fig. 10: Workflow of DRL-based switching.

challenges. First, PPO agent trials may result in poor segmentation accuracy, potentially leading to safety concerns. Second, during the online learning phase, the mIoU cannot be measured during online learning due to the lack of ground-truth labels. To address these challenges, we adopt an offline training approach as illustrated in Fig. 10, which consists of three steps. First, we collect the real data traces from the deployment environment. Second, we use the real data traces collected in the first step to drive the offline training of the PPO agent. Third, the well-trained PPO agent is deployed to make decisions for switching.

Real-world operating environments may differ from the offline training environment. A possible solution is to adopt a Tesla-like over-the-air (OTA) mechanism [44] that collects image traces, latency statistics, and solar energy harvesting data during operation. These data are uploaded to the cloud server when the vehicle is parked. The uploaded data can then be used to further train the deep reinforcement learning agent in the cloud, after which the updated model is sent back to the vehicle through OTA updates.

7 Evaluation

This section first presents the experiment settings and evaluation results of accuracy-priority ECSeg, followed by those of low-carbon ECSeg.

7.1 Experiment Settings

7.1.1 System implementation

We conduct a series of experiments to investigate the performance of ECSeg in a real-world testbed. Fig. 11 illustrates the system setup. For image sensing, we use a See3Cam USB camera [45] with a 13-megapixel CMOS sensor, capturing images at 2048×1024 resolution at 17 Hz. We utilize a laptop equipped with an NVIDIA GeForce RTX 3080 Ti GPU as the edge platform mounted in the vehicle. For mobile network connectivity on the laptop, we utilize the SIM8202G module [46], which is a USB3.1 5G dongle equipped with four antennas and the Qualcomm Snapdragon X55 RF-modem. The cloud server is prototyped using a tower server with an RTX 8000 GPU and an Intel Xeon Gold 6246 CPU located in a university server room. We use Python 3.8's Multiprocessing library to manage concurrent processes, including image capturing, edge processing, and cloud processing. The InternImage model serves as the cloud model, using TensorRT 8.2.5.1. The deep learning-based segmentation model and PPO agents are implemented using PyTorch 1.11 and Tensorforce 0.6.5, respectively. The optical flow computing and image propagation are implemented using OpenCV 4.5.0.

We use the testbed to collect a self-collected real-world image dataset which comprises 12,036 road scene images captured with a USB camera at 17 Hz. The deadline is set to the image interval of 58.8 ms. During image capture, the testbed is deployed on a car that travels across various urban traffic conditions, including traffic jams, traffic lights, smooth traffic, and pedestrian crossings. The images are labeled with 19 classes. Moreover, we use the segmentation results obtained by executing the InternImage model as pseudo labels for the collected image. In addition to the self-collected dataset, we also use the Cityscapes dataset [30] to evaluate our system.

We build a PPO-based DRL model with an input layer, three hidden layers and an output layer. The three hidden layers has 128, 64, and 32 ReLUs, respectively. The Adam optimizer with a learning rate of 10^{-3} is used for training. The PPO agent is trained with a likelihood ratio clipping set at 0.25 to balance stability during the learning process. Moreover, the switching period τ is set to 1 second. At the beginning of every period, the DRL agent observes a system state $\mathbf{x}$ including the image distance and cloud transmission latency. Then, it selects an action $a \in \{0, 1\}$ to choose the edge or cloud processing options. We employ ESPNet as the local model and InternImage as the cloud model. During the offline training phase, we utilize a trace-driven approach to accelerate the training process of the PPO agent. Specifically, the PPO agent makes decisions based on segmented image and latency traces. Subsequently, the edge and cloud model results' traces are utilized to compute intermediate reward values.

7.1.2 Comparison baselines

To assess the efficacy of the proposed ECSeg system, we compare our ECSeg with four baseline methods as follows: (1) *Edge:* The vehicle always executes the ESPNet [5] model to process all image frames without switching to the cloud processing option. (2) *KD:* The vehicle always executes a lightweight model with five convolutional layers, trained using knowledge distillation with a soft-label-based loss function, to process all image frames without switching to the cloud processing option. (3) *Offline:* Largely resembling of DeepDecision proposed in [17]. Switching between edge and cloud processing options is based on thresholds for observed latency and image distance. These thresholds are determined using offline data traces [17]. (4) *MPC:* Largely resembling of ACCUMO proposed in [18]. Switching between edge and cloud processing options is determined by a model predictive control (MPC) method, similar to [18]. The MPC uses autoregressive moving average (ARMA) models to predict cloud transmission latencies and single-frame image distances, which are then used to compute the average image distance of received cloud results for the next switching period. An offline lookup table maps the average image distance to the delay-mitigated mIoU. To reduce computing overhead, the horizon length of MPC is set to one. Based on the image distance, an offline polynomial regression model estimates the delay-mitigated mIoU for the future period. The action with the highest delay-mitigated mIoU for the

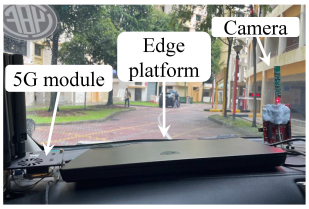


Fig. 11: Setup installed under a car's front windshield.

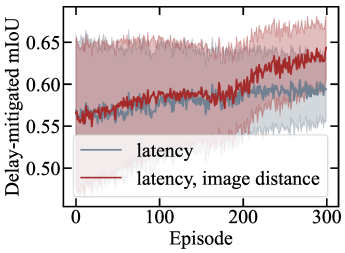
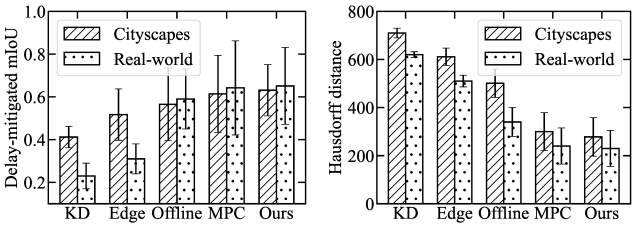


Fig. 12: Training results.



(a) Segmentation performance. (b) Boundary performance.

Fig. 13: Results from the Cityscapes and real-world dataset experiment.

next switching period is selected. The original DeepDecision and ACCUMO adopt an image-offloading strategy where input images are sent to the cloud and the system waits for the returned cloud results. However, cloud latency may exceed the image processing deadline, leading to missed deadlines. To avoid deadline misses in their methods, we adopt our propagation approach to generate the segmentation output by propagating the last available result when the cloud latency exceeds the image processing deadline, and only use their decision-making mechanism for comparison.

7.1.3 Performance metrics

We adopt the following two metrics to evaluate our proposed ECSeg and baseline approaches. (1) *Segmentation performance:* The average delay-mitigated mIoU is used as the metric to assess the accuracy of the image segmentation. (2) *Boundary detection performance:* The image segmentation results can serve as input for higher-level applications, such as boundary detection in AVs. Boundary detection can help constrain AVs to avoid the collision, ensuring driving safety [47]. In this paper, we use boundary detection as a case study to investigate the performance of the ECSeg for high-level applications. In this case study, we use the metric of Hausdorff distance to measure the similarity between the all moving object boundary of the ground truth and the segmented results.

7.2 Evaluation of Accuracy-Priority ECSeg

7.2.1 Training results

The offline training is conducted for 300 episodes, each of which includes 500 switching periods. In addition, we use 6,000 images from the Cityscapes dataset and 6,000 latency samples measured in a moving vehicle over six hours to train the PPO agent. Fig. 12 shows the PPO training traces of the rewards with various state inputs, including the latency and latency combined with image distance. Along the training episodes, the reward trace increases and then becomes flat under different inputs. The results show that the PPO agent can converge after a certain number of training episodes, and the agent with the state input combining latency and image distance achieves better convergence performance compared with single latency input.

7.2.2 Execution results

Fig. 13(a) presents the delay-mitigated mIoU for the proposed accuracy-priority ECSeg and four baseline approaches over a half-hour using the Cityscapes and self-collected datasets. The edge approach can achieve 0.505 delay-mitigated mIoU across all periods. The performance difference between the Cityscapes and real-world datasets occurs because the real-world dataset uses pseudo labels as ground truths for evaluation. These pseudo labels contain errors or misclassifications, leading to lower evaluation accuracy compared with using actual ground truths. Compared with the four baseline methods, our approach achieves the highest delay-mitigated mIoU and outperforms KD by over 48.8% in Cityscapes dataset. This is because the PPO agent makes good decisions to utilize accurate cloud results when the mobile network channel conditions are good. Our approach requires only 10 ms per decision, while the MPC method takes 400 ms, making it unsuitable for timely decisions within the deadline. Additionally, increasing the horizon length of the MPC leads to exponential growth in compute time.

Fig. 13(b) presents the moving boundary detection accuracy for the proposed accuracy-priority ECSeg and four baseline approaches. The results show a similar trend to the image segmentation performance, with our approach achieving the lowest Hausdorff distance compared with the other four approaches. This is because the larger segmentation model deployed in the cloud provides accurate segmented pixels for detecting small moving objects. Moreover, our approach achieves a Hausdorff distance that is over 60.5% shorter than that of the edge approach. Fig. 15 presents the average delay-mitigated mIoU results of Offline, MPC, and our method on the Cityscapes dataset under varying latency conditions. Overall, the delay-mitigated mIoU decreases as latency increases. The mIoU of the Offline method drops from 0.78 to 0.45, highlighting that it does not effectively adapt to dynamic latency fluctuation due to the lack of a good switching policy.

We also extend our experiments to a 30 FPS setting. Specifically, we create a higher-frame-rate version of our self-collected dataset by reducing the frame interval from 58.8 ms to 33.3 ms. Then, we conduction additional experiments that use the resulting 30 FPS trace to retrain and evaluate our proposed DRL-based switching approach. As shown in Fig. 14, our method still provides accuracy gains, which result in a higher mIoU compared with the edge-only method.

Fig. 16 presents the visualization cloud processing results under various latency settings. For each case, we show the groundtruth, cloud results with 0 ms and 900 ms latency, and edge results. When the cloud latency reaches 900 ms, cloud results can still outperform edge results in relatively

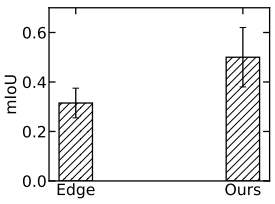


Fig. 14: mIoU performance of 30 FPS.

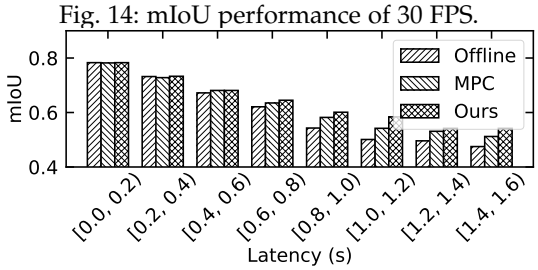


Fig. 15: mIoU performance under cloud latency.

static environments, as seen in Case a and Case b, where the scene remains mostly unchanged and the delayed cloud output remains accurate. In contrast, Case c illustrates a negative case for cloud processing, where the environment changes rapidly and the outdated cloud result becomes less accurate than the edge output. These observations indicate that the effectiveness of cloud processing is closely tied to the dynamics of the driving environment. In fast-changing scenes, new objects may enter or existing objects may leave the frame during the latency period, making the propagated cloud results unreliable. This highlights the necessity of dynamically switching between edge and cloud processing to ensure accurate and timely segmentation.

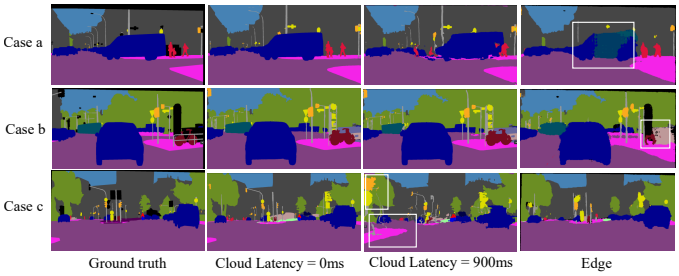


Fig. 16: Visualization of segmentation results. Segmentation errors are highlighted within white boxes.

7.2.3 Performance of ECSeg under different cloud latency conditions

We conduct experiments to investigate the performance of ECSeg in different cities. We measure 20-minute cloud latency for one cloud server in two cities: City 1, close to the server, and City 2, over 1000 km away. In addition to the cloud geo-distance, the cities may have different mobile network data transmission speeds, resulting in different latency. Note that the evaluation in §7 is conducted in City 1. Fig. 17 shows the box plots for the distributions of the data transmission latency in the two cities. The average latency in City 1 is about 110 ms, while in City 2 it is about 205 ms.

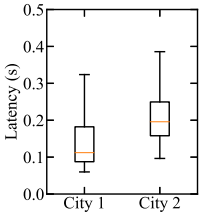


Fig. 17: Communication latency in two city.

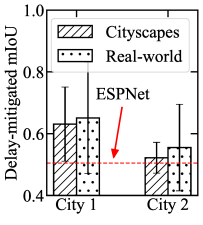


Fig. 18: ECSeg performance in two city.

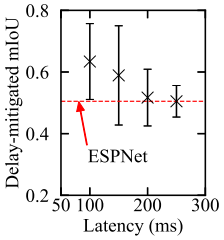


Fig. 19: ECSeg performance vs. latency.

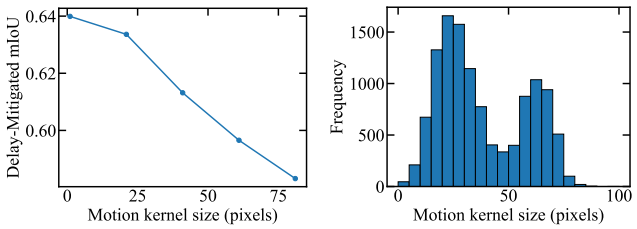
We evaluate ECSeg using these two latency traces on the Cityscapes and real-world datasets. Fig. 18 shows the average delay-mitigated mIoU for the two datasets under the two latency traces. Compared with City 1, the PPO agent's performance drops in City 2 for both datasets. For the Cityscapes dataset, the PPO agent's performance is only 0.517, close to the edge-based approach. This indicates that the PPO agent selects the edge processing option most of the time.

We further investigate ECSeg performance under different transmission latencies. Using the Cityscapes dataset and 10-minute latency traces from City 1 with various artificial biases, we simulate different latency situations. Fig. 19 shows ECSeg's mIoU under different average latency traces. The mIoU drops with average latency. When the average latency reaches 250 ms, ECSeg's mIoU is almost the same as the edge processing option because the PPO agent always selects the edge option over the cloud option. Therefore, cities prefer to have cloud servers within the city to support ECSeg.

7.2.4 Impact of motion blur on delay-mitigated mIoU

We also evaluate the impact of motion blur on delay-mitigated mIoU, using 267 images and a fixed cloud latency of 300 ms. To investigate the impact of different blur strengths on delay-mitigated mIoU, we generate artificial motion blur on 267 clean images from the Cityscapes dataset by convolving them with horizontal blur kernels. Specifically, we simulate motion blur by convolving the image with horizontal kernels of size 1, 21, 41, 61 and 81 pixels. A larger kernel size corresponds to stronger motion blur, as it averages pixel values over a wider horizontal span. Fig. 20(a) presents the delay-mitigated mIoU under varying motion blur kernel sizes. The results show a consistent decrease in mIoU as blur severity increases. Specifically, as the kernel size increases from 1 to 81, the mIoU drops from 0.639 to 0.584, indicating that stronger motion blur degrades optical flow accuracy and leads to larger error in the propagated segmentation results. This degradation occurs because heavier blur makes it more difficult to estimate the precise movement velocity of pixels to be propagated, thereby reducing the accuracy of the propagated output.

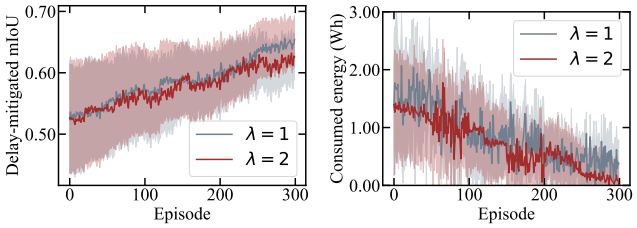
To understand whether the blur strength settings in Fig. 20(a) are realistic, we have conducted an additional experiment. Specifically, we estimate the motion blur strength of 10,000 real-world images using a Radon-based method [48]. We apply the Radon transform to the log-magnitude Fourier spectrum of each image to estimate its blur angle and motion kernel size, with the estimated length serving as a quantitative indicator of motion blur strength. Fig. 20(b) presents the distribution of the estimated motion blur kernel sizes, which are mostly concentrated in the range of 0 to 80 pixels. Therefore, the motion kernel size settings



(a) Effect of motion kernel size on delay-mitigated mIoU.

(b) Motion blur distribution.

Fig. 20: Motion blur impact on delay-mitigated mIoU.



(a) Delay-mitigated mIoU.

(b) Non-renewable energy usage.

Fig. 21: Training results with various λ.

adopted in our experiments as presented in Fig. 20(a) are realistic.

7.3 Evaluation of Low-Carbon ECSeg

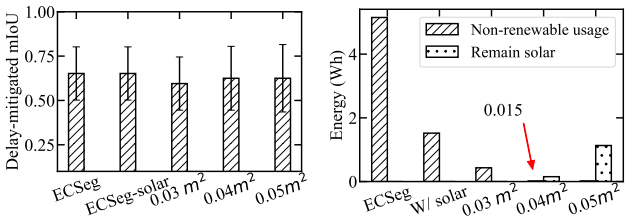
In this section, we compare our proposed low-carbon ECSeg with several baseline approaches using various performance metrics.

7.3.1 Training results

We use the real-world data traces collected from our testbed to train the PPO agent offline. Specifically, we use 5,000 images from the Cityscapes dataset, 5,000 latency samples and corresponding energy consumption collected from a moving vehicle over one hour. The energy harvesting data is recorded over two hours from a moving vehicle with a solar panel. The offline training is conducted for 300 episodes, each of which includes 3600 switching periods. The efficiency of harvested energy is related to the size of the solar panel. To demonstrate the effectiveness of low-carbon ECSeg, we scale down the solar energy traces harvested from a 100 W solar panel to simulate those of an 8W solar panel, which corresponds to a panel size of 0.04 m^2. Figs. 21(a) and (b) illustrate the training trajectories of the delay-mitigated mIoU and the non-renewable energy usage over 300 epochs. Along with the training epochs, the delay-mitigated mIoU consistently increases and then becomes flat across different λ settings. When λ = 2, the non-renewable energy usage can almost reach zero at the end of training period. The results show that the PPO agent converges after a certain number of training epochs (e.g., 250).

7.3.2 Execution results

We evaluate the DRL execution performance of the proposed low-carbon ECSeg for an AV equipped with solar



(a) Image segmentation accuracy.

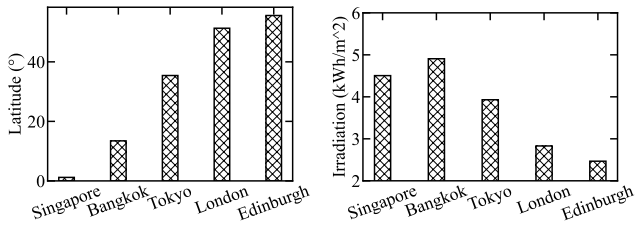
(b) Non-renewable energy usage and harvested solar energy remaining after one hour.

Fig. 22: The performance results of two baselines and three variants of low-carbon ECSeg with the solar panels sized from 0.03 m^2, 0.04 m^2, 0.05 ^{2}. For the reward function r_k of the low-carbon ECSeg, the weight λ = 2.

panels of different sizes: 0.03 m^2, 0.04 m^2, and 0.05 m^2. We compare the low-carbon ECSeg with the following two baselines: (1) ECSeg is the proposed DRL-based switching approach (c.f. §6) for selecting the edge and cloud processing options of the AV without the solar energy usage; (2) ECSeg-solar is the low-carbon ECSeg which is equipped with a solar panel sized 0.04 m^2 but does not consider the energy usage for selecting the edge or cloud action, i.e., the weight λ of the reward function r_k is set to zero.

Fig. 22 presents the delay-mitigated mIoU and energy usages of the proposed low-carbon ECSeg with various solar panel sizes and two baseline approaches over one hour using the Cityscapes dataset. From Fig. 22(a), three variants of the low-carbon ECSeg and two baseline approaches achieve the similar delay-mitigated mIoU (i.e., image segmentation accuracy) values. From Fig. 22(a), ECSeg uses the highest amount of the non-renewable energy since the AV is not equipped with the solar panel for harvesting the renewable energy. The low-carbon ECSeg uses the less amount of the non-renewable energy with the larger solar panel. This is because the amount of energy harvested by the solar panel increases the panel size. For example, with the solar panel size of 0.05 m^2, the harvested solar energy is always sufficient for executing the AV's image segmentation tasks without the usage of non-renewable energy. These results imply that the low-carbon ECSeg can minimize the carbon emission by minimizing the usage of non-renewable energy while maintaining the similar segmentation accuracy, compared with the two baseline approaches. Moreover, with the same solar panel size of 0.04 m^2, the ECSeg-solar and low-carbon ECSeg use 1.32 Wh and 0.015 Wh from the non-renewable energy sources, respectively, over the execution period of one hour. It means that the low-carbon ECSeg can reduce the non-renewable energy by 98.8%, compared with the ECSeg-solar.

We also evaluate the total carbon emission reduction that can be achieved by deploying the low-carbon ECSeg system. We estimate the total daily carbon emission of the cars worldwide as: $E_{\text{total}} = N_{\text{cars}} \times E_{\text{saved}} \times H_{\text{daily}}$, where E_{total} represents the total daily carbon emission reduction (gCO_2-eq), N_{cars} is the number of cars worldwide, E_{saved} is the carbon emission reduction per hour of operation (gCO_2-eq/h), and H_{daily} is the number of operating hours per day. According to [49], there are approximately N_{cars} = 1.475



(a) Latitude in varies cities. (b) Solar irradiation in different cities.

Fig. 23: Latitude factors on solar panel performance.

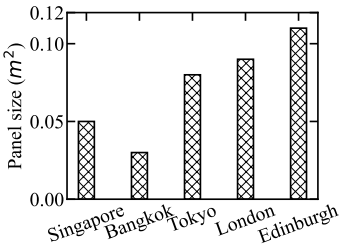


Fig. 24: Minimum panel size required to achieve zero carbon emissions in different cities.

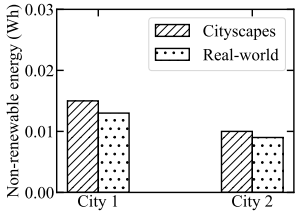


Fig. 25: Low-carbon ECSeg performance under in different cities latency.

billion cars operating worldwide in 2024. From Fig. 22(b), the low-carbon ECSeg with the solar panel sized 0.04 m^2 can reduce an amount of non-renewable energy by 5.135 Wh per hour of operation, compared with the ECSeg approach. It is assumed that the production of 1 kWh non-renewable energy approximately 390 gCO_2-eq [32]. Then, the low-carbon ECSeg can reduce the carbon emission E_{saved} =2 gCO_2-eq/h, compared with the ECSeg. We assume ideal conditions with sunny weather and good sunlight availability, where solar irradiation provides an average of $H_{\text{daily}} = 4.5$ peak sun hours per day [50]. Thus, our proposed low-carbon ECSeg approach can achieve the total daily carbon emission reduction E_{total} = 13,275 tons of CO_2-eq per day.

7.3.3 Solar energy harvesting in various cities

The performance of solar panels may vary across cities at different latitudes. Due to the Earth's axial tilt, cities at lower latitudes receive higher solar irradiance. For instance, cities near the equator receive more consistent solar irradiation compared with those located closer to the poles. We collect the solar irradiation data in five cities with different latitudes from a global solar energy platform, called Global Solar Atlas [51]. Fig. 23 presents the latitudes of five cities located at approximately sea level and their corresponding average daily global horizontal irradiation over a 10-year period. As latitude increases, solar irradiation decreases due to reduced direct sunlight. For example, the solar irradiation in Bangkok (latitude 13.45°) is 1.98× that of Edinburgh (latitude 55.57°).

In this section, we conduct an analysis to determine the minimum solar panel size required for our proposed low-carbon ECSeg to achieve zero carbon emission in various cities across different latitudes. The harvested energy is proportional to solar irradiation levels. To simulate harvested solar energy traces in different cities, we convert the real-world harvested solar energy traces collected from a data collection city to different target cities. Specifically, we apply the uniform scaling method [52] to scale the energy traces from the data collection city to match the solar energy characteristics of the target city. For each value in the harvested energy trace of the data collection city, we multiply it by the ratio of the target city's solar irradiation level to that of the data collection city. This ratio ensures that the scaled energy traces reflect the solar energy potential of the target city. Additionally, to account for differences in solar panel size, we further adjust the scaled energy traces by proportionally scaling the values according to the relative size of the solar panel.

Fig. 24 illustrates the minimum solar panel size required to achieve zero non-renewable energy consumption. Bangkok requires the smallest solar panel size to eliminate non-renewable energy usage due to its high solar irradiation. Even in Edinburgh, where solar irradiation is lower, the required panel size remains within the typical 5 m^2 car roof area [53], making it feasible for installation.

7.3.4 Performance of low-carbon ECSeg under different cloud latency conditions

We further conduct experiments to evaluate the performance of low-carbon ECSeg under different mobile network latency conditions in different cities. Based on the latency results of City 1 and City 2 mentioned in §7.2.3, we evaluate the two latency traces using the Cityscapes dataset, self-collected, and harvested solar energy datasets. We assume that City 1 and City 2 have the same solar irradiation condition. The size of solar panel is set as the 0.04 m^2. To simulate the energy traces corresponding to specific latency ranges in City 2, we employ a Gaussian model to estimate energy consumption across different latency intervals: [0, 0.2), [0.2, 0.4), [0.4, 0.6), [0.6, 0.8), and [0.8, 1). These models are constructed based on real-world latency and energy traces collected from City 1.

Fig. 25 shows the non-renewable energy usage over 1-hours for the two image datasets. Both energy consumption values are very low (0.01 Wh), nearly zero, indicating that the low-carbon ECSeg system can achieve low-carbon emissions under different latency traces.

8 CONCLUSION

This paper presents ECSeg, an edge-cloud switched image segmentation system for autonomous vehicle (AV) applications. We formulate a switching problem to maximize the delay-mitigated mIoU of image segmentation, accounting for dynamic variations in mobile network conditions and image content changes. Additionally, we use a solar panel to charge the car-borne battery, avoiding the use of non-renewable energy for edge and cloud processing to reduce carbon emissions. To achieve this, we apply deep reinforcement learning to learn the optimal switching policy for image segmentation processing and carbon emission reduction. Our results demonstrate that cloud computing can enhance AV performance, and low-carbon ECSeg achieves a high delay-mitigated mIoU for image segmentation in autonomous driving applications while maintaining low carbon emissions compared with baseline approaches.

In future work, our edge-cloud switching framework can also be extended to other perception tasks, such as object detection and depth estimation. For object detection, a high-accuracy model can be deployed on the cloud, while the edge device propagates the detected bounding boxes across frames to mitigate cloud latency. Similarly, for depth estimation, the cloud can execute a high-accuracy model, and the estimated depth maps can be propagated on the edge for latency mitigation.

ACKNOWLEDGMENT

This research is supported by Singapore Ministry of Education under its AcRF Tier-1 grants RT14/22 and RG88/22.

REFERENCES

[1] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, 2017.

[2] D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool, "Towards end-to-end lane detection: an instance segmentation approach," in *IEEE Intell. Veh. symposium*, 2018.

[3] J. Xu, Y. Nie, P. Wang, and A. M. López, "Training a binary weight object detector by knowledge transfer for autonomous driving," in *ICRA*, 2019, pp. 2379–2384.

[4] "Apollo." [Online]. Available: https://www.apollo.auto

[5] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, and H. Hajishirzi, "Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation," in *ECCV*, 2018.

[6] L. Liu, S. Lu, R. Zhong, B. Wu, Y. Yao, Q. Zhang, and W. Shi, "Computing systems for autonomous driving: State of the art and challenges," *IEEE IoT J.*, vol. 8, no. 8, pp. 6469–6486, 2020.

[7] X. Zhang, S. Ji, H. Wang, and T. Wang, "Private, yet practical, multiparty deep learning," in *ICDCS*, 2017.

[8] H. Kiani Galoogahi, A. Fagg, C. Huang, D. Ramanan, and S. Lucey, "Need for speed: A benchmark for higher frame rate object tracking," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1125–1134.

[9] Y. Cai, T. Luan, H. Gao, H. Wang, L. Chen, Y. Li, M. A. Sotelo, and Z. Li, "Yolov4-5d: An effective and efficient object detector for autonomous driving," *IEEE Trans. Instrum. Meas.*, 2021.

[10] J. H. Gawron, G. A. Keoleian, R. D. De Kleine, T. J. Wallington, and H. C. Kim, "Life cycle assessment of connected and automated vehicles: sensing and computing subsystem and vehicle level effects," *Environmental Sci. & Techno.*, vol. 52, no. 5, pp. 3249–3256, 2018.

[11] A. Hassan, A. Narayanan, A. Zhang, W. Ye, R. Zhu, S. Jin, J. Carpenter, Z. M. Mao, F. Qian, and Z.-L. Zhang, "Vivisecting mobility management in 5g cellular networks," in *SIGCOMM*, 2022.

[12] S. Liu, J. Du, K. Nan, Z. Zhou, H. Liu, Z. Wang, and Y. Lin, "Adadeep: A usage-driven, automated deep model compression framework for enabling ubiquitous intelligent mobiles," *IEEE Trans. Mobile Comput.*, vol. 20, no. 12, pp. 3282–3297, 2021.

[13] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, "Segment anything," *ICCV*, 2023.

[14] Z. Chen, J. Wu, W. Wang, W. Su, G. Chen, S. Xing, M. Zhong, Q. Zhang, X. Zhu, L. Lu *et al.*, "Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks," in *CVPR*, 2024, pp. 24 185–24 198.

[15] X. Tian, J. Gu, B. Li, Y. Liu, Y. Wang, Z. Zhao, K. Zhan, P. Jia, X. Lang, and H. Zhao, "Drivevlm: The convergence of autonomous driving and large vision-language models," in *8th Annual Conference on Robot Learning*, 2024.

[16] Z. Cao, Y. Cheng, Z. Zhou, Y. Chen, Y. Hu, A. Lu, J. Liu, and Z. Li, "Edge-cloud collaborated object detection via bandwidth adaptive difficult-case discriminator," *IEEE Trans. Mobile Comput.*, vol. 24, no. 2, pp. 1181–1196, 2024.

[17] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, "Deepdecision: A mobile deep learning framework for edge video analytics," in *INFOCOM*, 2018.

[18] Z. J. Kong, Q. Xu, J. Meng, and Y. C. Hu, "Accumo: Accuracy-centric multitask offloading in edge-assisted mobile augmented reality," in *MobiCom*, 2023.

[19] X. He, S. Wang, X. Wang, S. Xu, and J. Ren, "Age-based scheduling for monitoring and control applications in mobile edge computing systems," in *INFOCOM*, 2022, pp. 1009–1018.

[20] T. Ren, Z. Hu, H. He, J. Niu, and X. Liu, "Feat: Towards fast environment-adaptive task offloading and power allocation in mec," in *INFOCOM*, 2023, pp. 1–10.

[21] S. Zhou, D. Van Le, R. Tan, J. Q. Yang, and D. Ho, "Configuration-adaptive wireless visual sensing system with deep reinforcement learning," *Trans. Mobile Comput.*, 2022.

[22] F. Fraternali, B. Balaji, Y. Agarwal, and R. K. Gupta, "Aces: Automatic configuration of energy harvesting sensors with reinforcement learning," *ACM Trans. Sensor Netw.*, vol. 16, no. 4, pp. 1–31, 2020.

[23] J. Ku, S.-M. Kim, and H.-D. Park, "Energy-saving path planning navigation for solar-powered vehicles considering shadows," *Renewable Energy*, vol. 236, p. 121424, 2024.

[24] M. H. Mobarak, R. N. Kleiman, and J. Bauman, "Solar-charged electric vehicles: A comprehensive analysis of grid, driver, and environmental benefits," *IEEE Trans. Transport. Electrific.*, vol. 7, no. 2, pp. 579–603, 2020.

[25] M. Masoudi and C. Cavdar, "Device vs edge computing for mobile services: Delay-aware decision making to minimize power consumption," *IEEE Transactions on Mobile Computing*, vol. 20, no. 12, pp. 3324–3337, 2020.

[26] D. Zhang, L. Tan, J. Ren, M. K. Awad, S. Zhang, Y. Zhang, and P.-J. Wan, "Near-optimal and truthful online auction for computation offloading in green edge-computing systems," *IEEE Transactions on Mobile Computing*, vol. 19, no. 4, pp. 880–893, 2019.

[27] "Orin." [Online]. Available: https://www.nvidia.com/en-sg/autonomous-machines/embedded-systems/jetson-orin/

[28] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *CVPR*, 2017.

[29] W. Wang, J. Dai, Z. Chen, Z. Huang, Z. Li, X. Zhu, X. Hu, T. Lu, L. Lu, H. Li *et al.*, "Internimage: Exploring large-scale vision foundation models with deformable convolutions," in *CVPR*, 2023.

[30] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *CVPR*, 2016.

[31] J. G. G. Jonker, M. Junginger, and A. Faaij, "Carbon payback period and carbon offset parity point of wood pellet production in the south-eastern united states," *Gcb Bioenergy*, vol. 6, no. 4, pp. 371–389, 2014.

[32] Y. Gan, A. Elgowainy, Z. Lu, J. C. Kelly, M. Wang, R. D. Boardman, and J. Marcinkoski, "Greenhouse gas emissions embodied in the us solar photovoltaic supply chain," *Environmental Research Lett.*, vol. 18, no. 10, p. 104012, 2023.

[33] "Carbon of electric." [Online]. Available: https://www.eia.gov/tools/faqs

[34] J. Peng, L. Lu, and H. Yang, "Review on life cycle assessment of energy payback and greenhouse gas emission of solar photovoltaic systems," *Renewable Sustain. Energy Rev.*, vol. 19, pp. 255–274, 2013.

[35] M. S. Chowdhury, K. S. Rahman, T. Chowdhury, N. Nuthammachot, K. Techato, M. Akhtaruzzaman, S. K. Tiong, K. Sopian, and N. Amin, "An overview of solar photovoltaic panels' end-of-life material recycling," *Energy Strategy Rev.*, vol. 27, p. 100431, 2020.

[36] S. El Himer, M. Ouaissa, and M. Ouaissa, "Systems for car-roof application," *Artificial Intelligence of Things for Smart Green Energy Management*, vol. 446, p. 67, 2022.

[37] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[38] T. ANDREW S and W. DAVID J, "Computer networks fifth edition," 2011.

[39] P. Wu, X. Jia, L. Chen, J. Yan, H. Li, and Y. Qiao, "Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline," *NIPS*, vol. 35, pp. 6119–6132, 2022.

[40] Y. Hu, J. Yang, L. Chen, K. Li, C. Sima, X. Zhu, S. Chai, S. Du, T. Lin, W. Wang *et al.*, "Planning-oriented autonomous driving," in *CVPR*, 2023, pp. 17 853–17 862.

[41] T. Kroeger, R. Timofte, D. Dai, and L. Van Gool, "Fast optical flow using dense inverse search," in *ECCV*, 2016.

[42] C. Shi, R. Wan, R. Song, W. Lu, and L. Leng, "Does the markov decision process fit the data: Testing for the markov property in sequential decision making," in *ICML*, 2020.

[43] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv*, 2017.
[44] "Tesla software updates," 2025. [Online]. Available: https://www.tesla.com/en-sg/support/software-updates/
[45] "See3cam camera." [Online]. Available: https://www.e-consystems.com
[46] "5g module." [Online]. Available: https://www.waveshare.com
[47] Z. Xu, Y. Sun, and M. Liu, "Topo-boundary: A benchmark dataset on topological road-boundary detection using aerial images for autonomous driving," *IEEE Trans. Robot. Autom.*, vol. 6, no. 4, pp. 7248–7255, 2021.
[48] S. Tiwari, V. P. Shukla, A. Singh, and S. Biradar, "Review of motion blur estimation techniques," *J. Image Graphics*, vol. 1, no. 4, pp. 176–184, 2013.
[49] "world health organization." [Online]. Available: https://www.whichcar.com.au/news/how-many-cars-are-there-in-the-world
[50] P. Megantoro, M. A. Syahbani, I. H. Sukmawan, S. D. Perkasa, and P. Vigneshwaran, "Effect of peak sun hour on energy productivity of solar photovoltaic power system," *Bulletin Electr. Eng. Inform.*, vol. 11, no. 5, pp. 2442–2449, 2022.
[51] "atlas." [Online]. Available: https://globalsolaratlas.info/map
[52] "scaling." [Online]. Available: https://en.wikipedia.org/wiki/scaling-geometry
[53] G. Minak, "Solar energy-powered boats: State of the art and perspectives," *J. Marine Sci. Eng.*, vol. 11, no. 8, p. 1519, 2023.

Siyuan Zhou received his Ph.D. degree from the College of Computing and Data Science (CCDS), Nanyang Technological University (NTU), Singapore. He received his B.Eng. degree in communication engineering from Soochow University, China, in 2018, and the M.S. degree in electrical engineering from the National University of Singapore in 2019. His research interests include cyber-physical systems (CPS), the Internet of Things (IoT), and edge artificial intelligence (Edge AI).

Duc Van Le (Senior Member, IEEE) received the BEng (Distinction) degree in electronics and telecommunications engineering from Le Quy Don Technical University, Vietnam, in 2011, and the PhD degree in computer engineering from University of Ulsan, South Korea, in 2016. He is currently a Postdoctoral Fellow with School of Electrical Engineering and Telecommunications, University of New South Wales (UNSW), Australia. From 2016 to 2024, he was a Senior Research Fellow with Nanyang Technological University (NTU), and a Research Fellow with National University of Singapore (NUS). His research interests include sensor networks, Internet of Things, cyber-physical systems, and applied machine learning. He was a recipient of the ACM/IEEE ICCPS'23 Best Paper Award.

Rui Tan is a Full Professor at College of Computing and Data Science, Nanyang Technological University, Singapore. Previously, he was a Research Scientist (2012-2015) and a Senior Research Scientist (2015) at Advanced Digital Sciences Center of University of Illinois at Urbana-Champaign, and a postdoctoral Research Associate (2010-2012) at Michigan State University. He received the Ph.D. (2010) degree in computer science from City University of Hong Kong, the B.S. (2004) and M.S. (2007) degrees from Shanghai Jiao Tong University. His research interests include cyber-physical systems and Internet of things. He is the recipient of Best Demo Award from SenSys'24, Best Paper Awards from ICCPS'23 and IPSN'17. He served as Associate Editor of IEEE Transactions on Mobile Computing and ACM Transactions on Sensor Networks, TPC Co-Chair of e-Energy'23, EWSN'24, SenSys'24, and General Co-Chair of e-Energy'24 and RTCSA'25. He received the Distinguished TPC Member recognition from SenSys in 2025 and from INFOCOM in 2017, 2020, and 2022. He is a Senior Member of IEEE.