Split Computing in Autonomous Mobility for Efficient Semantic Segmentation using Transformers

***Abstract*—With the advancement of deep learning in Connected Autonomous Vehicles (CAVs), real-time semantic segmentation has emerged as a crucial task. The integration of semantic segmentation vision transformers into CAV perception systems capitalizes on the efficiency of the vision transformers and their capacity to capture a global context. To further optimize transformer's performance, we propose the integration of a Split Computing (SC) into the transformer architecture to enable the processing of the computation related to transformers to be distributed between the CAVs and an Edge Computing Server (ECS). The objective of our work is to evaluate SC efficiency in minimizing latency of intensive computational tasks of the transformers with minimal loss in semantic segmentation's accuracy. Simulations demonstrate that vision transformer architectures are well-suited for the SC integration and outperform both ECS-Only and CAV-Only baseline approaches in various scenarios, offering an acceptable latency-accuracy trade-offs. Specifically, our findings indicate that SC outperforms baseline methods in terms of latency by up to 79.36%, incurring a minor accuracy reduction. This makes them particularly suitable for applications where real-time processing and minimal latency are critical considerations.**

***Index Terms*—Connected Autonomous Vehicles, Edge Computing, Semantic Segmentation, Split Computing, Vision Transformers**

I. Introduction

The rise of Connected Autonomous Vehicles (CAVs) signifies a major advancement in the transportation sector, with the potential to revolutionize safety, efficiency, and convenience [1]. The CAVs integrate a diverse range of user-centric applications focused on safety, comfort, and enhanced functionality [2]. To deliver these applications with required performance, real-time processing is paramount. To achieve this, CAVs often leverage connectivity to nearby Edge Computing Server (ECS) co-located with base stations [3]. This network architecture allows CAVs to offload computationally intensive tasks, thereby minimizing execution latency and ensuring timely responses [3]. Optimizing execution latency through effective offloading decisions and resource allocation strategies is an active area of research [4]. These, however, often focus on generic tasks and do not explore more specific tasks, such as computer vision tasks, which are gaining significant traction within CAVs [5]. Compared to the more generic tasks, computer vision tasks frequently utilize Deep Learning (DL) models [6].

The computationally intensive nature of DL models [7] drives research towards Split Computing (SC). This approach strategically partitions the model's workload, with a portion executing on the CAV and the remainder processed by the ECS [8]. The study [9] shows that utilizing SC can significantly improve end-to-end latency, reduce energy consumption on the CAV, and enhance overall datacenter throughput. Despite the advantages of SC, the study [9] identifies a crucial challenge, which is offloading latency. This latency arises from the high volume of intermediate data generated during DL model processing. The transfer of this data between the CAV and the ECS introduces significant delays, potentially compromising the real-time performance. In response to this challenge, current research explores compression of the intermediate data generated at split points within the DL models [10], [11]. One promising approach involves the creation of artificial bottlenecks [8] using autoencoder architectures [12].

Object detection and semantic segmentation are the two main computer vision tasks used in CAVs [13]. SC is efficient in both of these areas [8], but the research on application of the SC to transformer-based models, a rapidly growing field in computer vision [14], remains unexplored. Vision Transformers (ViTs) achieves state-of-the-art performance in semantic segmentation, surpassing Convolutional Neural Networks (CNNs) [15]. This is due to the ViTs ability to capture long-range dependencies using self-attention mechanisms [16]. However, ViTs have a high computational complexity, especially for high-resolution tasks [17]. Architectures like Swin Transformer address this by reducing complexity through hierarchical feature maps and shifted windows [18]. The Swin Transformer's proficiency in handling image features makes it suitable for the autonomous driving perception models [19].

Motivated by the potential benefits of SC and the growing interest in transformer-based models for semantic segmentation in CAVs, this we aim at an evaluation of an applicability of SC for the transformer-based models. To the best of our knowledge, no existing research has investigated integration of the SC within transformer models. Our proposed approach strategically inserts split points within the transformer model to effectively partition the computational workload between the CAV and the ECS. At each split point, we utilize artificial bottlenecks to compress the data for offloading to the ECS. Our objective is to minimize a drop in semantic segmentation's accuracy of our tasks (i.e., execution of the transformer-based DL model) while minimizing the latency of these tasks. Through comprehensive simulations, we evaluate each split point and demonstrate that enabling partial computation offloading via integration of the split points to the transformers can significantly improve latency with a marginal impact on the model's accuracy when compared to compete processing of the transformers solely by the CAV and by the ECS.

The remainder of the paper is organized as follows. Section II defines the system model used for evaluation. Section III details our proposed approach, including the insertion of split points and the creation of artificial bottlenecks. Section IV presents an evaluation of the proposed approach's performance. Finally, Section V summarizes the proposed approach and discusses potential future directions.

II. System model

In this section, we introduce the system model covering communication, computation, and latency aspects adopted in this paper.

The system model adopts a discrete-time approach with time progression denoted by t, where $t \in \{1, 2, \ldots, T\}$, with each time step having a constant duration. We consider N^{CAV} CAVs, each uniquely identified by an index v, where $v \in \{1, 2, \ldots, N^{\text{CAV}}\}$. As the offloading decision within edge computing for transformers falls outside the scope of this study, we posit a single base station equipped with an ECS within our system model, similarly to [20]. Because SC focuses solely on how the DL model is executed, by distributing execution between CAVs and ECS, it seamlessly integrates with offloading decision-making algorithms (i.e., choosing which CAVs are connected to which ECSs).

The computation task generated by CAVs is represented by the transformer-based DL model for semantic segmentation with integrated split points. We define N^{split} split points within the DL model, indexed as s, where $s \in \{1, 2, \ldots, N^{\text{split}}\}$. Similarly to [12], we systematically iterate through these split points and incorporate an artificial bottleneck using autoencoder architectures. This artificial bottleneck comprises channel-reduction and channel-restoration layers. Its purpose is to compress the intermediate features generated by the split point by reducing their channel dimension to a predefined value c, resulting in a lower communication delay for data transfer between CAV and ECS, which is a key component in SC [21].

To assess the performance of the SC-enhanced transformer-based DL model, we adopt semantic segmentation accuracy [22] as the primary metric. This aligns with related works, see e.g., [23]. Semantic segmentation aims to categorize each pixel within an image. Therefore, we initially define accuracy for the d-th image as:

$$\mu_{s,d}^{\text{img}}(c) = \frac{N_{s,d}^{\text{true}}(c)}{N^{\text{pix}}}, \tag{1}$$

where $N_{s,d}^{\text{true}}(c)$ denotes the total number of correctly classified pixels in the d-th image processed by our DL model with integrated artificial bottleneck at the s-th split point with channel-reduction value of c. Furthermore, N^{pix} represents the total number of pixels in the image, which remains constant across all images in the validation dataset. The overall accuracy is then obtained by averaging the individual image accuracies across the validation dataset as:

$$\mu_s(c) = \frac{\sum_{d=1}^{N^{\text{img}}} \mu_{s,d}^{\text{img}}(c)}{N^{\text{img}}} \times 100. \tag{2}$$

where N^{img} represents the total number of images within the validation dataset.

We calculate the communication data rate $\lambda_v(t)$ for transferring the tasks between the v-th CAV and the ECS at the time step t using [24] as:

$$\lambda_v(t) = \eta_v(t)\rho_v(t)\nu(t)\left[\frac{N^{\text{RB}}}{N^{\text{CAV}}}\right], \tag{3}$$

where $\eta_v(t)$ and $\rho_v(t)$ are the number of bits per symbol and the code rate for the v-th CAV, respectively, $\nu(t)$ is the current symbol rate, N^{RB} denotes the number of available resource blocks within the ECS, and N^{CAV} represents the number of CAVs associated with the ECS.

The Signal-to-Interference-Plus-Noise Ratio (SINR) at the time step t is mapped to a modulation and coding scheme to determine the values for $\eta_v(t)$ and $\rho_v(t)$ according to [25].

Using the data rate $\lambda_v(t)$, we calculate the offloading latency of the z-th task as:

$$t_z^{\text{uplink}} = \frac{D_s(c)}{\lambda_v(t)} + t_{z,w}^{\text{uplink}}, \tag{4}$$

where $D_s(c)$ represents the volume of data to be offloaded to the ECS if the s-th split point is selected and the corresponding artificial bottleneck with channel size c. Additionally, $t_{z,w}^{\text{uplink}}$ denotes the time z-th task spent in queues awaiting the start of offloading process. We assume a system model, where the segmentation results of the CAV-generated tasks remain on the ECS for purposes related to aggregated data from multiple CAVs (e.g., traffic analysis) [26]. As a consequence, the size of the output data is notably smaller in comparison to the offloaded data, and therefore, similarly to other related works (see e.g., [27]), we can neglect downlink latency.

During the process of splitting a transformer-based DL model, the inherent consequence is the division of the computational demand associated with the z-th task, denoted by f_z and measured in Tera Floating Point Operations (TFLOPs), into two distinct components: the computational demand on the v-th CAV (i.e., CAV, that generated the z-th task) and the demand on the ECS. This relationship can be mathematically expressed as:

$$f_z = f_z^{\text{CAV}} + f_z^{\text{ECS}}.$$

To compute the processing latency for the v-th CAV $t_{z,v}^{\text{CAV}}$, and the processing latency for the ECS t_z^{ECS} for the z-th task, we employ the following formulas:

$$t_{z,v}^{\text{CAV}} = \frac{f_z^{\text{CAV}}}{F_v^{\text{CAV}}(t)} + t_{z,v,w}^{\text{CAV}}, \tag{5}$$

$$t_z^{\text{ECS}} = \frac{f_z^{\text{ECS}}}{F^{\text{ECS}}(t)} + t_{z,w}^{\text{ECS}}, \tag{6}$$

where $F_v^{\text{CAV}}(t)$ corresponds to the computational power in Tera Floating Point Operations per Second (TFLOPS) of the v-th CAV and $t_{z,v,w}^{\text{CAV}}$ denotes the time spent waiting in the queues before the processing of the z-th task on the v-th CAV begins. The computational power of ECS is denoted by $F^{\text{ECS}}(t)$, and $t_{z,w}^{\text{ECS}}$ corresponds to the time the z-th task spends waiting on ECS before being processed.

The total latency t_z^{total} required to complete the z-th task, including communication latency t_z^{uplink}, CAV processing latency $t_{z,v}^{\text{CAV}}$, and BS processing latency t_z^{ECS}, is calculated as:

$$t_z^{\text{total}} = t_z^{\text{uplink}} + t_{z,v}^{\text{CAV}} + t_z^{\text{ECS}}. \tag{7}$$

III. Proposed Approach

This paper investigates the effectiveness of SC in transformer-based DL models for semantic segmentation. In this section, we propose a method for inserting splits, integrating artificial bottlenecks for compression, and determining optimal channel-reduction c for latency-accuracy trade-off. We further address limitation of SC in transformers and propose mitigation strategy.

The transformer-based DL models for semantic segmentation are typically divided into distinct logically connected blocks, commonly referred to as stages [18], [28], [29]. To maintain each stages integrity, we avoid splitting within stages and insert split points between consecutive stages. We then add artificial bottlenecks at these points, configuring them with a channel-reduction value (c). This value is crucial as it impacts latency and accuracy. Our proposed Algorithm 1 integrates these bottlenecks and determines optimal channel-reduction values (C^*)$usingascoringmethodologythatbalancesaccuracyandlatency.Th$

Algorithm 1 Configuration of Artificial Bottlenecks using Scoring Methodology

1: **Input:** Trained SC-enhanced transformer-based DL model, number of splits N^{split}
2: $\mathcal{C}^* \leftarrow \{\}$
3: **for** each split $s \in \{1, \ldots, N^{\text{split}}\}$ **do**
4: $\quad \alpha_s^*(c) \leftarrow -\infty$
5: $\quad$ **for** each $c \in \mathcal{C}_s$ **do**
6: $\quad\quad$ Insert an artificial bottleneck to the s-th split point of DL model
7: $\quad\quad$ Train inserted artificial bottleneck according to [12]
8: $\quad\quad$ compute accuracy $\mu_s(c)$ according to (2)
9: $\quad\quad$ compute data size $D_s(c)$ for offloading
10: $\quad\quad \mu'_s(c) \leftarrow \frac{\mu_s(c) - \mu_s^{\min}}{\mu_s^{\max} - \mu_s^{\min}}$
11: $\quad\quad D'_s(c) \leftarrow \frac{D_s(c) - D_s^{min}}{D_s^{max} - D_s^{min}}$
12: $\quad\quad \alpha_s(c) \leftarrow w^{\text{data}}(1 - D'_s(c)) + w^{\text{acc}}\mu'_s(c)$
13: $\quad\quad$ **if** $\alpha_s(c) > \alpha_s^*(c)$ **then**
14: $\quad\quad\quad \alpha_s^*(c) \leftarrow \alpha_s(c)$
15: $\quad\quad\quad c^* \leftarrow c$
16: $\quad\quad$ **end if**
17: $\quad$ **end for**
18: $\quad \mathcal{C}^* \leftarrow \mathcal{C}^* \cup \{c^*\}$
19: **end for**
20: **Output:** Set of optimal channel-reduction values $\mathcal{C}^*$

At input, we expect a trained SC-enhanced transformer-based DL model for semantic segmentation with N^{split} split points (line 1). First, we initialize an empty set of optimal channel-reduction val'ues $\mathcal{C}^*$ (line 2). Then, we iterate over each split point indexed with s (line 3). We initialize the s-th split's best score (score representing the accuracy-latency trade-off) $\alpha_s^*(c)$ to $-\infty$ (line 4). This ensures that any initial score encountered during the search will be considered better and replace negative infinity. Next, we insert an artificial bottleneck at the s-th split point in the DL model and train it using the channel-reduction value $c \in \mathcal{C}_s$, where $\mathcal{C}_s$ includes all

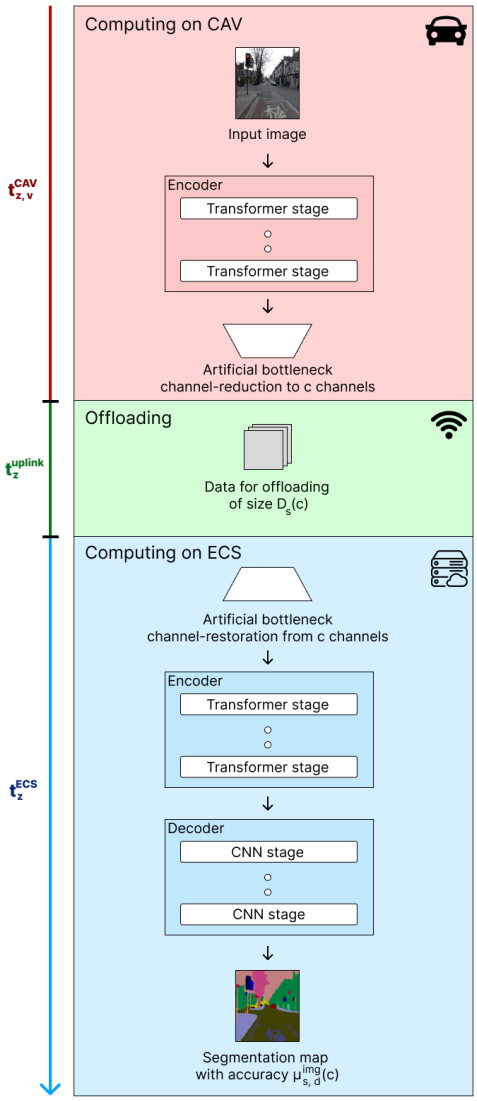


Fig. 1: Scheme of SC-enhanced transformer-based DL model for semantic segmentation with integrated artificial bottleneck at s-th split point

possible channel-reduction values for the artificial bottleneck at the s-th split point. These values range from 1 to the number of channels in the intermediate features entering the artificial bottleneck, excluding the upper bound (lines 6 and 7). Next, we calculate the accuracy $\mu_s(c)$ of the DL model with the inserted artificial bottleneck (line 8). We then compute the amount of data $D_s(c)$ to be offloaded from the CAV to the ECS (line 9), which changes based on current split index s and the channel-reduction value c. The data size $D_s(c)$ for offloading is equal to the size of the output matrix from the channel-reduction layers of the corresponding artificial bottleneck. Next, we normalize the accuracy $\mu_s(c)$ and data size for offloading $D_s(c)$ using min-max normalization [30] with minimal and maximal possible values for accuracy in range $[\mu_s^{\min}, \mu_s^{\max}]$ and for data size for offloading in range $[D_s^{min}, D_s^{max}]$. In this context, D_s^{min} represents the size of the output from the artificial bottleneck's channel-reduction layers when $c = 1$, and D_s^{max} corresponds to the size of the output

from the artificial bottleneck's channel-reduction layers when c is set to the highest possible channel-reduction value for this artificial bottleneck, integrated within the s-th split point (lines 10 and 11). The variables $\mu'_s(c)$ and $D'_s(c)$ denotes the normalized accuracy and the normalized data size for offloading for the s-th split point, respectively. The score $\alpha_s(c)$ is calculated based on a weighted sum of normalized accuracy $\mu'_s(c)$ and normalized data size for offloading $D'_s(c)$ (line 12). Note, that we subtract 1 from normalized data size $D'_s(c)$ to ensure a lower data size contributes to a higher overall score. The weights w^{acc} and w^{data}, which range from 0 to 1, determine the priority assigned to normalized accuracy $\mu'_s(c)$ and normalized data size for offloading $D'_s(c)$, respectively. A higher weight value results in a higher priority for the corresponding criteria. If the computed score $\alpha_s(c)$ is greater than the current best score $\alpha^*_s(c)$, we update best score $\alpha^*_s(c)$ to the value of the current score $\alpha_s(c)$ and assign the corresponding channel-reduction value c to the optimal channel-reduction value c^* (lines 13-16). Then, we add the optimal channel-reduction value c^* for the s-th split point's artificial bottleneck to the set $\mathcal{C}^*$ (line 18). Finally, the algorithm outputs the set of the best channel-reduction values $\mathcal{C}^*$ (line 20).

The stages leading up to s-th split point, along with the artificial bottleneck channel-reduction layers, are computed on the CAV. The stages following the s-th split point, along with the artificial bottleneck channel-restoration layers, are computed on the ECS. These computational dependencies are illustrated in Fig.1.

Semantic segmentation DL models often employ an encoder-decoder architecture, including lateral connections between corresponding encoder and decoder stages. These connections are utilized to transfer information for the upsampling process, such as pooling indices [31] or skip connections for intermediate feature maps [18], [28], [29]. However, [32] demonstrates that skip connections provide a marginal benefit. Moreover, this approach presents challenges for semantic segmentation SC-enhanced DL models, as transferring large amounts of data between CAVs and ECS is often impractical. Therefore, for purposes of SC, we propose using a hybrid architecture featuring a transformer encoder and a CNN decoder [33] with no such additional information transferred, similar to [34].

After the implementation of the proposed changes to the transformer-based DL model, we have architecture featuring integrated splits between DL model stages. Each split incorporates an artificial bottleneck, with the channel-reduction value c strategically chosen based on a latency-accuracy trade-off analysis as outlined in Algorithm 1. This enables a systematic evaluation of the effectiveness of each split point by measuring the corresponding latency and accuracy metrics.

IV. Performance Evaluation

In this section, we evaluate the performance of the proposed approach. We start by describing the experimental setup, which includes both the training and simulation configurations. Next, we assess the process of configuring the artificial bottlenecks. Finally, we evaluate the effectiveness of the proposed approach through simulation across various environmental conditions.

A. Simulation Setup

Our SC-enhanced transformer-based DL model is trained using the CamVid dataset [35], [36], which serves as a benchmark for semantic segmentation [15], [37]. As encoder, we employ the smallest Swin transformer variant (Swin-T) for its computational efficiency. We pair Swin encoder with a progressive upsampling convolutional decoder, similar to [34]. The convolutional decoder architecture employed is the decoder component of SegNet [31], with modifications to the upsampling process as detailed in Section III. The first four splits are located after transformer encoder stages, while the rest of the splits are located between the stages of the convolutional decoder. For training of artificial bottlenecks, we use batch size of 8, AdamW optimizer with learning rate of 0.001 and PolynomialLR scheduler with the power of 0.9. These hyperparameters are chosen using grid search, as in [38].

The primary simulation parameters are listed in Table I. The computational power of the ECS is set to 41.29 TFLOPS, simulating a high-capacity ECS capable of handling higher volume of tasks. The symbol rate $\nu(t)$ is fixed at 0.168. Each second, 10 tasks are generated. We evaluate the proposed approach by adjusting the computational power of CAVs $F_v^{\text{CAV}}(t)$, using values 1.1, 4.1 and 7.1 TFLOPS, reflecting varying CAVs computational capabilities. Similarly, we investigate the impact of number of resource blocks N^{RB}, using values 1000, 2000, and 4000, allowing for the analysis of performance under different network capacities. The simulation area emulates an urban environment, with CAVs following the Manhattan mobility model [39]. The results are averaged over 10 simulations, each with 5000 steps.

TABLE I: Edge Simulation Configuration

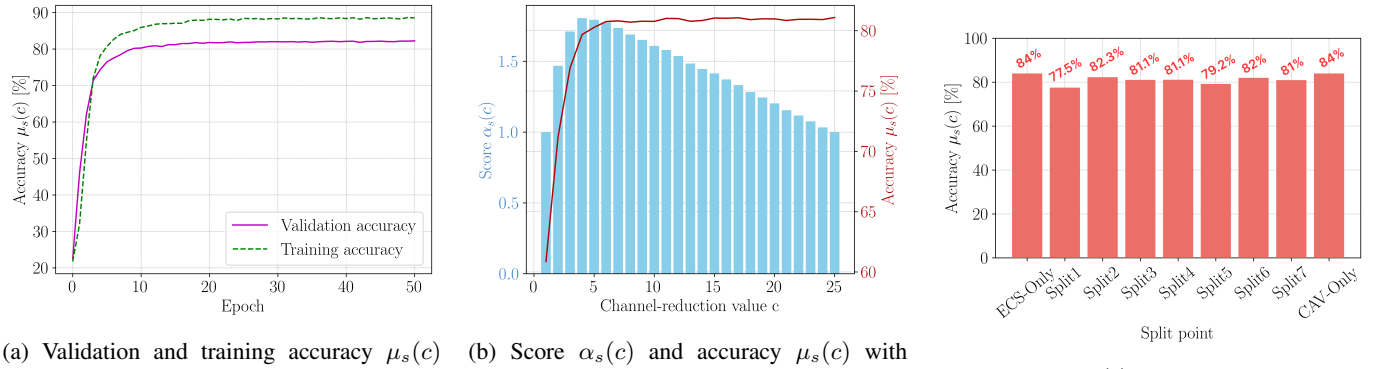
Parameter	**Value**
Number of Simulations	10
Number of steps per Simulation (T)	5000
Number of CAVs (N^{CAV})	10
Computational Power of ECS (f_z^{ECS})	41.29
Symbol Rate ($\nu(t)$)	0.168
Number of Tasks Generated per Second	10
Number resource blocks (N^{RB})	[1000, 2000, 4000]
Computational power of CAVs ($F_v^{\text{CAV}}(t)$)	[1.1, 4.1, 7.1]

We evaluate the proposed approach against two baseline approaches that do not integrate splits into the DL model:

1) **CAV-Only Approach**: The entire DL model is computed solely on the CAV without any offloading.
2) **ECS-Only Approach**: The entire DL model is computed solely on the ECS after transferring all data from a CAV to the ECS.

B. Artificial Bottleneck Configuration

We present the results of configuring artificial bottlenecks for each split point within our SC-enhanced transformer-based DL model. This DL model represents the tasks for which the CAVs require computation results. To achieve this, we leverage Algorithm 1 to identify the optimal channel-reduction values $\mathcal{C}^*$. The weights assigned to both accuracy w^{acc} and



(a) Validation and training accuracy $\mu_s(c)$ curves computed during artificial bottleneck training in the second split point with $c = 4$

(b) Score $\alpha_s(c)$ and accuracy $\mu_s(c)$ with respect to channel-reduction values $c \in \mathcal{C}_s$ for the second split point

(c) Accuracy $\mu_s(c)$ of each split point and baselines

Fig. 2: Score $\alpha_s(c)$ and accuracy $\mu_s(c)$ analysis for the configuration of artificial bottlenecks

transferred data size w^{data} are set equally to 1. Fig. 2 illustrates the obtained results.

Fig. 2a depicts the training process for an artificial bottleneck implemented within the second split point, employing a channel-reduction value of $c = 4$. To conserve space, only this specific training process is presented, however, all other artificial bottlenecks exhibit similar training patterns. Each bottleneck undergoes training for 50 epochs, as demonstrated in Fig. 2a, which establishes a well-balanced trade-off between training time and attained accuracy $\mu_s(c)$. Following the training of each artificial bottleneck, the score $\alpha_s(c)$ is calculated. This score $\alpha_s(c)$ is a function of both the achieved accuracy $\mu_s(c)$ and the potential data size $D_s(c)$ that CAVs would need to offload to the ECS.

The achieved score $\alpha_s(c)$ and accuracy $\mu_s(c)$ for each channel-reduction value $c \in \mathcal{C}_s$ within the second split point are presented in Fig. 2b. The data size $D_s(c)$ is excluded from the figure due to its straightforward positive linear correlation with the channel-reduction values $c \in \mathcal{C}_s$. Furthermore, to maintain conciseness, only the initial 25 channel-reduction values $c \in \mathcal{C}_s$ are displayed, with the remaining values in $\mathcal{C}_s$ exhibiting the same decreasing trend for score $\alpha_s(c)$.

Finally, Fig. 2c shows the accuracy, $\mu_s(c)$ (where $c \in \mathcal{C}^*$ for all split points), for the proposed approach and the baselines. As observed in Fig. 2c, both the ECS-Only and CAV-Only baslines achieve an accuracy $\mu_s(c)$ of 83.98%, as both do not utilize lossy compression through artificial bottlenecks. Nonetheless, each split point exhibits small drop in accuracy $\mu_s(c)$, with the first split point achieving the lowest accuracy $\mu_s(c)$ of 77.52%.

C. Modeling Latency of Proposed Network via Edge Simulation Environment

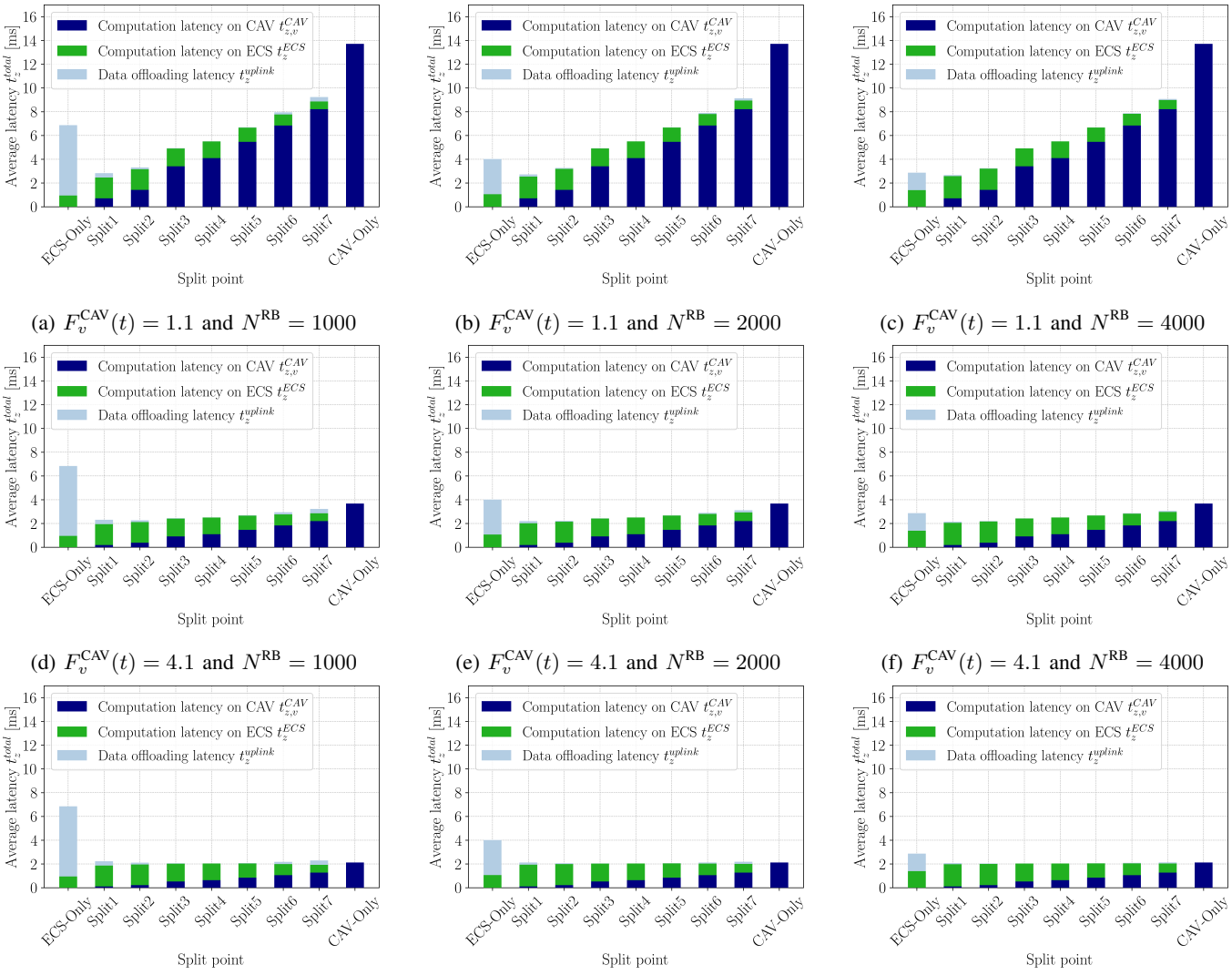
To evaluate the effectiveness of different split points, we conducted a series of simulations across a range of system parameters. We assessed the average execution latency t_z^{total} of tasks under different splitting configurations. Additionally, we benchmarked the performance against CAV-Only and ECS-Only baselines. The results of these simulations are presented in Fig. 3.

As depicted in Fig. 3a, SC demonstrates its greatest advantage in resource-constrained environments. The first split point achieves a significant latency t_z^{total} reduction compared to both baselines, exhibiting 58.81% improvement against the ECS-Only baseline and 79.36% against the CAV-Only baseline. However, due to the inherently high latency of CAV computations $t_{z,v}^{\text{CAV}}$, only the first four splits outperform ECS-Only baseline. As the number of resource blocks N^{RB} increases (refer to Figs. 3b and 3c), the ECS-Only baseline becomes increasingly competitive. Consequently, fewer split points remain viable options for achieving lower latencies t_z^{total} compared to ECS-Only baseline. However, even in scenarios where the ECS-Only baseline achieves comparable latency t_z^{total} to the first split point (as observed in Fig. 3c), SC might still be preferable due to its potential privacy benefits, as elaborated in [40].

An increase in the computational capabilities of CAVs $F_v^{\text{CAV}}(t)$ reveals two key trends. First, the CAV-Only baseline becomes increasingly viable, achieving latency values t_z^{total} comparable to SC. Second, a larger number of split points outperform the baselines. Across all split points depicted in Figs. 3d, 3e, and 3f, SC exhibits latency t_z^{total} that is either lower or comparable to the baselines. Specifically, we observe improvements in latency t_z^{total} of up to 66.81% against the ECS-Only baseline and up to 41.58% against the CAV-Only baseline. Finally, as evident in Figs. 3g, 3h, and 3i, with a further increase in CAVs computational power $F_v^{\text{CAV}}(t)$, the CAV-Only baseline closes the performance gap even surpassing certain split points.

V. Conclusion

In this work, we have investigated the integration of SC into a transformer-based DL model for semantic segmentation. We have proposed an algorithm to identify optimal channel-reduction values for artificial bottlenecks, enabling data compression for offloading. This algorithm leverages a scoring function that incorporates a weighted sum of accuracy and data size. Our results showcase the effectiveness of configuring these artificial bottlenecks within a Swin Transformer-based DL model. Specifically, we employ a hybrid transformer-CNN DL model, where the Swin Transformer serves as the encoder



(a) $F_v^{\text{CAV}}(t) = 1.1$ and $N^{\text{RB}} = 1000$ (b) $F_v^{\text{CAV}}(t) = 1.1$ and $N^{\text{RB}} = 2000$ (c) $F_v^{\text{CAV}}(t) = 1.1$ and $N^{\text{RB}} = 4000$

(d) $F_v^{\text{CAV}}(t) = 4.1$ and $N^{\text{RB}} = 1000$ (e) $F_v^{\text{CAV}}(t) = 4.1$ and $N^{\text{RB}} = 2000$ (f) $F_v^{\text{CAV}}(t) = 4.1$ and $N^{\text{RB}} = 4000$

(g) $F_v^{\text{CAV}}(t) = 7.1$ and $N^{\text{RB}} = 1000$ (h) $F_v^{\text{CAV}}(t) = 7.1$ and $N^{\text{RB}} = 2000$ (i) $F_v^{\text{CAV}}(t) = 7.1$ and $N^{\text{RB}} = 4000$

Fig. 3: Results of edge simulation with variable number of resource blocks N^{RB} and CAV computational power $F_v^{\text{CAV}}(t)$ in TFLOPS.

and the CNN acts as the decoder. Furthermore, we conduct a comprehensive evaluation of the most effective artificial bottleneck configurations within a simulated environment by varying key system parameters to assess their performance. The proposed approach demonstrates superior performance compared to the established ECS-Only and CAV-Only baselines across diverse scenarios. This translates to achieving a lower or comparable latency while maintaining only very small accuracy loss. These findings convincingly demonstrate the suitability of leveraging SC for transformer-based architectures and their potential in real-time applications and latency-critical domains.

Acknowledgment

This work was supported by The Slovak Research and Development Agency project no. APVV SK-CZ-RD-21-0028, APVV-23-0512, the Slovak Academy of Sciences project no. VEGA 1/0685/23, the Ministry of Education, Youth and Sports, Czech Republic, under the project LUASK22064 and the National Competence Centre for High-Performance Computing (project code: 311070AKF2).

Icons used in this work were made by Freepik and Pixel perfect from www.flaticon.com.

References

[1] P. Kopelias, E. Demiridi, K. Vogiatzis, A. Skabardonis, and V. Zafiropoulou, "Connected & autonomous vehicles–environmental impacts–a review," *Science of the total environment*, vol. 712, p. 135237, 2020.

[2] A. Dabboussi, "Dependability approaches for mobile environment: Application on connected autonomous vehicles." Ph.D. dissertation, Université Bourgogne Franche-Comté, 2019.

[3] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge computing for autonomous driving: Opportunities and challenges," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1697–1716, 2019.

[4] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE communications surveys & tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[5] J. Janai, F. Güney, A. Behl, A. Geiger *et al.*, "Computer vision for autonomous vehicles: Problems, datasets and state of the art," *Foundations and Trends® in Computer Graphics and Vision*, vol. 12, no. 1–3, pp. 1–308, 2020.

[6] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational intelligence and neuroscience*, vol. 2018, no. 1, p. 7068349, 2018.

[7] C. Chen, P. Zhang, H. Zhang, J. Dai, Y. Yi, H. Zhang, and Y. Zhang, "Deep learning on computational-resource-limited platforms: A survey," *Mobile Information Systems*, vol. 2020, no. 1, p. 8454327, 2020.

[8] Y. Matsubara, M. Levorato, and F. Restuccia, "Split computing and early exiting for deep learning applications: Survey and research challenges," *ACM Computing Surveys*, vol. 55, no. 5, pp. 1–30, 2022.

[9] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *SIGARCH Comput. Archit. News*, vol. 45, no. 1, p. 615629, apr 2017. [Online]. Available: https://doi.org/10.1145/3093337.3037698

[10] J.-D. Guerrero-Balaguera, J. E. R. Condia, M. Levorato, and M. S. Reorda, "Evaluating the reliability of supervised compression for split computing," in *2024 IEEE 42nd VLSI Test Symposium (VTS)*. IEEE, 2024, pp. 1–6.

[11] Z. Yuan, S. Rawlekar, S. Garg, E. Erkip, and Y. Wang, "Split computing with scalable feature compression for visual analytics on the edge," *IEEE Transactions on Multimedia*, 2024.

[12] A. E. Eshratifar, A. Esmaili, and M. Pedram, "BottleNet: A deep learning architecture for intelligent mobile cloud computing services," 2019.

[13] W. Xu, B. Li, S. Liu, and W. Qiu, "Real-time object detection and semantic segmentation for autonomous driving," 02 2018, p. 44.

[14] S. Jamil, M. Jalil Piran, and O.-J. Kwon, "A comprehensive survey of transformers for computer vision," *Drones*, vol. 7, no. 5, p. 287, 2023.

[15] J. Wang, C. Gou, Q. Wu, H. Feng, J. Han, E. Ding, and J. Wang, "RTFormer: Efficient design for real-time semantic segmentation with transformer," 2022.

[16] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in vision: A survey," *ACM computing surveys (CSUR)*, vol. 54, no. 10s, pp. 1–41, 2022.

[17] D. P. Pau and F. M. Aymone, "Mathematical formulation of learning and its computational complexity for transformers layers," *Eng*, vol. 5, no. 1, pp. 34–50, 2023.

[18] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," 2021.

[19] P. Deshmukh, G. Satyanarayana, S. Majhi, U. K. Sahoo, and S. K. Das, "Swin transformer based vehicle detection in undisciplined traffic environment," *Expert Systems with Applications*, vol. 213, p. 118992, 2023.

[20] Z. Zhao, K. Wang, N. Ling, and G. Xing, "EdgeML: An AutoML framework for real-time deep learning on the edge," in *Proceedings of the international conference on internet-of-things design and implementation*, 2021, pp. 133–144.

[21] Y. Matsubara, M. Levorato, and F. Restuccia, "Split computing and early exiting for deep learning applications: Survey and research challenges," 2022.

[22] M. Thoma, "A survey of semantic segmentation," *CoRR*, vol. abs/1602.06541, 2016. [Online]. Available: http://arxiv.org/abs/1602.06541

[23] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "TextonBoost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation," 07 2006, pp. 1–15.

[24] J. Plachy, Z. Becvar, E. C. Strinati, and N. d. Pietro, "Dynamic allocation of computing and communication resources in multi-access edge computing for mobile users," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 2089–2106, 2021.

[25] [Online]. Available: https://www.etsi.org/deliver/etsi_ts/138200_138299/138211/16.02.00_60/ts_138211v160200p.pdf

[26] F. Malandrino, C.-F. Chiasserini, and G. M. Dell'Aera, "Edge-Powered assisted driving for connected cars," *IEEE Transactions on Mobile Computing*, vol. 22, pp. 874–889, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:235743022

[27] M. Vološin, E. Šlapak, Z. Becvar, T. Maksymyuk, A. Petík, M. Liyanage, and J. Gazda, "Blockchain-based route selection with allocation of radio and computing resources for connected autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[28] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Álvarez, and P. Luo, "SegFormer: Simple and efficient design for semantic segmentation with transformers," *CoRR*, vol. abs/2105.15203, 2021. [Online]. Available: https://arxiv.org/abs/2105.15203

[29] Z. Geng, L. Liang, T. Ding, and I. Zharkov, "RSTT: Real-time spatial temporal transformer for space-time video super-resolution," 03 2022.

[30] S. G. K. Patro and K. K. Sahu, "Normalization: A preprocessing stage," *CoRR*, vol. abs/1503.06462, 2015. [Online]. Available: http://arxiv.org/abs/1503.06462

[31] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," 2016.

[32] A. Kamath, J. Willmann, N. Andratschke, and M. Reyes, "Do we really need that skip-connection? understanding its interplay with task complexity," in *Medical Image Computing and Computer Assisted Intervention – MICCAI 2023*, H. Greenspan, A. Madabhushi, P. Mousavi, S. Salcudean, J. Duncan, T. Syeda-Mahmood, and R. Taylor, Eds. Cham: Springer Nature Switzerland, 2023, pp. 302–311.

[33] A. Khan, Z. Rauf, A. Sohail, A. R. Khan, H. Asif, A. Asif, and U. Farooq, "A survey of the vision transformers and their CNN-transformer based variants," *Artificial Intelligence Review*, vol. 56, no. S3, p. 29172970, Oct. 2023. [Online]. Available: http://dx.doi.org/10.1007/s10462-023-10595-0

[34] S. Zheng, J. Lu, H. Zhao, X. Zhu, Z. Luo, Y. Wang, Y. Fu, J. Feng, T. Xiang, P. H. S. Torr, and L. Zhang, "Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers," *CoRR*, vol. abs/2012.15840, 2020. [Online]. Available: https://arxiv.org/abs/2012.15840

[35] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla, "Segmentation and recognition using structure from motion point clouds," in *ECCV (1)*, 2008, pp. 44–57.

[36] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognition Letters*, vol. 30, no. 2, pp. 88–97, 2009.

[37] J. Xu, Z. Xiong, and S. P. Bhattacharyya, "PIDNet: A real-time semantic segmentation network inspired by PID controllers," 2023. [Online]. Available: https://arxiv.org/abs/2206.02066

[38] J. Hanna, M. Mommert, and D. Borth, "Sparse multimodal vision transformer for weakly supervised semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2023, pp. 2145–2154.

[39] A. Hanggoro and R. F. Sari, "Performance evaluation of the manhattan mobility model in vehicular ad-hoc networks for high mobility vehicle," in *2013 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT)*. IEEE, 2013, pp. 31–36.

[40] T. Nishio, K. Yorita, S. Ohta, K. Maejima, K. Kodera, Y. Horikawa, and K. Fukui, "Split computing-based privacy-preserving image classification and object detection," in *2024 IEEE 21st Consumer Communications & Networking Conference (CCNC)*. IEEE, 2024, pp. 1092–1093.