Task-Oriented Video Compressive Streaming for Real-time Semantic Segmentation

Xuedou Xiao, Yingying Zuo, Mingxuan Yan, Wei Wang, *Senior Member, IEEE*, Jianhua He, *Senior Member, IEEE,* Qian Zhang, *Fellow, IEEE*

***Abstract*—Real-time semantic segmentation (SS) is a major task for various vision-based applications such as self-driving. Due to the limited computing resources and stringent performance requirements, streaming videos from camera-embedded mobile devices to edge servers for SS is a promising approach. While there are increasing efforts on task-oriented video compression, most SS-applicable algorithms apply more uniform compression, as the sensitive regions are less obvious and concentrated. Such processing results in low compression performance and significantly limits the capacity of edge servers supporting real-time SS. In this paper, we propose STAC, a novel task-oriented DNN-driven video compressive streaming algorithm tailed for SS, to strike accuracy-bitrate balance and adapt to time-varying bandwidth. It exploits DNN's gradients as sensitivity metrics for fine-grained spatial adaptive compression and includes a temporal adaptive scheme that integrates spatial adaptation with predictive coding. Furthermore, we design a new bandwidth-aware neural network, serving as a compatible configuration tuner to fit time-varying bandwidth and content. STAC is evaluated in a system with a commodity mobile device and an edge server with real-world network traces. Experiments show that STAC can save up to 63.7-75.2% of bandwidth or improve accuracy by 3.1-9.5% compared to state-of-the-art algorithms, while capable of adapting to time-varying bandwidth.**

***Index Terms*—DNN-driven compression, adaptive streaming, semantic segmentation, edge computing.**

I. Introduction

With the proliferation of artificial intelligence and large-scale camera deployments, vision-based applications have become ubiquitous in modern society. These applications include object detection (OD) for public safety surveillance [2], semantic segmentation (SS) for self-driving cars/drones [3], [4], AR/VR devices [5], as well as virtual try-on/make-up apps [6], [7]. For example, SS provides pixel-level semantic classification for images/videos, enabling self-driving cars to separate pedestrians, vehicles, and traffic signs for improved safety and efficiency. However, such applications typically demand significant resources that surpass the capabilities of a large portion of camera-embedded mobile devices, particularly in meeting latency and accuracy requirements. This drives camera videos to be streamed over wireless networks to resource-rich edge/cloud servers for real-time processing [8]. However, with the fast-increasing number of vision-based applications and devices, limited and time-varying network bandwidth becomes the most pressing challenge.

To address this challenge, task-oriented video compression techniques [9]–[19] have undergone rapid evolution in recent years, aiming at minimizing bandwidth consumption. Instead of catering to human perception, these techniques prioritize the accuracy and real-time performance of video analytics tasks as their new targets. Despite their potential, these techniques however present significant development imbalance across OD tasks [9]–[11] and SS tasks [12]–[19], both of which are building blocks of modern vision-based applications. The inherent reason is that SS tasks typically require classifying each pixel across the entire image/frame plane, including semantic boundary delimitation. Therefore, the sensitive regions for SS tasks, which contribute more to the final accuracy, are generally scattered and ambiguous, compared to OD tasks

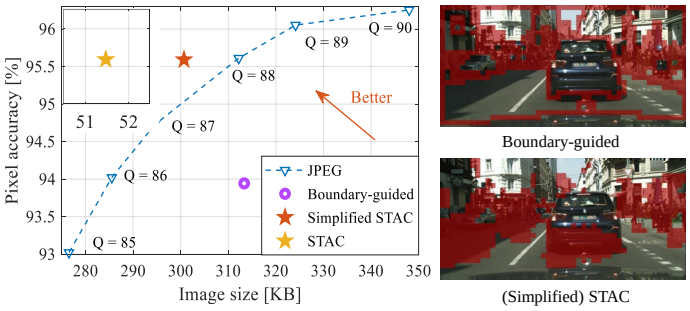


Fig. 1: A preliminary comparison of boundary-guided compression, JPEG and STAC. To ensure fairness, JPEG, boundary-guided compression, and the simplified STAC process consecutive frames as individual images, with the latter two algorithms employing the same two levels of compression ratios: the lower ratio is assigned to the red regions in the images, while the higher ratio is applied to the rest. Additionally, the performance of the full-version STAC is also showcased, where STAC compresses consecutive frames using our proposed spatio-temporal algorithm.

Part of this work has been presented at IEEE INFOCOM 2022 [1].

This work was supported in part by the National Natural Science Foundation of China with Grants 62071194, 62402353, the Knowledge Innovation Program of Wuhan-Shuguang, the European Union's Horizon 2020 programme under grant No 824019 and 101022280, the Horizon European program under grant No 101086228, EPSRC with RC Grant No EP/Y027787/1 and UKRI under grant No EP/Y028317/1, the Fundamental Research Funds for the Central Universities(WUT: 104972024RSCbs0045). *(Corresponding author: Wei Wang.)*

X. Xiao is with the School of Navigation, Wuhan University of Technology, Wuhan, China (e-mail:xuedouxiao@whut.edu.cn).

Y. Zuo, M. Yan, W. Wang are with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, China (e-mail: {yingyingzuo, mingxuanyan, weiwangw}@hust.edu.cn).

J. He is with the School of Computer Science and Electronic Engineering, Essex University, Colchester CO4 3SQ, U.K. (e-mail:j.he@essex.ac.uk).

Q. Zhang is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong (e-mail:qianzh@cse.ust.hk).

where the regions of interest (RoIs) are clear and concentrated. Many SS-applicable researches [12]–[16] thus apply uniform spatial compression strategies to the whole image/frame plane (such as GRACE [12] and auto-encoder methods [14]–[16]) or simply discriminate compression ratios according to human visual quality (such as H.26x [20]). In this case, the compression ratio of the whole image/frame is sacrificed to ensure accuracy, which leads to severe bandwidth wastage and significantly limits the capacity of edge servers supporting real-time SS. As such, a compressive streaming algorithm dedicated to SS is urgently needed to minimize bandwidth consumption while not sacrificing accuracy.

One may hypothesize whether the sensitive regions of SS align with segment boundaries, following human perception. We therefore conduct a preliminary experiment by assigning a lower compression ratio to regions around boundaries and a higher compression ratio to others on individual images. The results in Fig. 1 show that the point representing boundary-guided compression falls entirely below the curve corresponding to JPEG. It indicates that the heuristic boundary-guided compression performs even worse than the uniform compression in the accuracy-size trade-off. Specifically, it leads to larger sizes for the same accuracy and lower accuracy for the same sizes.

To accurately detect the sensitive regions, we present STAC, a new video compressive streaming algorithm tailored for semantic segmentation, achieving the best accuracy-bitrate balance and adapting to time-varying available network bandwidth. The core idea behind is to use the target deep neural network (DNN)'s gradients to as spatial sensitivity metrics (i.e., the importance degree), so as to customize a non-uniform compression strategy for the content. Specifically, the gradient quantifies the DNN loss increment during lossy compression process, allowing for non-uniform compression based on the maximum compression ratio that each pixel can withstand. Furthermore, a bandwidth-aware configuration adaptive scheme is proposed, compatible with this compression, to maximize accuracy under time-varying available bandwidth. The system architecture is demonstrated in Fig. 2.

The design of STAC involves three technical challenges.

(i) How to minimize the bandwidth wastage brought by real-time synchronization of compression strategy? It is noteworthy that this DNN-driven compression strategy is generated online at servers, requiring continuous feedback and transmission between transceivers to support (de)compression. However, a pixel-level compression strategy consumes huge bandwidth. To tackle this problem, we propose a spatial adaptive scheme, which no longer targets the pixel-based compression strategy, but uses regions of frames as the basic unit. Additionally, STAC brings part of the online strategy generation process offline by designing multi-level block-based compression strategies based on average gradients. In this regard, only the levels of compression strategies selected by each region need to be fed back and transmitted.

(ii) How to adapt the compression strategy to changing video content? As the spatial sensitivity continuously changes with video content, the content-customized strategy can neither be applied to already-transmitted frames (based on which the strategy is derived at servers), nor the subsequent frames. To address this issue, we propose a temporal adaptive scheme that first extracts the dense optical flow to propagate both compression strategy and SS results across frames; and then integrates the spatial adaptive compression strategy with inter-/intra-frame predictive coding. In this respect, the bandwidth can be saved from all three dimensions, including spatial adaptive compression, inter-/intra-frame compression and frame rate reduction.

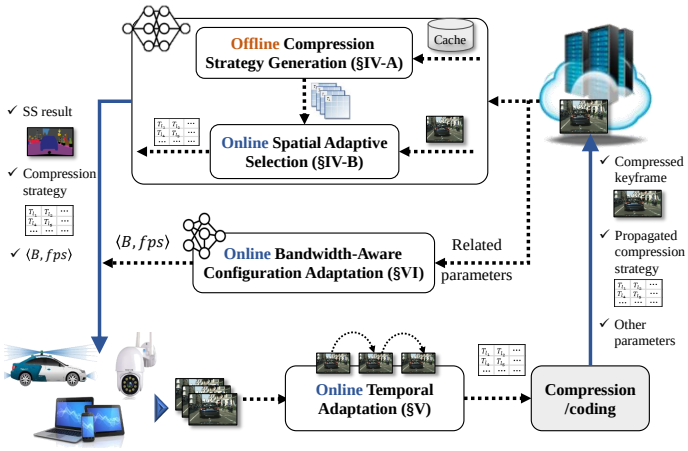


Fig. 2: System architecture of STAC.

(iii) How to adapt the spatio-temporal adaptive compression strategy to time-varying available bandwidth? No existing configuration/bitrate adaptive works [21]–[24] can directly fit STAC, as quantization steps of all regions and frame rates need to be dynamically adjusted according to content and bandwidth, covering a huge action space. To cope with this problem, we propose a novel bandwidth-aware configuration adaptive scheme compatible with STAC, which converges the large-dimensional action space to a two-dimensional vector including the upperbound B of the DNN loss function and the frame rate fps. This scheme builds on a supervised-trained NN-based B/fps-tuner, which uses offline generated B/fps scaling labels and takes as input bandwidth, historical B/fps, and historical content-related metrics such as frame complexity and variance.

We implement STAC on a portable and small form factor Intel NUC Kit NUC7i5DNHE and run DNN inference on an edge server equipped with an Nvidia Tesla T4 GPU. We compare STAC with state-of-the-art compressive streaming algorithms including GRACE [12], CASVA [21] and AccMPEG [18]. The experiments cover 3 target DNN models, 3 semantic segmentation datasets and various types of networks including LTE/4G [25], FCC [26] and WiFi. Evaluation results show that STAC is able to achieve superior accuracy-bitrate balance by saving up to 63.7-75.2% of bandwidth or improving 3.1-9.5% of accuracy, compared to state-of-the-art algorithms. Meanwhile, STAC can also fit time-varying available bandwidth while ensuring real-time semantic segmentation.

The main contributions are summarized as follows.

- To our knowledge, STAC is the first fine-grained multi-level non-uniform compressive streaming algorithm tai-

lored for SS. Customized on DNN's spatial sensitivity metrics and video content, task-oriented STAC strikes the best accuracy-bitrate balance.
- We propose a temporal adaptive scheme that propagates spatial adaptive compression strategies across frames, and integrates such spatial adaptation with inter-/intra-frame predictive coding.
- We create a new bandwidth-aware configuration adaptive scheme compatible with our compression algorithm to guarantee ultra-low delay and high accuracy under dynamic bandwidth.
- Benchmark tests on 3 target DNNs, 3 SS datasets, and various types of networks demonstrate the superiority of STAC compared to state-of-the-art algorithms.

The rest of this paper is organized as follows. §II introduces the background knowledge of SS and video codecs. §III overviews the system design of STAC. §IV and §V elaborate on the spatial adaptive compression strategy and temporal adaptive scheme. §VI presents the compatible bandwidth-aware configuration adaptive algorithm. Implementation and evaluation are detailed in §VII and §VIII, respectively. §IX gives a literature review, followed by some discussions in §X and the conclusion in §XI.

II. Primers on Semantic Segmentation and Video Codecs

In this section, we give a brief introduction to semantic segmentation and the typical video codec framework, which lays the foundation for a better understanding of STAC.

A. Semantic Segmentation

Semantic segmentation refers to the vision task of assigning specific class labels to each pixel in an image/frame, thereby dividing it into meaningful segments. The most notable characteristic is providing pixel-level semantic understanding, including precise boundary delineation, which distinguishes it from bounding box-based tasks like OD. This distinction also makes it more challenging to generate SS-oriented compression algorithms. Unlike OD tasks where RoIs are clear and concentrated, SS tasks often involve scattered and ambiguous sensitive regions, as they generally require understanding each pixel across the entire plane. This motivates the proposal of STAC, a compressive streaming algorithm tailored for SS.

Semantic segmentation has been widely applied in various fields such as automatic driving [3], [4], AR/VR devices [5], virtual try-on/make-up apps [6], [7], etc. For self-driving cars, SS plays a crucial role in segmenting pedestrians, vehicles and traffic signs to enhance navigation safety and efficiency. Currently, two types of approaches are commonly used in semantic video segmentation. One is to input consecutive frames into DNNs [27], [28], and the other is to perform DNN-based semantic image segmentation on each frame [12] or keyframes [29], [30]. The DNNs taking consecutive frames as input [27], [28] generally own massive parameters and are computationally intensive. Worse still, the transmission of consecutive frames incurs severe bandwidth consumption. We hence adopt the second approach that only keyframes are transmitted for DNN inference. To further obtain continuous SS for videos, we extract the dense optical flow to propagate both compression strategies and SS results across frames based on temporal consistency, detailed in §V.

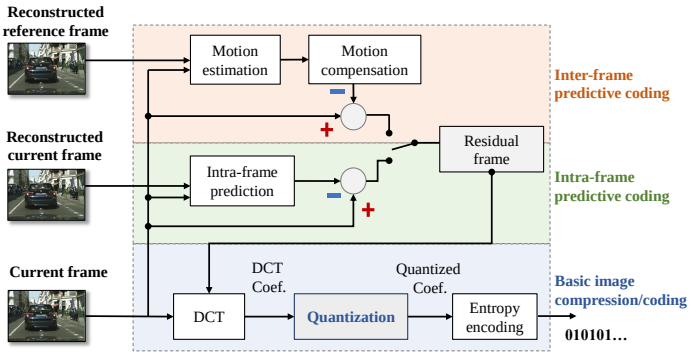


Fig. 3: Typical video codec framework.

B. Typical Video Codec Framework

Numerous compression algorithms [9]–[11], [13], [17]–[19] are developed based on classical video codecs, and STAC is no exception. The common steps of well-known video codecs (e.g., H.26x [31], VPx [32]) are shown in Fig. 3, including the inter-/intra-frame predictive coding and basic image coding (Similar to JPEG [33]). For a single image, the encoder converts the RGB to YUV format, and utilizes block-wise (e.g. 8×4, 4×4, etc.) discrete cosine transform (DCT) to obtain frequency-based DCT coefficients. These coefficients are then quantized and compressed by entropy coding, such as variable length coding and Huffman coding, to create the final bitstream. Based on the basic image coding, the video codecs further incorporate inter-/intra-frame predictive coding to minimize redundancy. To reduce inter-frame redundancy, motion-compensated prediction is performed via motion vectors to predict blocks in current frames from reference frames. To reduce intra-frame redundancy, the pixels located to the left or above are adopted as predictions for the current block. Finally, the residual frame calculated by subtracting the prediction from the original frame replaces the original frame for basic image coding. Since this subsection discusses spatial adaptive compression, all the designs are based on basic image coding. How to integrate them with predictive coding will be explained in §V.

III. System Overview

Fig. 2 illustrates the system architecture of STAC, where a camera-embedded end device compresses video frames, transmits them to a server, and obtains corresponding SS results and compression strategy feedback in real time. The end device then applies the compression strategy to subsequent frames via inter-frame propagation. Specifically, STAC consists of four key modules, which together generate spatio-temporal adaptive compression and transmission strategies. Three of them, including the offline compression strategy generation (§IV-A), online spatial adaptive selection (§IV-B) and online bandwidth-aware configuration adaptation (§VI), run on the

server, while only the online temporal adaption (§V) runs on the end device.

(i) Offline compression strategy generation: This module is executed offline on the server prior to the online running stage. Specifically, it takes input images/frames collected offline to construct multi-level frequency-sensitive quantization tables for the subsequent online spatial adaptive selection. These tables are generated based on average DNN's gradients w.r.t. frequency bins of DCT coefficients [12], since STAC optimizes quantization in the frequency domain.

(ii) Online spatial adaptive selection: This module also runs on the server, but takes the role of generating spatial adaptive compression strategies for the transmitted frames during the online stage. Specifically, based on DNN's gradients w.r.t. pixel plane, different quantization tables are assigned to different regions of the video frame, e.g., low compression-ratio quantization tables are used for high-sensitive regions. The levels of quantization tables are then fed back to the end device, which eases the transmission burden brought by compression strategy synchronization.

(iii) Online temporal adaptation: This module is implemented online on the end device, and is proposed to adapt the above spatial adaptive compression strategy to the changing video content. This is achieved through the propagation of both compression strategies and SS results across frames using dense optical flows. Additionally, the compression strategy is further integrated with inter-/intra-frame predictive coding techniques through compatibility modification.

(iv) Online bandwidth-aware configuration adaptation: This module runs online on the server and is designed to adapt the above spatio-temporal adaptive compression to time-varying bandwidth, aiming at maximizing SS accuracy while maintaining real-time performance. Specifically, an NN-based configuration adaptive scheme is utilized to map the estimated bandwidth and video content to compression configuration adjustment.

A list of key notations and abbreviations is presented in Table I.

TABLE I: KEY NOTATIONS AND ABBREVIATIONS

Notation	**Description**
Q, B	Loss function, upperbound of the allowed loss increment
M, N	Number of pixels/DCT coefficients in a frame or a block
x_i, s_i	i_{th} pixel, i_{th} DCT coefficient, $i \in \{1, 2, ..., M\}$
g_{x_i}, g_{s_i}	DNN's gradient w.r.t. i_{th} pixel or i_{th} DCT coefficient
$s_i \rightarrow n$	Frequency n that s_i corresponds to, $n \in \{1, 2, ..., N\}$
g_n, n	Average gradient w.r.t. DCT Coef. of n_{th} frequency
L, r_{max}	Number of quantization tables or regions in a frame
R_r	r_{th} region in a frame, $r \in \{1, 2, ..., r_{max}\}$
q_{s_i}	Quantization step used to s_i
T_l	l_{th}-level quantization table, $l \in \{1, 2, ..., L\}$
q_n^l	quantization step of n_{th} frequency in T_l
l_r	Level of quantization table used in r_{th} region
Q_F, Q_T	True or fake loss function
g_F, g_T	True or fake gradient calculated by Q_F or Q_T
j	Segment j decomposed by a larger compression ratio
V, V_r	Estimated size of a frame or the r_{th} region
V_s, V_u	Estimated frame size w/ or w/o two relaxations, $V_s = V$
fps, t	Offloading frame rate, video segment index
c, d	Single frame complexity, inter-frame difference
Acc, Bit	Average inference accuracy and video size per second
α_t, β_t	Scaling factors of B or fps over last segment $t-1$
$(')$, $('')$	Labeling of original compression strategy for the individual image or new strategy for the residual frame
P	Number of operations per pixel during encoding
Abbreviation	**Description**
SS, OD	Semantic segmentation, object detection
DNN, RL	Deep neural network, reinforcement learning
(i)DCT	(Inverse) discrete cosine transform
mIoU, PA	Mean intersection over union, pixel accuracy
QP, QoE	Quantization parameter, quality of experience
RoI, PID	Region of interest, proportional integral derivative

IV. Leveraging Non-uniform Spatial Sensitivity

In this section, we propose a spatial adaptive compression strategy, which consists of offline and online stages.

A. Offline Compression Strategy Generation

We first elaborate on the offline compression strategy generation from the perspective of the following two questions.

1) Why we should generate compression strategy offline: Consider a DNN with loss function Q and an M-pixel input image represented as $\mathbf{x} = \{x_1, x_2, ..., x_M\}$. We follow the approach in GRACE [12] to obtain the DNN's gradient g_{x_i} w.r.t. each pixel x_i by calculating the partial derivative of Q with respect to x_i, i.e., $g_{x_i} = \frac{\delta Q}{\delta x_i}$ through back-propagation from results. Fig. 4 displays the heatmap of g_{x_i} on a random raw image, which confirms the non-uniform spatial sensitivity. For more clarity, we quantize the gradient and observe that the sensitive regions don't align with the boundaries. Depending on these spatial sensitive metrics, the next step is to convert them into spatial adaptive compression strategies. According to the total differential equation, when all Δx_i are very small during compression, the loss change ΔQ of DNN can be modeled as $\Delta Q = \sum_{i=1}^{M} g_{x_i} \Delta x_i$. As quantization is one of the major lossy compression techniques, we convert DNN's gradients w.r.t. pixels $g_{\mathbf{x}}$ to DNN's gradients w.r.t. DCT coefficients $g_{\mathbf{s}}$ for quantization optimization[1]. Therein, $\mathbf{s}$ is a vector of DCT coefficients $\{s_1, s_2, ..., s_M\}$ and the corresponding gradient is g_{s_i}. ΔQ can also be expressed as [12]

$$\Delta Q = \sum_{i=1}^{M} g_{x_i} \Delta x_i = \sum_{i=1}^{M} g_{s_i} \Delta s_i, \tag{1}$$

where $\Delta s_i = s_i - \lfloor \frac{s_i}{q_{s_i}} \rceil q_{s_i}$ denotes the quantization error and $|\Delta s_i| \leq \frac{q_{s_i}}{2}$. Therefore, $\Delta Q_{max} = \sum_{i=1}^{M} |g_{s_i}| \frac{q_{s_i}}{2}$ represents

[1]This conversion is detailed in GRACE [12], which includes: (1) $g_{\mathbf{x}}$ is calculated through back-propagation after $\mathbf{x}$ is fed to the DNN. (2) One more step of back-propagation against the inverse DCT (IDCT) (from $\mathbf{x}$ back to $\mathbf{s}$) is performed to obtain $g_{\mathbf{s}}$, where IDCT is one step of decoding.

the worst-case loss increment caused by quantization steps of the whole image $\{q_{s_1}, q_{s_2}, ..., q_{s_M}\}$.

The end goal of STAC is to guarantee the DNN inference accuracy while minimizing bandwidth consumption. To guarantee the accuracy, an upperbound B of the allowed loss increment is configured through the constraint: $\sum_{i=1}^{M} |g_{s_i}| \frac{q_{s_i}}{2} \leq B$. Then, the goal evolves to find optimal $\{q_{s_1}, q_{s_2}, ..., q_{s_M}\}$ to achieve minimum image size under this constraint, denoted as $\arg\min_{\mathbf{q}} V = \sum_{i=1}^{M} \log_2 \left| \frac{s_i}{q_{s_i}} \right|^2$. By configuring $d_{s_i} = |g_{s_i}| \frac{q_{s_i}}{2}$, this problem is further simplified to [12]

$$\arg\min_{\mathbf{q}} \sum_{i=1}^{M} \log_2 \left| \frac{s_i}{q_{s_i}} \right| = \arg\max_{\mathbf{q}} \prod_{i=1}^{M} d_{s_i} \quad s.t. \sum_{i=1}^{M} d_{s_i} \leq B, \tag{2}$$

The optimal solution is when all $\{d_{s_i}\}_{i=1}^{M}$ are equal and $d_{s_i} = \frac{B}{M}$. In other words, the ideal quantization steps $\{q_{s_1}, q_{s_2}, ..., q_{s_M}\}$ for all DCT coefficients in an image are $q_{s_i} = \frac{2B}{M|g_{s_i}|}$, which minimize the image size while not exceeding B to guarantee the inference accuracy [12]. It is noteworthy that the deviation and solution hinge on two upperbound relaxations, i.e., $\Delta s_i \leq \frac{q_{s_i}}{2}$ and $g_{s_i} < |g_{s_i}|$. The key benefit is to render the computational overhead of finding the optimal solution negligible. Without these relaxations, STAC would have to traverse all quantization steps or quantization tables over all pixels or blocks, with the complexity reaching $\mathcal{O}(L^M)$. Wherein, L is the number of quantization steps or quantization tables. Furthermore, through a combined theoretical and experimental analysis in §IV-B, it is confirmed that STAC under these relaxations can still present a superiority in accuracy-size trade-off.

However, this pixel-level compression strategy is generated online on the resource-rich server/receiver based on the received frame, requiring continuous feedback and transmission to support (de)compression process of subsequent frames. The heavy transmission load makes this compression strategy impractical. For example, supposing a frame has $M = 2048 \times 1024$ pixels, the total number of DCT coefficients reaches $3 \times 2048 \times 1024$. Therefore, up to 6M quantization steps need to be transmitted, which is as large as the raw frame. In addition, the higher the frame rate, the heavier the transmission load of compression strategy synchronization. To ease this load, the most effective approach is to generate a portion of the compression strategy offline, thus minimizing the size of the online compression strategy needed for characterizing the time-varying spatial sensitivity.

2) What this offline compression strategy is: When using offline-generated compression strategies to replace a portion of pixel-level online-generated strategies, performance compromise is inevitable. The problem hence evolves to how to design offline compression strategies that can inherit sensitive features of DNNs in both spatial and frequency domains, approaching pixel-level strategies as closely as possible. The core idea is to generate multi-level block-wise quantization tables offline based on DNN's average gradients w.r.t. spatio-frequency bins of DCT coefficients, which can be collected and computed offline. STAC then converts the online generation of pixel-level quantization steps to quantization table selection on the basis of region. In this way, STAC no longer transmits the pixel-level compression strategy, but simply the levels of quantization tables selected online for different regions.

[2]This equation actually represents the intermediate image size without subsequent entropy coding, but proportional to the final size.

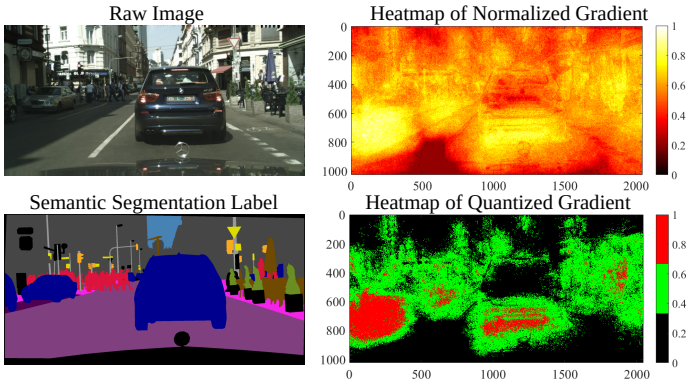


Fig. 4: The heatmaps of DNN's gradients.

The detailed steps for generating multi-level quantization tables offline are as follows: *(i)* First, we calculate DNN's average gradients $\{g_n\}_{n=1}^{N}$ w.r.t. different spatio-frequency bins $\{n\}_{n=1}^{N}$ of DCT coefficients, following GRACE [12]. Therein, N is the number of pixels/DCT coefficients in a block and each DCT coefficient corresponds to n_{th} frequency. Therefore, g_n is the average value of gradients w.r.t. DCT coefficients of n_{th} frequency across all blocks. *(ii)* Then, we configure L levels of upperbounds $\{B_l\}_{l=1}^{L}$ to generate quantization tables $\{T_l\}_{l=1}^{L}$, covering a wide range of fine-grained compression ratios and accuracy levels. Therein, $\{T_l\}_{l=1}^{L}$ are calculated by $\{T_l\}_{l=1}^{L} = \{q_1^l, q_2^l, ..., q_N^l\}_{l=1}^{L}, q_n^l = \frac{2B_l}{M|g_n|}$, with each q_n representing the quantization step on DCT coefficients of n_{th} frequency. Notably, these offline quantization tables are customized to the target DNN but can be fully computed from the captured video frames before formal deployment. Moreover, when the semantic segmentation tasks are the same, these quantization tables further exhibit generalizability across DNNs and datasets, as confirmed in §VIII-D.

B. Online Spatial Adaptive Selection

Based on the offline-generated quantization tables, STAC can select proper T_l for different regions of the received frames on the server/receiver. For a region with low sensitivity (i.e., low gradient), T_l with a high compression ratio is given priority. Conversely, for a region with high sensitivity, T_l with a low compression ratio is preferred. For simplicity of experiments, we divide the frame into r_{max} regions, each of which contains the same number of blocks except the borders. We use the region as the basic unit for T_l selection, which further reduces the transmission load of compression strategies.

As shown in Fig. 5 and Alg. 1, the steps of online quantization table selections for regions $\{R_r\}_{r=1}^{r_{max}}$ are as follows: *(i)* First, STAC measures DNN's gradients $\{g_{s_i}\}_{i=1}^{M}$ w.r.t. all DCT coefficients $\{s_i\}_{i=1}^{M}$ of the real-time received frame. *(ii)* Then,

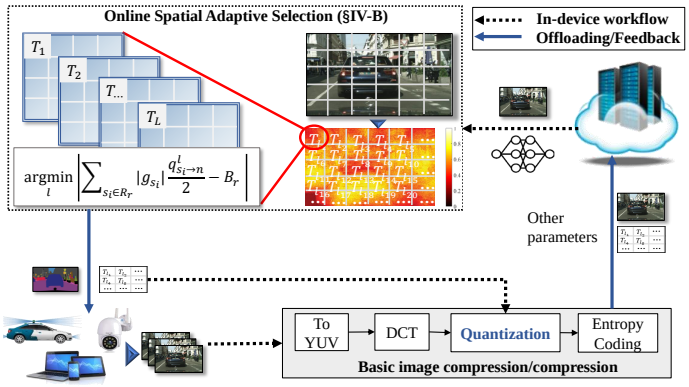


Fig. 5: The framework of online spatial adaptive compression.

STAC calculates the worst-case loss increment for each region R_r under each level of quantization table $T_l = \{q_1^l, q_2^l, ..., q_N^l\}$, which is denoted as $\Delta Q^l_{max,r} = \sum_{s_i \in R_r} |g_{s_i}| \times \frac{q^l_{s_i \to n}}{2}$. Therein, $s_i \to n$ denotes DCT coefficient s_i corresponds to n_{th} frequency. *(iii)* Next, for each region R_r, STAC selects the quantization table T_{l_r} that has the worst-case loss increment $\Delta Q^l_{max,r}$ closest to the upperbound B_r assigned to this region. Therein, $B_r \approx \frac{B}{r_{max}}$ is decided by the number of pixels that each region contains, based on the uniform allocation principle. Moreover, B is adjusted online according to the real-time available bandwidth and video content (detailed in §VI). *(iv)* Finally, only the levels $\{l_r\}_{r=1}^{r_{max}}$ of quantization tables selected for each region are fed back and transmitted online, which significantly reduces bandwidth consumption.

Although appearing promising, it remains uncertain whether this region-based design and the adopted relaxations[3] can still allow STAC to achieve a good performance gain over traditional uniform compression strategy. In the following, we investigate this issue using a combined approach of theoretical derivation and experimental testing, so as to preliminarily verify the feasibility and effectiveness of the proposed compression strategy generation method. Specifically, we suppose that the size and accuracy under the uniform compression strategy without any relaxation are V_u and ΔQ_u, while the size and accuracy under STAC are V_s and ΔQ_s. If we can prove $|\Delta Q_s| < |\Delta Q_u|$ when $V_s = V_u$, STAC is confirmed to present better accuracy-size performance than the uniform compression strategy. Therein, V_u and V_s can be calculated by

$$V_u = \log_2 \left| \frac{\prod_{i=1}^{M} s_i}{(\prod_{n=1}^{N} q_n^l)^{\frac{M}{N}}} \right|, q_n^l \in T_l, \tag{3}$$

$$V_s = \log_2 \left| \frac{\prod_{i=1}^{M} s_i}{\prod_{r=1}^{r_{max}} (\prod_{n=1}^{N} q_n^{l_r})^{\frac{M}{N \times r_{max}}}} \right|, q_n^{l_r} \in T_{l_r}. \tag{4}$$

When supposing $V_u = V_s$, we can obtain the following relationship

$$\prod_{n=1}^{N} q_n^l = \left[\prod_{r=1}^{r_{max}} \left(\prod_{n=1}^{N} q_n^{l_r} \right)^{\frac{M}{N \times r_{max}}} \right]^{\frac{N}{M}}. \tag{5}$$

From this equation, we can employ the $\{T_{l_r}\}_{r=1}^{r_{max}}$ generated by STAC to determine the appropriate T_l for uniform compression to ensure the same size. After obtaining both $\{T_{l_r}\}_{r=1}^{r_{max}}$ and T_l, $|\Delta Q_u| - |\Delta Q_s|$ can be calculated by

$$\begin{aligned} |\Delta Q_u| - |\Delta Q_s| = & \left| \sum_{i=1}^{M} g_{s_i} \left(s_i - \lfloor \frac{s_i}{q^l_{s_i \to n}} \rceil q^l_{s_i \to n} \right) \right| \\ & - \left| \sum_{r=1}^{r_{max}} \sum_{s_i \in R_r} g_{s_i} \left(s_i - \lfloor \frac{s_i}{q^{l_r}_{s_i \to n}} \rceil q^{l_r}_{s_i \to n} \right) \right|. \end{aligned} \tag{6}$$

Based on Eq. (3)-(6), we test $|\Delta Q_u| - |\Delta Q_s|$ with respect to $V_u(V_s)$ in the cityscapes dataset [34] by continuously adjusting the upperbound B that determines $\{T_{l_r}\}_{r=1}^{r_{max}}$ in STAC. The results are shown in Fig. 6. We can notice that the value of $|\Delta Q_u| - |\Delta Q_s|$ remains 100% greater than 0 regardless of $V_u(V_s)$ and keeps growing as the $V_u(V_s)$ decreases. This indicates that STAC outperforms the uniform compression strategy all the time, and the performance improvement becomes more pronounced when the size is relatively small. The experiment preliminarily validates the effectiveness of the region-based design and relaxations employed to generate the compression strategy, before formal experiments in §VIII.

However, there still remain several concerns regarding the compression strategy, including *(i)* whether it is necessary to transmit raw frames for calculation of DNN's gradients, as the loss function Q requires ground truth; *(ii)* whether the premise that supports the validity of the equation $\Delta Q = \sum_{i=1}^{M} g_{s_i} \Delta s_i$,

Algorithm 1 Online Spatial adaptive compression strategy.

Input: $\{q_1^l, q_2^l, ..., q_N^l\}_{l=1}^{L}$ – L quantization table levels
B – Upperbound of the allowed DNN loss increment
Q – DNN loss function
$\{s_i\}_{i=1}^{M}$ – DCT coefficients of the received frame
$\{R_r\}_{r=1}^{r_{max}}$ – Regions of the received frame
Output: $\{l_r\}_{r=1}^{r_{max}}$ – Levels of quantization tables selected by regions

for $i \leftarrow 1$ to M **do**
  $g_{s_i} = \frac{\delta Q}{\delta s_i}$
end for
for $r \leftarrow 1$ to r_{max} **do**
  $min = +\infty$
  $B_r \approx \frac{B}{r_{max}}$
  for $l \leftarrow 1$ to L **do**
    $\Delta Q^l_{max,r} \leftarrow \sum_{s_i \in R_r} |g_{s_i}| \times \frac{q^l_{s_i \to n}}{2}$
    if $\left|Q^l_{max,r} - B_r\right| < min$ **then**
      $l_r = l$
      $min = \left|Q^l_{max,r} - B_r\right|$
    end if
  end for
end for

[3]The relaxations include $\Delta s_i \le \frac{q_{s_i}}{2}$ and $g_{s_i} < |g_{s_i}|$, which enable STAC to generate spatial adaptive compression strategy with minimum complexity, as detailed in §IV-A.

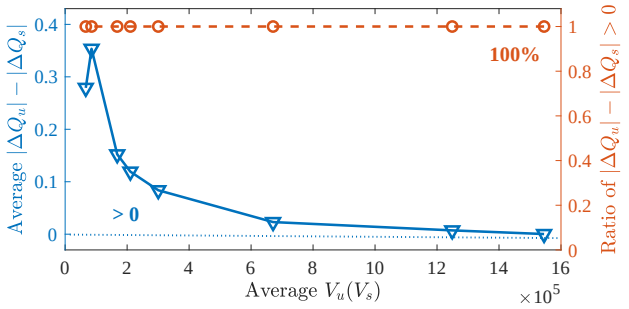


Fig. 6: $|\Delta Q_u| > |\Delta Q_s|$ holds 100% regardless of $V_u(V_s)$.

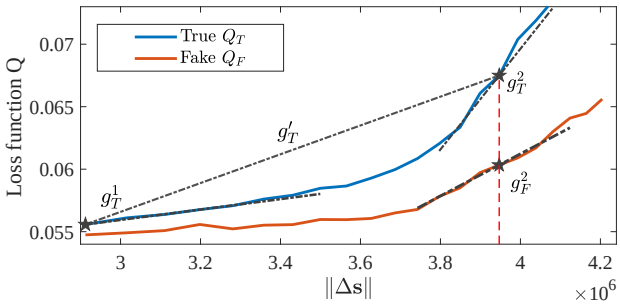


Fig. 7: Fake Q_F vs. true Q_T when compression ratio increases.

i.e., all Δs_i are very small, can be extended to a larger compression range; *(iii)* whether the transmitted frame that is non-uniformly compressed by STAC will make the spatial sensitivity metrics less objective and affect the subsequent new non-uniform compression strategies.

For the first concern, we argue that the fake gradient computed by the compressed frame is even more appropriate than the true gradient computed by the raw frame. The fake gradient is computed as follows: *(i)* We treat the DNN inference result of the compressed frame as the fake ground truth/label. *(ii)* We calculate the fake loss function Q_F between the probability vector of DNN output and the fake label. *(iii)* The fake gradient is calculated. We have tested both true Q_T and fake Q_F. The results are shown in Fig. 7, with x-axis $\|\Delta \mathbf{s}\|$ representing the total quantization errors as compression ratio increases, i.e., $\|\Delta \mathbf{s}\| = |\Delta s_1| + ... + |\Delta s_M|$. Both true and fake Q are concave, but fake Q_F increases slowly. The reason is that the fake label is closer to the DNN output vector than the true label. For example, assuming that DNN output is $(0.6, 0.4, 0)$ and the true label is $(0, 1, 0)$, the fake label however can be computed as $(1, 0, 0)$, which results in a lower loss value. g_T^1 and g_T^2 in Fig. 7 respectively denote the true gradients of the raw frame (nearly) and the compressed frame. g_T' represents the true speed of loss increment under compression. Both g_T^1 and g_T^1 are very different from g_T^3, while fake g_F^2 that is computed by fake Q_F and compressed frames is more similar to g_T', which is more appropriate to calculate the quantization strategy. This phenomenon exists when $\|\Delta \mathbf{s}\|$ represents blocks, regions or frames, but with different concave shapes.

For the second concern, we argue that the premise can be extended to larger compression ranges. As described, the fake g_F^2 of the same compressed frame is exactly more in line with the true speed g_T' of loss increment. When we decompose the total loss increment ΔQ_T under a larger compression ratio to the sum of losses in small segments j, i.e., $\Delta Q_T = \sum_j \Delta Q_T^j = \sum_j \sum_{i=1}^{M} g_{T,s_i}^j \Delta s_i^j$, it can be converted to $\Delta Q_T = \sum_{i=1}^{M} \sum_j g_{T,s_i}^j \Delta s_i^j = \sum_{i=1}^{M} g_{T,s_i}' \sum_j \Delta s_i^j \approx \sum_{i=1}^{M} g_{F,s_i} \Delta s_i$, which exhibits the same format as the original equation. Therein, each Δs_i^j denotes a small fraction of pixel error along quantization, and the total quantization error/range $\Delta s_i = \sum_j \Delta s_i^j$ is much larger than the small Δs_i^j. The s_i can also be used as the block or region.

For the third concern, we claim that the transmitted frames that are non-uniformly compressed based on STAC will not affect the subsequent new non-uniform compression strategies. The reason is that the loss increment upperbound is uniformly assigned. When assuming $\frac{B}{r_{max}} \approx \Delta Q_{T,max,r}$ and $\Delta s_i \approx \frac{q_{s_i}}{2}$, the following mappings hold and keep repeating, i.e., $\{q_{s_i}\}|_r \rightarrow \{s_i - \frac{q_{s_i}}{2}\}|_r \rightarrow \{g_{F,s_i}^{\frac{q_{s_i}}{2}}\}|_r \xrightarrow{B_r} \{q_{s_i}\}|_r \rightarrow \{q_{s_i}\}|_{r'}$ (next frame). That is to say, the regions mapped between the front and back frames generate the same quantization strategy. Furthermore, as the first frame is uniformly compressed and transmitted (as described in §V-B), the compression strategy will gradually converge to the non-uniform format.

V. Leveraging Temporal Consistency

The compression strategy above is customized for individual frames/images, yet videos transmitted in a frame-independent mode introduce huge redundancy. Moreover, the generation via back-propagation from results makes the compression strategy useless for both already-transmitted and subsequent frames whose spatial sensitivity changes with video content. We hence propose a temporal adaptive scheme, which propagates both SS results and compression strategies across frames, and integrates the spatial adaptation with inter-/intra-frame predictive coding in H.26x through compatibility modification.

The framework of the entire temporal adaptive scheme is depicted in Fig. 8. Specifically, both the compression strategy and SS results for the new frame are not directly obtained from server-side feedback but via local in-device inter-frame propagation from previously cached contents. This compression strategy, oriented to the individual frame (§IV-B), is then converted into a new one compatible with the inter-/intra-frame residual to minimize redundancy. After compression and encoding, this residual is offloaded to the server, decoded, and fed into the DNN to generate new SS results and compression strategies. These are then fed back and cached on the end device for future propagation. The full cycle repeats continuously, ensuring efficient compression and timely updates of semantic segmentation results.

A. A Primer on Dense Optical Flow

We first briefly introduce the dense optical flow for propagation. Optical flow is defined as the motions of pixels between adjacent frames. Unlike the sparse optical flow that only focuses on interesting features, dense optical flows provide flow vectors of the entire frame, up to one flow vector per pixel. Recent works [29], [30] have extracted dense optical flows [35] to propagate SS results across frames. Similarly, STAC adopts

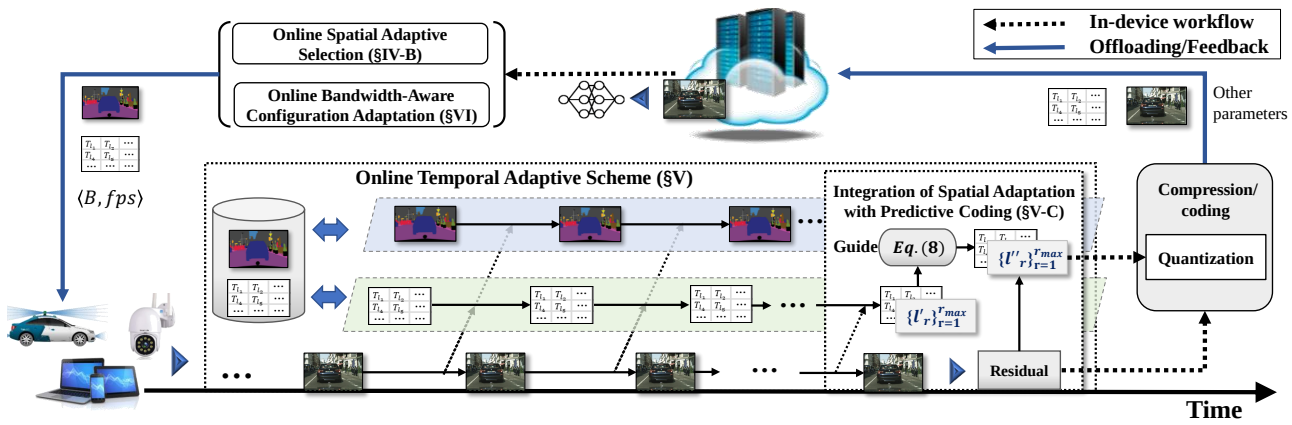


Fig. 8: The framework of online temporal adaptive scheme.

DIS [35], which is state-of-the-art in terms of computational efficiency on CPU to propagate both compression strategy and SS results. Compared to deep optical flow methods, DIS achieves competitive accuracy and higher frame rates of 10-600 Hz on 1024×436 images on a single CPU core. This makes it fully compatible with mobile vision systems such as mid-end smartphones, tablets, AR headsets, etc. Besides, DIS operates in a coarse-to-fine fashion, giving STAC more flexibility to adjust computational costs according to available runtime resources of mobile vision systems.

In addition to the above dense optical flow extraction method, H.26x [31] itself also estimates motion vectors to generate inter-frame residuals. However, it is noteworthy that these motion vectors extracted in H.26x cannot be used in STAC for propagation, as they are highly misleading, more inaccurate and don't represent "true motion vectors". The fundamental reason lies in the different purposes. The calculation of motion vectors in H.26x doesn't target the best motion estimation, but rather finds the best-matching block with minimal color difference to achieve the highest compression ratios. Hence, the method employed is coarse-grained and has low complexity. For each block, H.26x simply searches a limited area in the reference frame (typically centered at the current block's position) to locate the best match, which may not align with the actual motion vector but has little impact. In contrast, the computation of dense optical flows in STAC aims to achieve as accurate motion estimation as possible, incorporating techniques such as iterative estimation, forward-backward consistency, etc. Since STAC requires accurate tracking and propagation of semantic information, it is necessary to estimate the motions of pixels as accurately as possible.

B. Inter-Frame Propagation

The specific inter-frame propagation workflow is depicted in Fig. 8. In our system, both SS results and spatial compression strategies are generated on the resource-rich server/receiver based on received frames (called keyframes) and fed back to the end device/sender. STAC then continuously caches and updates the latest SS results and spatial compression strategies fed back from the server on the end device for future separate propagation as needed.

We first introduce the workflow of propagating spatial compression strategies. Specifically, the cached compression strategy on the end device is only propagated to frames decided as new keyframes for transmission, where the keyframe decision depends on the fps adjustment in §VI. Through dense optical flows, this propagation is performed on the basis of the region R, with regions not covered by optical flows directly applying the nearest quantization table on the frame plane. Additionally, the first frame without a corresponding spatial adaptive compression strategy is uniformly compressed using a middle-level quantization table. It is noteworthy that the dense optical flow is computed only between adjacent raw frames to ensure accuracy, based on which the compression strategies are propagated continuously across adjacent frames.

After detailing the propagation of compression strategies, we further evaluate how much deviation it might introduce into the compression strategy, which potentially affects performance. To this end, we conduct experiments by comparing the compression strategy obtained after propagating a certain distance with the newly generated one. Fig. 9 demonstrates the experimental results in CDF format. We can observe that the compression strategy deviation increases with the propagation distance, quantified by the difference in the selected quantization table level for each pixel block. It is noteworthy that even when the propagation distance reaches 8 frames (that corresponds to a keyframe transmission rate of 2 fps), there are still approximately 90% of the pixel blocks that have no compression strategy deviation. Most of the deviations are even no more than 3 levels. The inherent reason is that the fine-grained pixel-level dense optical flows [29] are sufficient to support coarse-grained region-level compression strategy propagation. Furthermore, the minimal impact of inter-frame propagation on the compression strategy deviation is further confirmed by the great performance of STAC in §VIII.

We next introduce the workflow of propagating SS results. This workflow is similar to propagating compression strategies, with only minor differences. First, the cached SS results are propagated to each frame at the pixel level to ensure real-time performance, avoiding the need to wait for server-side feedback. Additionally, depending on different types of applications, the workflows of propagating SS results vary accordingly: *(i)* The first type refers to applications where video

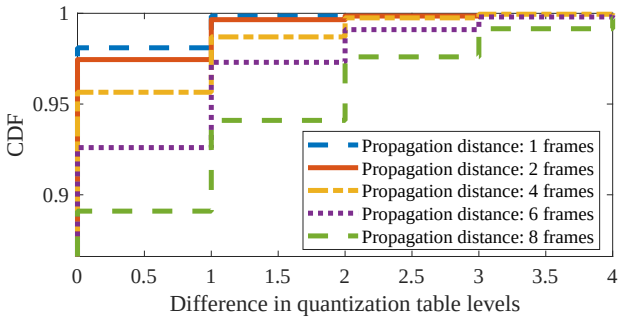


Fig. 9: The compression strategy deviation caused by inter-frame propagation.

receivers are consumers, such as real-time surveillance/drone video analytic [2], [4] and semantic communication [16], etc. In this case, the segmentation results obtained through inter-frame propagation will be further transmitted from the video sender to the receiver, replacing the original frames. This design exhibits effectiveness in the accuracy-bitrate trade-off, as the size of semantic segmentation frames is typically only 1-2% of the original frame size due to the sparse content features, when both are losslessly compressed. *(ii)* Conversely, for applications where the video sender itself requires segmentation results, such as try-on/make-up applications on cell phones [6], [7], there is no need to transmit propagated segmentation results to reduce the bandwidth consumption. To validate this design, the experimental scenarios in §VIII are set to the first type that encounters more bandwidth pressure.

C. Integration of Spatial Adaptation with Predictive Coding

Building upon the spatial adaptive compression strategy for individual images/frames, STAC further integrates it with inter-/intra-frame predictive coding to mitigate huge inter-/intra-frame redundancy. As detailed in §II-B, the residual frame and motion vector replace the original frame to be quantized and transmitted. Therein, only a small number of bits are required to signal motion vectors due to the effective motion vector prediction. The residual frame, however, contains a significant amount of energy in frame areas with high detail [31]. Yet, how to perform spatial adaptive compression for residual frames like §IV poses a new challenge.

Two heuristic approaches may be considered: *(i)* directly applying the original compression strategy to the residual frame, or *(ii)* recomputing DNN's gradients w.r.t. the residual frame and thus generating new compression strategy. However, the problem with the first approach is that the residual frame captures more inter-/intra-frame difference, while the original compression strategy is subject to the single-frame content. The spatial sensitivity (i.e., gradients) of these two may be quite different. As for the second approach, the residual-customized compression strategy makes no sense. It can neither be applied to the already-transmitted residual frame due to the back-propagation of results, nor propagated to subsequent residual frames due to lack of theoretical proof of temporal consistency between residual frames.

We then turn to an indirect approach that utilizes the original compression strategy as a reference to guide the quantization of residual frames. For distinguishing purposes, we use the notation $(')$ to mark parameters related to the original compression strategy and $('')$ to the new one for the residual frame. Recall that for any keyframe to be transmitted, we first obtain the spatial adaptive compression strategy $\{l'_r\}_{r=1}^{r_{max}}$ for the original frame through propagation. Then, the keyframe is compressed and reconstructed individually based on the basic image codec, which is not transmitted but provides size and accuracy constraints to generate $\{l''_r\}_{r=1}^{r_{max}}$ for the inter-/intra frame residuals. The principle of generating spatial adaptive compression strategy $\{l''_r\}_{r=1}^{r_{max}}$ is summarized as follows

$$\min_{l''_r} \quad V''_r = \sum_{s''_i \in R_r} \log_2 \left| \frac{s''_i}{q^{l''_r}_{s''_i \to n}} \right|, \tag{7}$$

$$s.t. \quad \sum_{x''_i \in R_r} \Delta x''_i \leq \sum_{x' \in R_r} \Delta x'_i, \tag{7a}$$

$$V''_r \leq V'_r. \tag{7b}$$

Note that the generation of $\{l''_r\}_{r=1}^{r_{max}}$ is still on a per-region basis. We take region $R_r, r \in \{1, 2, ..., r_{max}\}$ as an example. The constraint Eq. (7a) guarantees its accuracy by requiring the compression level l''_r applied to the residual frame to cause less pixel error than the compression level l'_r applied to the original frame. Besides, the Eq. (7b) constrains its size by requiring the size of resulting residual frame V''_r to be smaller than the size of the original frame region V'_r[4], where the size of motion vectors is negligible. If both constraints are satisfied, the compression level l''_r that achieves the minimum size is selected for the R_r of the residual frame, otherwise, it falls back to the original frame with only l'_r-level basic image coding. In general, $\{l'_r\}_{r=1}^{r_{max}}$ is directly calculated by DNN's gradients w.r.t. original frame plane, while $\{l''_r\}_{r=1}^{r_{max}}$ is generated by using $\{l'_r\}_{r=1}^{r_{max}}$ as a reference, which strategically integrates spatial sensitive features with predictive coding.

During the $\{l''_r\}_{r=1}^{r_{max}}$ generation process, there are two additional issues that deserve attention: *(i)* The first is to guarantee the generation of $\{l'_r\}_{r=1}^{r_{max}}$. Since the residual frame is transmitted after inter-frame compression, the DNN on the server/receiver cannot obtain gradients $g'_{s'}$ w.r.t. DCT coefficients s' as before. To tackle this problem, we make the decoded full frames undergo DCT and IDCT once more on the server/receiver. As such, the subsequent NN inference can still utilize back-propagation to obtain an approximate value of $g'_{s'}$. *(ii)* The second is to minimize the computational complexity. As Eq. (7a) constrains the pixel error, each block requires continuous repetition of (de)quantization-IDCT process to traverse and select appropriate compression levels l''_r. To reduce the complexity, we convert this constraint to the frequency domain for IDCT avoidance. Therein, the threshold for frequency-domain error cannot be directly calculated by $\sum_{s' \in R_r} \Delta s'_i$ during l'_r-level basic image coding on the original frame. Instead, the threshold for the residual frame is necessary for alignment. Hence, we use the reconstructed frame (inverse to

[4]Similar to V''_{R_r}, $V'_{R_r} = \sum_{s'_i \in R_r} \log_2 \left| \frac{s'_i}{q^{l'_r}_{s'_i \to n}} \right|$, representing the size of region R_r after applying l'_r to the original frame.

Data	Compression configurations		Content-related metrics		Performance	
	B	fps	c	d	$Bit\ (KB)$	$Acc\ (\%)$
1	$1e^{-4}$	10	$2.33e^6$	$8.77e^7$	1622.5	88.17
2	$1e^{-5}$	15	$2.23e^6$	$7.85e^7$	9617.4	95.94
...	...	...	...	...	...	...

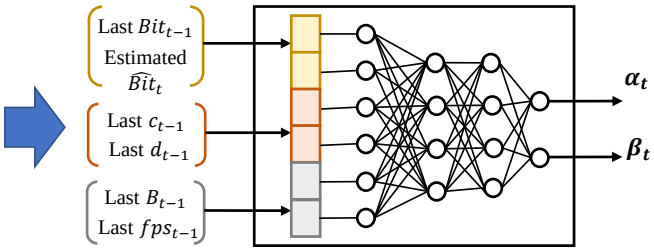


Fig. 10: The training and architecture of bandwidth-aware configuration adaptive neural network.

l'_r-level basic image coding) to regenerate the residual, perform DCT, and compute the coefficient error as the frequency-domain threshold. Furthermore, we take the compression level l''_r of each region R_r in the last keyframe as the starting point for up/down traversal to reduce the traversal range.

VI. Bandwidth-Aware Configuration Adaptation

The aim of this section is to make our proposed compression strategy automatically adapt to varying bandwidth and video content. Given that our compression strategy is tailored for SS, the conventional human-oriented configuration adaptive scheme [20] in H.26x for frames and blocks is no longer suitable. Existing task-oriented configuration adaptive schemes are either tailored for OD tasks [9]–[11], unable to adapt to varying video complexity [18], [19], or lack fine-grained block-wise allocation [13], [21]–[24]. Nevertheless, adjusting video configurations is essential to align the resulting video bitrates with the available bandwidth, ensuring real-time performance while enhancing SS accuracy.

To address this problem, we propose an NN-based bandwidth-aware configuration adaptive scheme that is compatible with our compression strategy. It maps the real-time bandwidth and video content to appropriate video configurations. For the first time, multi-level block-wise configuration adaptive streaming is achieved for semantic segmentation, which not only minimizes redundancy but also matches the bandwidth. Specifically, STAC converges the large-dimensional quantization table selections for all pixel blocks (i.e., $\{l''_r\}_{r=1}^{r_{max}}$) across the entire frame plane to the one-dimensional upperbound B of DNN loss increments. This minimizes the action space of the NN for configuration adaptation, ensuring convergence without affecting block-wise quantization table selections.

During the offline stage, a supervised training set is first produced by compressing videos under different configurations (B and fps). Therein, fps is the frequency of keyframes to be transmitted. We run SS models on these videos to record the accuracy Acc and video size Bit per second. As the relationship between accuracy/size and compression configurations varies with video content [9], [21], we additionally record two content-related metrics: the single-frame complexity c and inter-frame difference d. Therein, c is represented by the sum of un-quantized coefficients $\sum_{i=1}^{M} \log_2 |s'_i|$ without increasing much overhead, and d is computed by the sum of absolute pixel differences between adjacent frames like [36]. Both c and d are averaged per second. With these two metrics, STAC can easily detect whether the current video has drastic or stable inter-frame changes, and whether the scenes are complex or simple. These metrics play a crucial role in determining the appropriate configurations to minimize redundancy, such as whether to filter fewer or more frames, i.e., adjusting fps. Then, after traversing the training data set, we construct a multi-dimensional lookup table $\langle B, fps, Acc, Bit, c, d\rangle$ in terms of video segments per second.

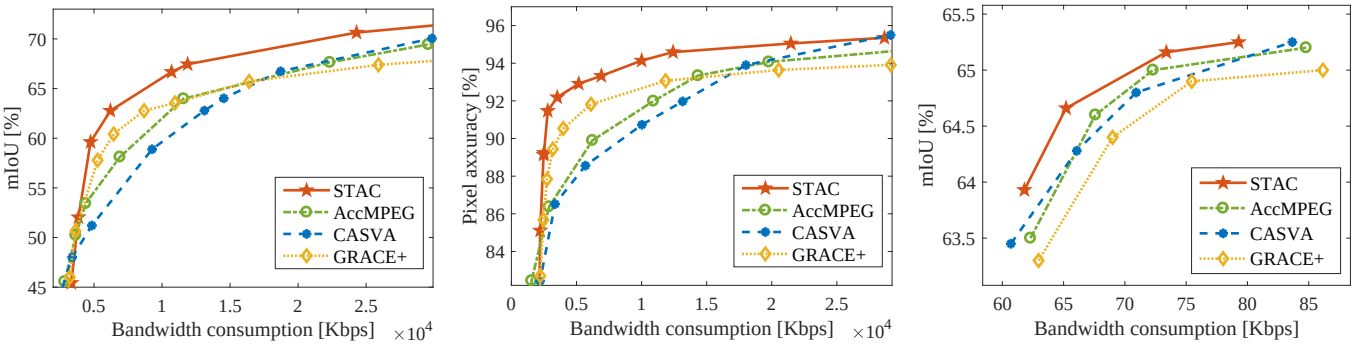
To resolve conflicts in the table brought by different DNNs, scenarios, etc., we use it to train an NN f as shown in Fig. 10, which extracts relatively unified relationships among elements to enhance generalization and save memory resources. Many studies [9], [37] prefer NNs like $f : \langle B, fps, c, d\rangle \rightarrow \langle Acc, Bit\rangle$. However, such networks make it necessary to traverse different configurations for finding the best at the online stage. Therefore, we turn to construct an NN that has opposite mappings, $f : \langle Bit_{t-1}, \widehat{Bit}_t, B_{t-1}, fps_{t-1}, c_{t-1}, d_{t-1}\rangle \rightarrow \langle \alpha_t, \beta_t\rangle$, which takes as input:

- *(i)* the video bitrate Bit_{t-1} of last video segment $t-1$, and the estimated available bandwidth $\widehat{Bit}_t$ for next video segment t.
- *(ii)* the last upperbound B_{t-1} of DNN loss increment and the last fps_{t-1} as the reference for next adjustment.
- *(iii)* the two content-related metrics c_{t-1}, d_{t-1} averaged over frames in the last video segment, used as a prediction of c_t, d_t based on temporal correlation.

This neural network f outputs the scaling factors $\alpha_t = \frac{B_t}{B_{t-1}}$ and $\beta_t = \frac{fps_t}{fps_{t-1}}$ instead of B_t and fps_t for the next video segment t to further improve generalization. The training is executed with any combination of $\langle ..., c_{t-1}, d_{t-1}\rangle$ and $\langle ..., c_t, d_t\rangle$ in training set. Moreover, we filter out each $\langle B', fps', Acc', Bit', c', d'\rangle$ in the training set, as long as there exists $\langle ..., c', d'\rangle$ with larger accuracy and smaller size. The experiments in §VIII-D show that f generalizes quite well across DNNs and scenarios. Of course, if one pursues to perfectly handle the domain shift between training and testing, continuous customization of f can be performed through transfer learning, lifelong learning, etc.

VII. Implementation

We prototype STAC on H.264 codec and modify only the quantization module, following the quantization table/level of each block selected by STAC. Specifically, we mimic the open source codes for H.264 (implemented in C++ [38]), but use Python and call C++ libraries through interfaces to re-implement the main H.264 architecture. The root cause is that STAC requires the back-propagation from DNN inference to



(a) DRN-D-22 on Cityscapes dataset. (b) DRN-D-38 on Cityscapes dataset. (c) BiSeNet on CamVid dataset.

Fig. 11: Evaluation of overall accuracy-bitrate performance across different DNN models and datasets.

the DCT coefficient to calculate gradients. Therefore, when using Pytorch for DNN inference, the H.264 modules involved in back-propagation are aligned via Python, giving convenience to experiments. Regarding H.264 architecture, we faithfully implement most modules including predictive coding, DCT, quantization and entropy coding, while simplifying partial functions without interfering with the evaluation of STAC. Therein, the block size is fixed to 4×4 pixels without multiple choices. All baseline algorithms built on H.264 are also implemented by our H.264 architecture for experimental consistency.

We implement STAC on a portable and small form factor Intel NUC Kit NUC7i5DNHE, which includes an Intel Core i5-7300U CPU (with 2 cores, 3 MB cache, 2.6 GHz). The specification of this hardware is comparable to what is available in today's mid-end smartphones such as OnePlus 9 and Samsung Galaxy A52 5G. We deploy an edge server equipped with an Intel Core i7-5820K CPU and an Nvidia Tesla T4 GPU. It is noteworthy that most STAC modules run on the resource-rich server, while only the inter-frame propagation and integration with predictive coding are additional overhead on the end device. Among them, the dominant real-time pressure arises from computing the dense optical flow for the propagation of SS results. Fortunately, DIS operates in a coarse-to-fine fashion to enable the propagation to reach 17 fps (i.e., the frame rate of Cityscapes dataset [34]). The computational overhead of each module is further detailed in Table II in §VIII-F.

We adopt a data-driven methodology that uses real-world network bandwidth to simulate a real-time video transmission system. Specifically, after each keyframe is encoded, the delay time for the keyframe to reach the server/receiver is simulated as the queuing time plus the transmission time, which is computed via dividing the compressed frame size by the instantaneous network bandwidth. Following §VI, the video compression configuration is dynamically adjusted per second depending on the real-time bandwidth and video content. In addition, once a new video compression configuration is decided and executed, previous frames that did not complete transmission are discarded to avoid delay accumulation.

VIII. Evaluation

A. Experimental Setup

Datasets, DNN models, and Metrics. We evaluate STAC on SS tasks using three different datasets, including well-known Cityscapes dataset [34] with 2048×1040 resolution, YouTube videos [39], [40] with 2048×1040 resolution and CamVid dataset [41] with 720×960 resolution. Among them, YouTube videos [39], [40] are captured by drives in Las Vegas and Seoul, and then downsampled to 2048×1040 resolution as a complement for the Cityscapes dataset, where the lengths of continuous videos are too short. Since the continuous videos in the Cityscapes dataset and YouTube videos are not artificially annotated, we regard the DNN outputs of raw videos as ground truth, following practices in previous studies [17], [21]. While introducing some bias in the accuracy assessment, it helps mitigate influences caused by the target DNN itself and focuses more on the impacts of compression strategies.

We employ three DNN models including DRN-D-22 [42], DRN-D-38 [43] and BiSeNet [44] for semantic segmentation to evaluate whether STAC can adapt to different DNNs. Besides, we test STAC on three different network datasets including LTE/4G [25], FCC [26], and WiFi to evaluate its adaptability to varying bandwidth. Each of them exhibits differentiated bandwidth fluctuation characteristics with mean and standard deviation of $\{2.14 \pm 1.02, 0.73 \pm 0.71, 0.51 \pm 0.27\} \times 10^4$ (Kbps), respectively. Moreover, we evaluate three key metrics, including the video transmission bitrate (Kbps) (i.e., bandwidth consumption), SS accuracy (%) and delay (s). The SS accuracy is tested by the pixel accuracy (i.e., $\mathrm{PA} = \frac{\sum_{i=0}^{k} p_{ii}}{\sum_{i=0}^{k} \sum_{j=0}^{k} p_{ij}}$) and the mean intersection over union (i.e., $\mathrm{mIoU} = \frac{1}{k+1} \sum_{i=0}^{k} \frac{p_{ii}}{\sum_{j=0}^{k} p_{ij} + \sum_{j=0}^{k} p_{ji} - p_{ii}}$). Therein, $k+1$ is the number of semantic classes including a class for do-not-care regions, p_{ij} denotes the number of pixels that predict the true class i to class j. More specifically, IoU represents the ratio of the intersection and union of the predicted and true classes of pixels. mIoU is the average IoU value of all classes. Thus, mIoU or $\mathrm{PA} = 100\%$ indicates that the semantic predictions of all pixels exactly match the true labels, while mIoU or $\mathrm{PA} = 0\%$ indicates that the semantic predictions of all pixels are wrong.

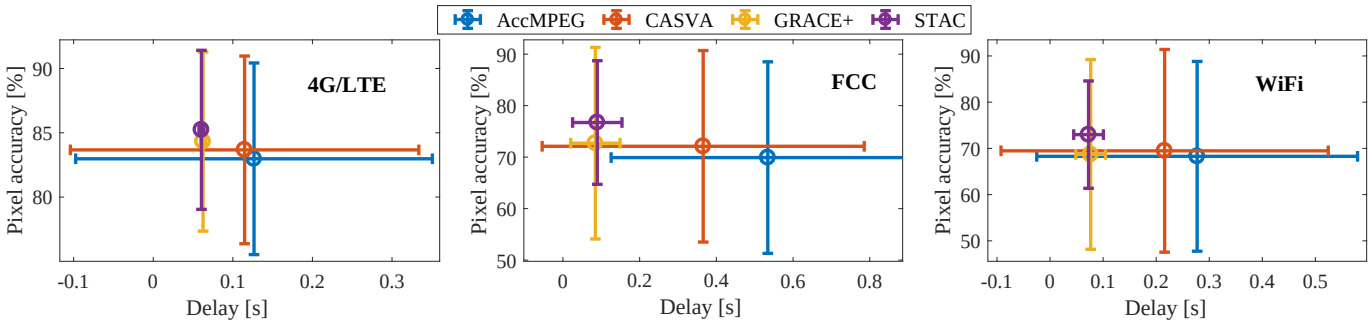


Fig. 12: Evaluation of STAC's adaptability to varying bandwidth under different network types.

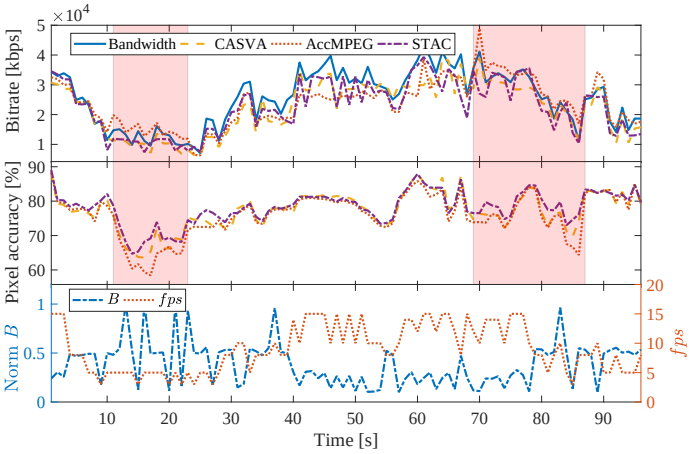


Fig. 13: A showcase of video configuration/bitrate adaptation.

Baseline. The baseline compressive streaming algorithms are summarized as follows.

- *AccMPEG* [18] is a state-of-the-art non-uniform compression algorithm applicable to SS. It also leverages DNN's gradients w.r.t. pixels to guide compression, but is coarse-grained and limited to two-level compression that assigns high/low quality based on a pre-defined gradient threshold. Furthermore, the high/low-quality assignments are real-time inferred by a separate DNN on the end device, which can only be executed once every 10 frames due to the computational burden.
- *CASVA* [21] is a compressive streaming algorithm built on top of the H.264 codec [31]. It is not innovative in compression strategy by applying a uniform quantization parameter (QP) to the whole frame plane. Yet, it uses reinforcement learning (RL) to automatically adjust configurations including the resolution, fps and uniform QP to maximize DNN inference accuracy under dynamic bandwidth. Overall, a comparison with CASVA equates to a comparison with an H.264 encoder enhanced by an RL-based configuration adaptive algorithm.
- *GRACE* [12] is a uniform compression algorithm oriented to SS. Prototyped on the JPEG codec, GRACE replaces the quantization tables with self-generated ones based on the frequency sensitivity of DNNs, and uniformly applies them to the frame plane. As GRACE transmits each frame as individual images, which introduces excessive redundancy, we enhance it to GRACE+ by incorporating inter-frame propagation of SS like STAC and transmitting only keyframes. Overall, a comparison with GRACE+ equates to a comparison with a JPEG encoder enhanced by inter-frame propagation and frequency-sensitive quantization tables.

B. Better Accuracy-Bitrate Balance

We first test the accuracy-bitrate balance by adjusting related compression configurations (B and fps in STAC). Three DNN models, two SS datasets, and two accuracy metrics are used to mitigate experimental randomness.

STAC vs. AccMPEG. We implement AccMPEG by applying non-uniform compression with two-level QPs as described in §VIII-A. To test its accuracy-bitrate balance, we systematically adjust frame rates and centers of QPs. Fig. 11 shows the overall performance boost by STAC across various models and datasets: *(i)* STAC achieves an average improvement of 0.2-2.7%, and in some cases up to 0.7-6.2%, in accuracy metrics of mIoU or PA when the bandwidth consumption is equal. *(ii)* When the accuracy is consistent, the bandwidth consumption is significantly reduced by up to 10.1-63.7%, with an average of 4.1-32.8%. In other words, STAC enables a real-time SS capability for up to 2.7× number of devices, without any upgrades to communication infrastructures. These improvements over AccMPEG are attributed to STAC's more fine-grained spatial adaptive compression and inter-frame propagation mechanism, which reduce bandwidth reliance in both temporal and spatial domains.

STAC vs. CASVA. According to CASVA, we comprehensively adjust the QP, resolution, and frame rate based on the H.264 codec, and test its optimal accuracy-bitrate balance curve. Shown in Fig. 11, *(i)* STAC can reduce bandwidth consumption by up to 8.7-69.6%, with an average of 5.3-34.2%, or boost accuracy (mIoU or PA) by up to 0.7-9.5%, with an average of 0.2-3.7%. These performance improvements still come from the spatial adaptive compression and inter-frame propagation mechanism. *(ii)* Additionally, AccMPEG also exhibits performance gains over CASVA, with an average bandwidth reduction of 1.2-1.4% or accuracy improvement of 0.03-1.0%. These results illustrate that even low-frequency and coarse-grained non-uniform compression can significantly eliminate redundancy in videos, further highlighting the crux of spatial adaptive compression strategies.

STAC vs. GRACE+. Following GRACE [12], we adjust the upperbound B to generate multi-level quantization tables

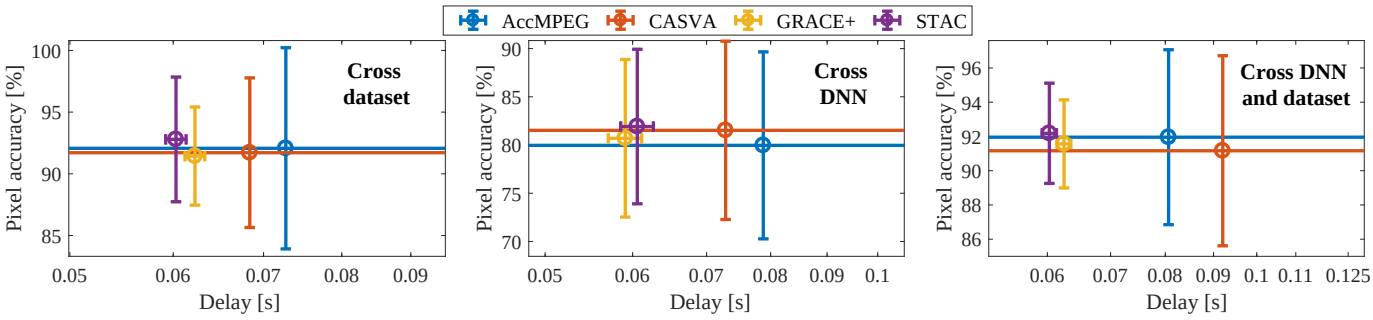


Fig. 14: Evaluation of STAC's generalizability across different video datasets and DNNs.

on the JPEG codec. On this basis, the inter-frame propagation of SS is further combined for GRACE+ to comprehensively adjust fps and B. The experimental results, depicted in Fig. 11, reveal that *(i)* STAC can substantially reduce bandwidth consumption by up to 17.1-75.2%, with an average of 10.1-21.6%, or increase the accuracy by up to 1.0-3.1%, with an average of 0.5-2.5%. These gains arise entirely from spatial adaptive compression and its integration with inter/intra-frame predictive coding, since the generation principle of frequency-sensitive quantization tables and the inter-frame propagation are consistent. *(ii)* The strength of GRACE+ at low bitrates and weakness at high bitrates, compared to AccMPEG and CASVA, are consistent with the feature that inter-frame predictive coding contributes less at low bitrates/fps and more at high bitrates/fps. These results provide further evidence for the necessity and effectiveness of integrating spatial adaptive compression with predictive coding in STAC.

C. Adaptability to Varying Bandwidth

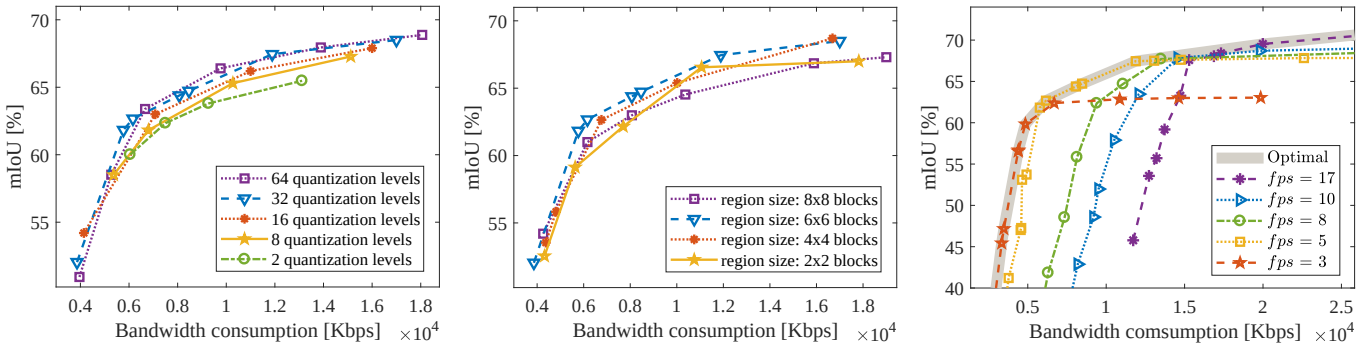
After verifying the superiority in accuracy-bitrate balance, we further evaluate whether these offline improvements can be converted to online gains under varying bandwidth. To this end, we test the adaptability of STAC to different networks including LTE/4G, FCC, and WiFi, with differentiated bandwidth fluctuation characteristics detailed in §VIII-A.

Experimental setup. Since STAC does not predict bandwidth, but instead proposes a bandwidth-aware configuration adaptive scheme, we directly feed the current real bandwidth into different algorithms for fixed-point testing. STAC can also be seamlessly integrated with bandwidth prediction schemes in the real world with unknown bandwidth variation. Besides, due to the lack of dedicated configuration adaptive schemes for AccMPEG and GRACE+, we follow the common practice of constructing offline accurate-bitrate profiles as in §VIII-B. Based on these profiles, AccMPEG and GRACE+ can map real-time bandwidth to configurations that maximize average DNN accuracy. As for CASVA, RL is exploited to train an NN, which takes as input content-related metrics and a bandwidth sequence including the current time. Notably, as the Cityscapes dataset has limited continuous video samples, this experiment is conducted based on the YouTube dataset for training and testing, targeting the DRN-D-22 model.

Results. Fig. 12 exhibits the SS accuracy and the delay of obtaining SS results per frame. The accuracy is evaluated by the PA metric, instead of the mIoU metric, as it is too unstable on a per-frame basis. Notably, compared to AccMPEG and CASVA whose delays gradually increase as the bandwidth decreases, STAC can always ensure real-time performance, i.e., keeping frame delay at the level of frame capturing interval ($\frac{1}{17fps} \approx 0.06$ s). This is because the local inter-frame propagation of SS results completely determines the delay, without waiting for the server-side feedback. Therein, the dominant component is dense optical flow extraction [35], operating in a coarse-to-fine manner to ensure completion as required, regardless of bandwidth, target DNNs and datasets. For AccMPEG and CASVA, the SS results are actually obtained via another pipeline of server-side feedback, including transmission, DNN inference, etc. Among them, we plot only the transmission delay of these algorithms to better compare the adaptability to dynamic bandwidth.

Besides, STAC can also improve accuracy, achieving a 0.9-4.0% gain on different network datasets. This gain is derived partly from its inherently superior compression strategy, and partly from the content-related metrics for reliable bitrate/accuracy estimation. Although the accuracy of STAC is under the influence of DNN inference time on the server, which affects the delay of server-side feedback, and in turn affects the local propagation distances, STAC still achieves better accuracy than other algorithms. As the time for running three different DNN models on the server in our experiment varies from 7 to 40 ms (detailed in §VIII-F), it results in extra propagation distances of only 0.17-0.68 frames on average, which has a limited impact on the accuracy.

A microscopic showcase. Fig. 13 showcases a 100-second video configuration/bitrate adaptive trajectory. By incorporating content-related metrics, STAC and CASVA effectively stabilize the video bitrate to stay close to the available bandwidth without exceeding it. In contrast, AccMPEG, which doesn't take video contents into account, often encounters bandwidth overshoot or wastage (see the red area in Fig. 13), resulting in reduced accuracy. The negative impact of bandwidth wastage on accuracy is evident, while the overshoot impairs accuracy due to the increased delay, as we discard frames that did not complete transmission before the new configuration adjustment to prevent delay accumulation. In comparison, STAC not only improves accuracy through a superior compression strategy, but also makes full use of bandwidth while avoiding accuracy degradation caused by overshoot and long inter-frame propagation. Fig. 13 further depicts the detailed configuration adjustment by STAC under varying bandwidths. Typically, a



(a) Adjusting the number of quantization tables. (b) Adjusting the size of regions. (c) Adjusting the transmission frame rates fps.

Fig. 15: The impact of different parameters in STAC.

decrease in bandwidth corresponds to a reduction in fps or an increase in B.

D. Generalizability across Datasets and DNNs

We further evaluate the generalizability across different datasets, different target DNN models, and both cases.

Experimental setup. We employ the configuration adaptive NN trained in §VIII-C to test its generalizability under various settings of *(i)* different dataset: Cityscapes and DRN-D-22; *(ii)* different model: YouTube and DRN-D-38; and *(iii)* different model and dataset: Cityscapes and DRN-D-38. Regarding AccMPEG and GRACE+, we still follow the common practice of constructing offline accuracy-bitrate profiles for different settings. As for CASVA, limited by insufficient training samples in the Cityscapes dataset, the RL model trained in §VIII-C is directly used under settings of *(i)* and *(iii)*, only retrained under *(ii)* setting. Furthermore, this experiment only adopts LTE/4G network dataset. Other setups remain consistent with §VIII-C.

Results. Fig. 14 shows that STAC can still achieve accuracy gains compared to other algorithms. Specifically, the accuracy gain of STAC, compared to AccMPEG and GRACE+, is greater across DNN, slightly weaker across datasets, and weakest in the experiment involving different datasets and DNN. Besides, STAC is comparable to CASVA in accuracy across different DNN (when CASVA is retrained), but shows noticeable improvements under the other two settings. These quantitative results provide a generalization assessment of the configuration adaptive scheme in STAC, which performs more robustly than that in CASVA. Essentially, several factors contribute to these results. First, the configuration adaptive scheme outputs scaling factors rather than configurations directly, which enhances generalizability. Second, it must be acknowledged that the YouTube and Cityscapes datasets share a high degree of similarity including the scenes, tasks, etc. Once these elements differ significantly such as CamVid dataset, transfer learning is always inevitable. Of course, there is no generalization issue when modifying STAC with JPEG/H.26x quantization tables and offline accuracy-bitrate profiles, which is a common practice in the field.

In addition to the accuracy improvement, STAC can also maintain real-time performance, keeping the delay at the level of frame capturing interval (i.e., $\frac{1}{17\ fps} \approx 60$ ms of our datasets) regardless of the experimental setups. This reliability stems from the timely acquisition of SS results for each frame through inter-frame propagation of previously cached SS results, eliminating the need to wait for server-side feedback. Consequently, only the time spent on local inter-frame propagation impacts the delay, which has little to do with setups like videos, networks and DNNs. In contrast, CASVA and AccMPEG lack this inter-frame propagation, necessitating full pipeline completion involving offloading, DNN inference, and feedback. Their performance is directly affected by videos, DNNs, networks and bitrate-bandwidth mismatches, leading to variability across different setups. Furthermore, GRACE+ integrates inter-frame propagation like STAC, aligning its delay closely with STAC.

After validating the superiority over GRACE+ and CASVA in the above, there is no necessity for further comparisons with JPEG and H.264, as GRACE+ evolves from JPEG and CASVA is a configuration adaptive algorithm based on H.264. For example, without inter-frame propagation, JPEG will render a much larger delay than GRACE+, and even much larger than CASVA and AccMPEG, as JPEG needs to offload consecutive frames in the form of individual images. Without customized quantization tables, the accuracy of JPEG has been confirmed to be much lower than GRACE [12]. Therefore, it is logical to infer that JPEG and H.264, which are theoretically inferior to GRACE+ and CASVA, would demonstrate worse performance, let alone compared to STAC.

E. STAC Deep Dive

There are several parameters in STAC that may affect its performance. In this subsection, we conduct a series of experiments to test their impacts and find the optimal settings.

Adjusting the number of quantization table levels. In theory, increasing the number of quantization tables can expand the accuracy limits, but introduces more transmission load required for compression strategy synchronization and the computational cost of selecting quantization tables. Fig. 15a indicates that the overall performance improves significantly as the number of quantization tables increases from 2 to 32. Whereas, when the number further increases to 64, the performance plateaus and even shows a trend of inferiority

in the low bandwidth consumption regime. The fundamental reason is that beyond a certain point, the benefits of increasing coverage and granularity resulting from additional quantization tables become negligible, yet the transmission load arising from compression strategy synchronization continues to rise. As a result, we set the number of quantization tables to 32, which conserves computational resources while ensuring optimal accuracy-bitrate balance.

Adjusting the size of regions. The impact of region size is intricate, with smaller regions enabling finer granularity and higher accuracy limits, but also incurring frequent quantization jitters in frame planes that diminish compression efficiency. Moreover, smaller regions increase transmission loads of compression strategy synchronization and raise computational costs of quantization table selection. To provide further insights into this impact, we conduct a series of experiments on region sizes ranging from 2×2 to 8×8 blocks, with a fixed block size of 4×4 pixels to reduce experimental redundancy. As shown in Fig. 15b, the overall performance improves when the size increases from 2×2 to 6×6, but begins to decline as it reaches 8×8 blocks. We analyze that the size of 6×6 blocks is the most appropriate, as it doesn't exhibit as obvious sensitivity variation as 8×8 blocks that affect accuracy, nor does it diminish compression efficiency, increase the burden of compression strategy synchronization, or raise computation costs as much as sizes of 2×2 and 4×4 blocks.

Adjusting the transmission frame rates. While §VIII-B has investigated the accuracy-bitrate balance, this overall curve cannot intuitively characterize the impact of fps and B. It remains ambiguous as to when it is most beneficial to increase fps or B, and whether adjusting one alone can yield comparable performance. Therefore, we present in Fig. 15c the accuracy-bitrate curves achieved by adjusting B under various fps. Notably, the accuracy-bitrate curves corresponding to different fps collectively constitute the optimal performance. In low bandwidth regimes, decreasing fps yields greater accuracy gains than blindly increasing B, as the accuracy loss caused by quantization exceeds that caused by inter-frame propagation of SS. As the bandwidth gradually increases, the effects of fps and quantization become increasingly intertwined. Appropriately increasing fps in combination with B reductions helps maintain accuracy at a good level. Ultimately, when the bandwidth is sufficient, 17 fps is the best choice to avoid any interference caused by inter-frame propagation.

Benefits of configuration adaptive module. To quantify this module's benefits, we further conduct an ablation experiment by solely altering the configuration adaptive scheme. Baseline algorithms include STAC with the offline accurate-bitrate profile (denoted as STAC-offline), STAC excluding spatial complexity c (denoted as STAC-w/o-c), STAC excluding temporal complexity d (denoted as STAC-w/o-d), and STAC utilizing complexity metrics in CASVA (denoted as STAC-CASVA). The experiment involves two DNNs (DRN-D-22, DRN-D-38), two video datasets (Cityscapes, YouTube videos), and three network datasets (LTE/4G, FCC, and WiFi).

Fig. 16 shows that STAC achieves superior overall performance. It enhances average accuracy by 10.3%, 5.9%, and 4.3% compared to STAC-offline, STAC-w/o-c, and STAC-w/o-d, while remaining at the same level as STAC-CASVA. This improvement stems from STAC's precise alignment of video bitrates with bandwidth, enabled by effective video complexity extraction that accurately captures dynamic video content. Additionally, regarding the computational overhead, STAC-offline adjusts video configurations based on an offline accurate-bitrate profile with near-zero computational complexity. For other algorithms, the primary computational overheads of their configuration adaptive modules come from the extraction of content-related metrics, while the overheads of the NNs for configuration adjustments are minimal. Therein, STAC, STAC-w/o-d and STAC-w/o-d extract content-related metrics with the low computational complexity of $\mathcal{O}(M \times fps)$, contrasting to STAC-CASVA's resource-intensive extraction through pre-encoding future frames [21]. Due to the real-time requirement, the number of future frames to be pre-encoded is limited to one in our experiment. The computational complexity is $\mathcal{O}(M \times P)$, where P denotes the operations per pixel, typically ranging from 100-500 FLOPs. Furthermore, all algorithms achieve similar delays due to the same inter-frame propagation of SS results regardless of configuration adaptive modules.

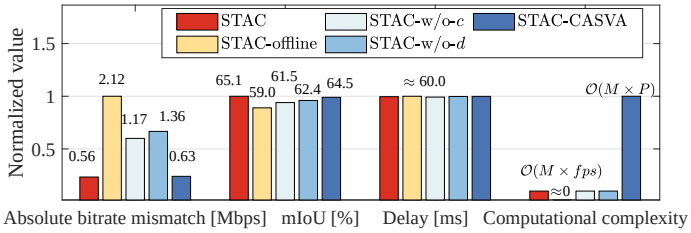


Fig. 16: Benefits of the configuration adaptive module in STAC.

F. Computational Overhead

Table II lists the computational overhead and time consumed for different modules in STAC. The spatial adaptive selection module (§IV-B) costs about 6-36 ms on the server, with the major computational overhead coming from gradient calculations. In the NN-based configuration adaptive module (§VI), the main component, i.e., NN inference for configuration adjustments, runs on the server with a low computational overhead of 20 KFLOPs, while the content-related metrics for input are extracted on the end device with a computational complexity of $\mathcal{O}(M \times fps)$, taking less than < 5 ms. Regarding the temporal adaptive module (§V) on the end device, the computation is dominated by the dense optical flow extraction and integration with predictive coding. The former takes about 55-60 ms, while the latter takes about 20 ms. Other computational overheads and delays, such as target DNN inference, transmission, and encoding/decoding, are all inherent in compressive streaming systems/algorithms for vision tasks and are not introduced by STAC.

It is noteworthy that only the dense optical flow extraction and pixel-level mapping affect the delay in obtaining real-time SS results in STAC. The pixel-level mapping incurs a negligible time cost, while the dense optical flow extraction operates in a coarse-to-fine manner for real-time assurance,

TABLE II: THE ONLINE COMPUTATIONAL OVERHEAD AND DELAY ANALYSIS.

Components			Computational complexity	Cost time	Delay for SS	Devices
DNN inference			298-1790 GFLOPs	7-40 ms	/	Server
Transmission (+feedback)			/	0-330 ms	/	/
Encoding			/	10-50 ms	/	End device
Decoding			/	$<$ 3 ms	/	Server
Added by STAC	Spatial adaptive selection (§IV-B)	Gradient calculation	274-1643 GFLOPs	6-36 ms	/	Server
		Compression strategy generation	$\mathcal{O}(M)$, 3-20 MFLOPs	$\ll$ 1 ms	/	
	Temporal adaptation (§V)	Optical flow extraction	/	55-60 ms	55-60 ms	End device
		Pixel-level mapping	$\mathcal{O}(M)$, 3-20 MFLOPs	$<$ 3 ms	$<$ 3 ms	
		Integration with predictive coding	$\mathcal{O}(M)$	$\approx$ 20 ms	/	
	NN-based configuration adaptation (§VI)	Extraction of content-related metircs	$\mathcal{O}(M \times fps)$	$<$ 5 ms	/	End device
		NN inference	20 KFLOPs	$<$ 1 ms	/	Server

maintaining the delay at the level of frame capturing interval (i.e., $\frac{1}{17fps} \approx 60$ ms in our datasets).

IX. Related Work

Task-oriented compression algorithm. Existing compression algorithms [9]–[11] for OD tasks generally start by locally identifying regions with target objects, and then offload them with high quality. To name a few, Liu et al. [10] exploit a dynamic RoI encoding technique to compress and offload regions with potential targets. EdgeDuet [11] further prioritizes the offloading of tiles that contain more objects than others. However, these algorithms are a poor fit for SS tasks, where sensitive regions are not obvious and concentrated like OD tasks. Many SS-applicable algorithms exploit uniform spatial compression strategies [12]–[16] or simply discriminate compression ratios according to human visual quality [20]. To name a few, GRACE [12] exploits DNN's gradients (i.e., sensitivity) w.r.t. different frequency and color components to optimize the quantization tables. However, the same quantization table is applied uniformly to all blocks without spatial adaptation. Runespoor [13] utilizes super resolution technique to selectively compensate for frames with low semantic segmentation accuracy, which we can combine in future works. Several studies [14]–[16] build autoencoders for OD or SS tasks. To be specific, Nakanoya et al. [14] learn task-relevant representations of sensory data for robotic tasks. Starfish [15] constructs a loss-resilient DNN-based image compressor that is optimized for task objectives. Qin et al. [16] further describe task-oriented autoencoders as a key technology for semantic communication. However, these encoders are still equivalent to using fixed compression strategies without adaptation in spatial and temporal domains.

Furthermore, a handful of SS-oriented non-uniform compression algorithms [17]–[19] have also been proposed recently. The most relevant is AccMPEG [18], which also uses DNN's gradients as spatial sensitivity metrics, but is limited to two-level compression that switches between high/low quality based on a gradient threshold. Moreover, AccMPEG relies on a separate DNN to predict spatial sensitivity online on the end device, imposing considerable computational overhead. DDS [17] is also limited to two-level compression strategies, and necessitates secondary offloading based on confidence feedback to enhance critical video content, which is not applicable to real-time systems. Orchestra [19] requires a prerequisite that spatial sensitivity remains stable over time and is inherent to the position on the frame plane, which is limited to instance segmentation and static cameras.

Bitrate/configuration adaptive algorithm. Based on advanced compression strategies, developing compatible configuration adaptive schemes plays a key role in real-world performance. Typically, many task-oriented non-uniform compression algorithms [9]–[11], [13], [17]–[19] build their accuracy-bitrate profiles by tuning compression configurations therein. That is to say, they can automatically adapt their non-uniform compression strategies to estimated bandwidth. To name a few, DDS's adaptive feedback control system [17] dynamically tunes the low and high-quality configurations based on the estimated available bandwidth. Taking the varying content into account, AdaMask [9] further customizes the offline accuracy-bitrate profiles according to online video content. Besides, specialized task-oriented configuration adaptive algorithms [21]–[24], [37] adjust only basic configurations such as fps, resolution and uniform QP on top of H.26x, without delving into non-uniform compression. Therein, FrameHopper [22] exploits RL to determine how many successive frames can be skipped. Zhang et al. [23] jointly adjust fps, resolution and bandwidth allocation for video analytics. A^2 [24] tunes the frame resolution and task model complexity to achieve a trade-off between latency and inference quality. To further adapt to dynamic contents, CASVA [21] relies on video pre-encoding to obtain indicators for bitrate changes, while DAO [37] builds a lightweight DNN model to formulate the relationship among video content, resolution, accuracy and bitrate.

Moreover, existing quality of experience (QoE)-driven bitrate adaptive streaming algorithms [45]–[49] can also integrate non-uniform compression on top of H.26x's built-in bitrate allocation scheme [20]. Division et al. [20] propose a block-wise bitrate allocation scheme for H.264 to ensure the fidelity, which builds quadratic rate-distortion models for each block to adjust QPs non-uniformly. In this H.26x modality, most QoE-oriented bitrate adaptive algorithms [45]–[49] can automatically map their bitrate decisions to per-frame, per-block QPs based on spatial diversity. Therein, various techniques such as the proportional integral derivative (PID) method [48], Lyapunov optimization [46] and learning approaches [45], [49] are employed to optimize QoE, consid-

ering factors like environments [45], motion states [49], chunk sizes [47], etc.

Semantic video segmentation. Existing DNN-based semantic video segmentation can be summarized into two streams. The most straightforward way is to simply run semantic image segmentation on each frame [12] or keyframes [29], [30]. The second stream is to take consecutive frames as input [27], [28] and leverage temporal correlation to alleviate the excessive computational overhead. For example, Mahasseni et al. [27] leverage the LSTM network and Clockwork [28] share DNN features between frames. Recent works [29], [30] extract dense optical flows to propagate SS results from keyframes to the target frame, either for real-time SS of high definition videos [29], or for creating more pixel-wise labels to scale up SS training datasets [30]. Our design also follows a similar framework, i.e., propagating both compression strategies and semantic segmentation results across frames.

X. Discussion

Joint optimization of target DNNs. One may wonder if it is possible to jointly optimize target DNNs and compression strategies. The common way is to train an end-to-end auto-encoder [14]–[16], where both two functions are achieved by DNNs. However, the major drawbacks include higher computational complexity, lower robustness, etc. Besides, it is also not practical to first decide a particular compression strategy, and then train the target DNN based on this criteria. On the one hand, it restricts the DNN's applicability and generalization, making it uneconomical to invest substantial time and data in training. In reality, most DNNs for vision tasks [43], [44] are trained on original raw video datasets, rather than being specifically tailored for a particular compression strategy. On the other hand, SS-oriented compression strategies designed without target DNN's sensitivity as a guide are usually heuristic (e.g., boundary-guided compression in Fig. 1), which renders much redundancy.

Generalization. The generalization relates to both the target DNN and STAC, and we discuss this issue in three cases: (1) When the target DNN can be directly employed in changing video environments, STAC also works well. (2) When the target DNN undergoes continuous tuning or replacement in response to environment changes, the generalization performance of STAC is evaluated in §VIII-D. Alternatively, STAC can further enhance its generalizability by dynamically updating the quantization tables and NN-based configuration adaptive module using new samples. Theoretically, this involves significantly fewer samples than tuning the target DNN, due to the lightweight characteristic. Another approach is to simplify STAC with JPEG/H.26x quantization tables and offline accuracy-bitrate profiles, which is a common practice in this field. (3) When the DNN cannot generalize or adapt its parameters to changing video environments, STAC can hardly improve the generalization as well, as it is customized for the target DNN.

Application scope. Technically, STAC can be applied to all DNN-based vision tasks, as they all have non-uniform sensitivity with respect to the pixel plane. Yet, the performance gain is significant on pixel-based tasks like SS and relatively insignificant on bounding box-based tasks like OD. When it comes to OD, the sensitive regions are often clearer and more concentrated. Most algorithms [9]–[11] transmit only the region within the bounding box, which typically consumes a small fraction of bits. In this case, using STAC to further extract important contents inside the bounding box may yield minimal performance gain. On the contrary, the sensitive regions of semantic segmentation are usually scattered and indistinct. Employing STAC to discriminate the importance of pixels across the frame plane is more likely to result in significant performance gains.

Characteristics of sensitive regions. Based on DNN's gradients, we analyze that the sensitive regions are closely related to multiple factors including segmentation boundaries, color differences, object sizes, etc. Regarding the color difference, the bottom of the vehicle, the shadow, and the ground in Fig. 4 are almost merged together with subtle color differences, making it challenging to segment them. Therefore, high clarity is always preferred to ensure SS accuracy, exhibiting high sensitivity. The reason lies in the high similarity of image information within the DNN's receptive field between adjacent pixels. If the adjacent pixels happen to be around the boundary, the high similarity is harmful. However, if the adjacent pixels belong to the interior of an object, the high similarity becomes advantageous. In addition, relatively small objects generally have higher sensitivity [17], such as vehicles and people in the distance in Fig. 4. The reason is that small objects are often difficult to fully display their details, making it challenging for the DNN to extract sufficient features and segment them from the background.

XI. Conclusion

This paper proposes STAC, which takes the first step to achieve fine-grained video spatio-temporal adaptive compression using DNN's gradients. STAC is tailor-designed for semantic video segmentation, which encounters challenges in compression strategy synchronization load, time-varying video content and bandwidth conditions. To tackle these challenges, we design a region-based spatial adaptive scheme, a temporal adaptive scheme, and an NN-based intelligent configuration adaptive scheme. Experimental results show that STAC achieves up to 63.7-75.2% reduction in bandwidth consumption or 3.1-9.5% increase in semantic segmentation accuracy. Moreover, STAC can adapt to various types of networks with differentiated fluctuation characteristics, improving accuracy while guaranteeing real-time performance.

References

[1] Xuedou Xiao, Juecheng Zhang, Wei Wang, Jianhua He, and Qian Zhang. DNN-driven compressive offloading for edge-assisted semantic video segmentation. In *Proc. IEEE INFOCOM*, pages 1888–1897, 2022.

[2] Lucy Booker. How is AI changing the security sector? https://www.ifsecglobal.com/video-surveillance/how-is-ai-changing-the-security-sector/. Online; accessed March 9, 2023.

[3] Dhanoop Karunakaran. Semantic segmentation - Udaity's self-driving car engineer nanodegree. https://medium.com/intro-to-artificial-intelligence/semantic-segmentation-udaitys-self-driving-car-engineer-nanodegree-c01eb6eaf9d. Online; accessed Jun 16, 2018.

[4] Cogito Tech LLC. How to improve computer vision in AI drones using image annotation services? https://www.cogitotech.com/blog/how-to-improve-computer-vision-drones-image-annotation/. Online; accessed January 27, 2020.

[5] Ben Lang. Meta shows new progress on key tech for making AR genuinely useful. https://www.roadtovr.com/meta-shows-progress-ai-segmentation-ar-sam-model-sa-1b/. Online; accessed Apr 11, 2023.

[6] Alena Arsenova. Hair segmentation for realistic virtual hair color try on. https://www.banuba.com/blog/hair-segmentation-virtual-hair-color-try-on. Online; accessed October 12, 2022.

[7] CAROLINE. Kivisense officially launches its AR clothing try-on. https://tryon.kivisense.com/blog/ar-clothing-try-on-launch/. Online; accessed June 7, 2022.

[8] Qing Li, Shangguang Wang, Ao Zhou, Xiao Ma, Fangchun Yang, and Alex X Liu. QoS driven task offloading with statistical guarantee in mobile edge computing. *IEEE Transactions on Mobile Computing*, 21(1):278–290, 2020.

[9] Shengzhong Liu, Tianshi Wang, Jinyang Li, Dachun Sun, Mani Srivastava, and Tarek Abdelzaher. AdaMask: Enabling machine-centric video streaming with adaptive frame masking for DNN inference offloading. In *Proc. ACM MM*, pages 3035–3044, 2022.

[10] Luyang Liu, Hongyu Li, and Marco Gruteser. Edge assisted real-time object detection for mobile augmented reality. In *Proc. ACM MobiCom*, pages 1–16, 2019.

[11] Zheng Yang, Xu Wang, Jiahang Wu, Yi Zhao, Qiang Ma, Xin Miao, Li Zhang, and Zimu Zhou. EdgeDuet: Tiling small object detection for edge assisted autonomous mobile vision. *IEEE/ACM Transactions on Networking*, 2022.

[12] Xiufeng Xie and Kyu-Han Kim. Source compression with bounded DNN perception loss for IoT edge computer vision. In *Proc. ACM MobiCom*, pages 1–16, 2019.

[13] Yiding Wang, Weiyan Wang, Duowen Liu, Xin Jin, Junchen Jiang, and Kai Chen. Enabling edge-cloud video analytics for robotics applications. *IEEE Transactions on Cloud Computing*, 2022.

[14] Manabu Nakanoya, Sandeep Chinchali, Alexandros Anemogiannis, Akul Datta, Sachin Katti, and Marco Pavone. Task-relevant representation learning for networked robotic perception. *arXiv preprint arXiv:2011.03216*, 2020.

[15] Pan Hu, Junha Im, Zain Asgar, and Sachin Katti. Starfish: resilient image compression for AIoT cameras. In *Proc. ACM SenSys*, pages 395–408, 2020.

[16] Zhijin Qin, Xiaoming Tao, Jianhua Lu, and Geoffrey Ye Li. Semantic communications: Principles and challenges. *arXiv preprint arXiv:2201.01389*, 2021.

[17] Kuntai Du, Ahsan Pervaiz, Xin Yuan, Aakanksha Chowdhery, Qizheng Zhang, Henry Hoffmann, and Junchen Jiang. Server-driven video streaming for deep learning inference. In *Proc. ACM SIGCOMM*, pages 557–570, 2020.

[18] Kuntai Du, Qizheng Zhang, Anton Arapin, Haodong Wang, Zhengxu Xia, and Junchen Jiang. AccMPEG: Optimizing video encoding for accurate video analytics. In *Proc. MLSys*, pages 450–466, 2022.

[19] Wufan Wang, Bo Wang, Lei Zhang, and Hua Huang. Sensitivity-aware spatial quality adaptation for live video analytics. *IEEE Journal on Selected Areas in Communications*, 2022.

[20] ZG Li, Wen Gao, Feng Pan, SW Ma, Keng Pang Lim, GN Feng, Xiao Lin, Susanto Rahardja, HQ Lu, and Yan Lu. Adaptive rate control for H.264. *Journal of Visual Communication and Image Representation*, 17(2):376–406, 2006.

[21] Miao Zhang, Fangxin Wang, and Jiangchuan Liu. CASVA: Configuration-adaptive streaming for live video analytics. In *Proc. IEEE INFOCOM*, pages 2168–2177, 2022.

[22] Md Adnan Arefeen, Sumaiya Tabassum Nimi, and Md Yusuf Sarwar Uddin. FrameHopper: Selective processing of video frames in detection-driven real-time video analytics. In *Proc. IEEE DCOSS*, pages 125–132, 2022.

[23] Sheng Zhang, Can Wang, Yibo Jin, Jie Wu, Zhuzhong Qian, Mingjun Xiao, and Sanglu Lu. Adaptive configuration selection and bandwidth allocation for edge-based video analytics. *IEEE/ACM Transactions on Networking*, 30(1):285–298, 2021.

[24] Jingyan Jiang, Ziyue Luo, Chenghao Hu, Zhaoliang He, Zhi Wang, Shutao Xia, and Chuan Wu. Joint model and data adaptation for cloud inference serving. In *Proc. IEEE RTSS*, pages 279–289, 2021.

[25] Jeroen Van Der Hooft, Stefano Petrangeli, Tim Wauters, Rafael Huysegems, Patrice Rondao Alface, Tom Bostoen, and Filip De Turck. HTTP/2-based adaptive streaming of HEVC video over 4G/LTE networks. *IEEE Communications Letters*, 20(11):2177–2180, 2016.

[26] Federal Communications Commission. Measuring broadband raw data releases - fixed. https://www.fcc.gov/oet/mba/raw-data-releases. Online; accessed October 12, 2022.

[27] Behrooz Mahasseni, Sinisa Todorovic, and Alan Fern. Budget-aware deep semantic video segmentation. In *Proc. IEEE/CVF CVPR*, pages 1029–1038, 2017.

[28] Evan Shelhamer, Kate Rakelly, Judy Hoffman, and Trevor Darrell. Clockwork convnets for video semantic segmentation. In *Proc. ECCV*, pages 852–868, 2016.

[29] Matthieu Paul, Christoph Mayer, Luc Van Gool, and Radu Timofte. Efficient video semantic segmentation with labels propagation and refinement. In *Proc. IEEE/CVF CVPR*, pages 2873–2882, 2020.

[30] Yi Zhu, Karan Sapra, Fitsum A Reda, Kevin J Shih, Shawn Newsam, Andrew Tao, and Bryan Catanzaro. Improving semantic segmentation via video propagation and label relaxation. In *Proc. IEEE/CVF CVPR*, pages 8856–8865, 2019.

[31] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the H. 264/AVC video coding standard. *IEEE Transactions on circuits and systems for video technology*, 13(7):560–576, 2003.

[32] Jim Bankoski, Paul Wilkins, and Yaowu Xu. Technical overview of VP8, an open source video codec for the web. In *Proc. IEEE ICME*, pages 1–6, 2011.

[33] Gregory K Wallace. The JPEG still picture compression standard. *IEEE Transactions on consumer electronics*, 38(1), 1992.

[34] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. IEEE/CVF CVPR*, pages 3213–3223, 2016.

[35] Till Kroeger, Radu Timofte, Dengxin Dai, and Luc Van Gool. Fast optical flow using dense inverse search. In *Proc. ECCV*, pages 471–488, 2016.

[36] Tiffany Yu-Han Chen, Lenin Ravindranath, Shuo Deng, Paramvir Bahl, and Hari Balakrishnan. Glimpse: Continuous, real-time object recognition on mobile devices. In *Proc. ACM SenSys*, pages 155–168, 2015.

[37] Taslim Murad, Anh Nguyen, and Zhisheng Yan. DAO: Dynamic adaptive offloading for video analytics. In *Proc. ACM MM*, pages 3017–3025, 2022.

[38] VideoLAN. x264, the best h.264/avc encoder. https://www.videolan.org/developers/x264.html. Online, accessed August 23, 2013.

[39] J Utah. Driving downtown - las vegas 4k - usa. https://www.youtube.com/watch?v=DL703lh_my8&t=156s. Online; accessed Jan 30, 2017.

[40] J Utah. Seoul 4K - South Korean wall street - driving downtown. https://www.youtube.com/watch?v=DlLqFGoNFZ8. Online; accessed Feb 3, 2023.

[41] Gabriel J Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2):88–97, 2009.

[42] Fisher Yu. Dilated residual networks. https://github.com/fyu/drn, 2018.

[43] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated residual networks. In *Proc. IEEE/CVF CVPR*, pages 472–480, 2017.

[44] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. BiSeNet: Bilateral segmentation network for real-time semantic segmentation. In *Proc. ECCV*, pages 325–341, 2018.

[45] Chunyu Qiao, Gen Li, Qiang Ma, Jiliang Wang, and Yunhao Liu. Trace-driven optimization on bitrate adaptation for mobile video streaming. *IEEE Transactions on Mobile Computing*, 21(6):2243–2256, 2022.

[46] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K Sitaraman. BOLA: Near-optimal bitrate adaptation for online videos. *IEEE/ACM Transactions on Networking*, 28(4):1698–1711, 2020.

[47] Tong Zhang, Fengyuan Ren, Wenxue Cheng, Xiaohui Luo, Ran Shu, and Xiaolan Liu. Towards influence of chunk size variation on video streaming in wireless networks. *IEEE Transactions on Mobile Computing*, 19(7):1715–1730, 2020.

[48] Yanyuan Qin, Ruofan Jin, Shuai Hao, Krishna R Pattipati, Feng Qian, Subhabrata Sen, Chaoqun Yue, and Bing Wang. A control theoretic approach to ABR video streaming: A fresh look at PID-based rate adaptation. *IEEE Transactions on Mobile Computing*, 19(11):2505–2519, 2019.

[49] Xuedou Xiao, Wei Wang, Taobin Chen, Yang Cao, Tao Jiang, and Qian Zhang. Sensor-augmented neural adaptive bitrate video streaming on uavs. *IEEE Transactions on Multimedia*, 22(6):1567–1576, 2019.