Think before You Leap: Content-Aware Low-Cost Edge-Assisted Video Semantic Segmentation

Mingxuan Yan
mingxuanyan@hust.edu.cn
Huazhong University of Science and Technology

Yi Wang
yi_wang_@hust.edu.cn
Huazhong University of Science and Technology

Xuedou Xiao
xuedouxiao@hust.edu.cn
Huazhong University of Science and Technology

Zhiqing Luo
zhiqing_luo@hust.edu.cn
Huazhong University of Science and Technology

Jianhua He
j.he@essex.ac.uk
University of Essex

Wei Wang*
weiwangw@hust.edu.cn
Huazhong University of Science and Technology

ABSTRACT

Offloading computing to edge servers is a promising solution to support growing video understanding applications at resource-constrained IoT devices. Recent efforts have been made to enhance the scalability of such systems by reducing inference costs on edge servers. However, existing research is not directly applicable to pixel-level vision tasks such as video semantic segmentation (VSS), partly due to the fluctuating VSS accuracy and segment bitrate caused by the dynamic video content. In response, we present *Penance*, a new edge inference cost reduction framework. By exploiting softmax outputs of VSS models and the prediction mechanism of H.264/AVC codecs, *Penance* optimizes model selection and compression settings to minimize the inference cost while meeting the required accuracy within the available bandwidth constraints. We implement *Penance* in a commercial IoT device with only CPUs. Experimental results show that *Penance* consumes a negligible 6.8% more computation resources than the optimal strategy while satisfying accuracy and bandwidth constraints with a low failure rate.

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**.

KEYWORDS

video analytics, edge offloading, video semantic segmentation

ACM Reference Format:
Mingxuan Yan, Yi Wang, Xuedou Xiao, Zhiqing Luo, Jianhua He, and Wei Wang. 2023. Think before You Leap: Content-Aware Low-Cost Edge-Assisted Video Semantic Segmentation. In *Proceedings of the 31st ACM International Conference on Multimedia (MM '23), October 29-November 3, 2023, Ottawa, ON, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3581783.3613808

*Corresponding author.

MM '23, October 29-November 3, 2023, Ottawa, ON, Canada

ACM ISBN 979-8-4007-0108-5/23/10…$15.00
https://doi.org/10.1145/3581783.3613808

1 INTRODUCTION

Recent years have witnessed a growing demand for IoT video analytics. The global IoT video analytics market is projected to increase from $5.32 billion in 2021 to $28.37 billion by 2029 [1]. Pixel-level video labeling task such as video semantic segmentation (VSS) is at the heart of IoT video analytics applications ranging from drones [5, 9], video surveillance [17, 33, 37], augmented-reality [3], traffic scene understanding [7, 15], to traffic safety [4, 11]. However, it is noted that the widespread adoption of VSS will heavily rely on deep learning techniques [32] and that the increasing computational complexity of deep neural networks (DNNs) poses a significant challenge for video understanding on resource-constrained IoT devices.

An emerging solution is to offload the video analytics task to edge/cloud servers. In typical edge-assisted VSS systems, IoT devices stream encoded video segments to the server through a network link with limited bandwidth. The server then performs VSS to meet the users' accuracy requirements. However, there are increasing concerns over the scalability of such systems. The state-of-the-art semantic segmentation models [44, 46, 54, 55] are computationally expensive, even for edge/cloud servers [22, 25, 48]. Furthermore, one server typically serves multiple users [25, 43], which amplifies the computational overhead. Although efforts [12, 21, 25, 48] have been made to reduce the edge inference cost by reusing cached predictions, these methods cannot be applied to pixel-level vision tasks as the pixel association varies greatly between frames, leading to drastic degradation of accuracy [51].

A promising solution to mitigate edge inference costs is to switch between multiple vision models with different costs. Specifically, this approach selects the best combination of the vision model version and compression settings to ensure minimal inference cost while satisfying accuracy and bandwidth constraints. Despite its success in image classification and object detection [22, 43, 53], exploiting this approach for edge-assisted VSS faces grand challenges due to the following major issues.

(i) How to manage the fluctuating VSS performance caused by dynamic video content? To switch between different models, the priority is to monitor their runtime accuracy. However, our measurements indicate that *VSS models experience large accuracy fluctuations over short time windows of tens of seconds due to the changes of video contents.* Recent works [25, 43, 53] adapt the accuracy function by exploiting cheap features such as object sizes and edges,

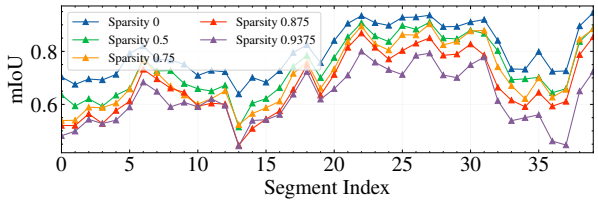


Figure 1: Showcase of the VSS accuracy fluctuation

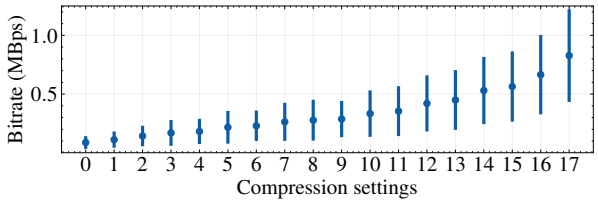


Figure 2: Bandwidth usage distributions

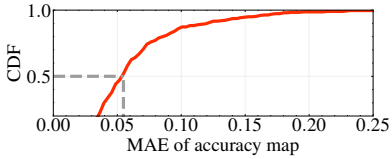


Figure 3: MAE of the reprofiled accuracy functions in the following 20 seconds

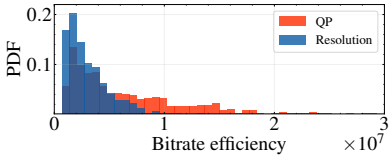


Figure 4: Compare bitrate efficiencies of QP and resolution

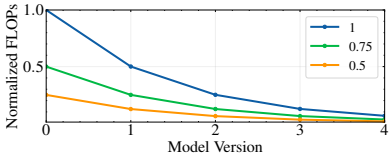


Figure 5: FLOPs of different resolution scaling factors

but they cannot be extended to VSS as semantic segmentation does not have explicit and concentrated regions of interest (RoIs) [45]. Other works [21, 25, 49] reprofile the edge vision model periodically (tens of minutes) and require raw frames fed to the edge server. Increasing reprofile frequency will overwhelm the limited network capacity and is computationally expensive.

(ii) How to tune the codec compression settings? Before being transmitted over a limited network link, video segments must be compressed with codecs such as H.264/AVC by adjusting compression settings, including frame resolution and quantization parameter (QP). Previous works [43, 53] map compression settings to segment bitrate by assuming a fixed relationship between them, *ignoring the variation of video contents that can significantly affect the segment bitrate* as concluded in our measurements. Simply iterating and encoding all possible compression settings would be computationally infeasible. Though recent work [34] employs constant bitrate (CBR) to compress video segments using given bitrates to bypass this problem, it overlooks the trade-off between resolution and QP in bitrate saving and cost reduction, which offers more space for optimization.

(iii) How to configure video segments? Optimizing the configuration of the VSS model version and compression settings is challenging. Previous works [22, 43, 53] formulate and solve the cost reduction as a mixed-integer programming problem requiring accurate configuration mapping to model accuracies. However, the complex, *context-dependent accuracy function of VSS makes accurate modeling challenging*, as mentioned above. Moreover, the optimization is even more challenging due to the *inherent conflict between optimization goals* of minimizing the inference cost and satisfying the accuracy and bandwidth usage constraints.

Given the above practical issues, we present *Penance*, the first content-aware and low-cost edge-assisted VSS system. First, to monitor runtime VSS performance, *Penance* exploits the most recent video frame's predicted softmax probabilities, from where a deep neural network (DNN) extracts an embedding that represents the runtime edge model performance. Second, to estimate the fluctuating segment bitrate, another special DNN is devised by exploiting the prediction mechanism of the H.264/AVC codecs to predict bandwidth usage for each compression setting using raw video frames. Finally, a deep reinforcement learning (DRL) model is presented to optimize the configuration for each segment, considering the accuracy and bandwidth constraints. *Penance* is designed to be lightweight and can be deployed on general IoT devices equipped with only CPUs.

Contributions. *(i)* We investigate the challenge and impact of dynamic video content on edge-assisted VSS systems. *(ii)* We propose *Penance*, the first low-cost edge-assisted VSS system for resource-constraint IoT devices by adapting both compression settings and edge model selection. *(iii)* We implement *Penance* on a commercial IoT device with only CPUs and evaluate its performance with baseline methods. The experiments show that our solution significantly lowers the edge inference cost while strictly adhering to all constraints.

The remainder of this paper is organized as follows. §2 introduces our measurement studies that motivates the design of *Penance*. §3 introduces the key ideas on the system design. §4 presents the implementation and evaluation of *Penance*. §5 gives a literature review of related works, followed by a conclusion in §6.

2 MEASUREMENT AND MOTIVATION

2.1 Measurement Setup

2.1.1 Semantic Segmentation Networks. This paper adopts the popular PSPNet [55] with the ResNet-50 backbone as the original VSS model. To obtain edge models with different inference costs, we resort to the DNN pruning technique [13] and derive 4 models from the original model with sparsity factors of 0.5, 0.75, 0.875, and 0.9375 respectively, whose computational overhead is inversely proportional to their sparsity. For simplicity, we name them as model 0 to 4, respectively.

2.1.2 Dataset. We conduct our measurement on BDD100k [47], a state-of-the-art large-scale diverse video dataset that contains 100K 1280x720 challenging videos with 40 seconds each, covering different scenarios such as urban and rural areas, highways, tunnels, etc. We first train and prune the 5 models with different sparsities on the training set. Then we randomly choose 20 videos spanning 800 seconds from the test set to conduct our measurement. We split the videos into 1-second segments with a frame rate of 15 fps and compress them using standard H.264/AVC to obtain varying bandwidth usages. We choose quantization parameter (QP) and frame resolution as knobs to control video quality, which are typical video encoding settings in video analytics [14, 45, 52]. Specifically,

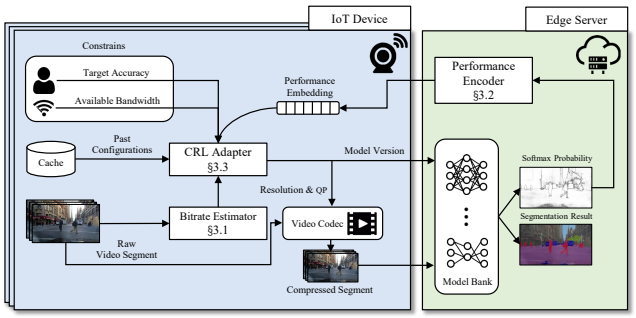


Figure 6: *Penance* Overview

we compress the original videos into 18 variances, with QPs ranging from 20 to 30 and resolution scaling factors of 1, 0.75, and 0.5. We call the combinations of edge model versions and compression settings "*configurations*". As there is only one frame annotated in each video, we follow the practices of previous works [21, 49] and regard predictions of the most expensive configuration (1280x720, QP=20 and model 0) as ground truths. We use the standard mean intersection over union (mIoU) metric to evaluate the performance of VSS models.

2.2 Impact of Dynamic Video Contents

2.2.1 Impact on Model Performances. We first investigate the runtime performances of the edge VSS models. In Fig. 1, we showcase the accuracy fluctuation of one 40-second video. It can be seen that the runtime model accuracy experiences drastic changes with time. We can further observe that the relative ratio between the accuracies of different models also changes with time. To handle accuracy variance, existing works [21, 25, 49] re-profile using raw video frames periodically to update the accuracy function. Following this method, we profile one frame every 20 seconds and update the accuracy function. We then calculate the distribution of its Mean Absolute Error (MAE) with respect to the ground-truth accuracies of the following 20 seconds in Fig. 3. As shown in the plot, in **50%** cases, the MAE surpasses **0.05**. As a comparison, the average mIoU difference of our adjacent model version is **0.042**, indicating that the accuracy estimation error starts to impact the model selection. As such, it suggests that *we need to handle the accuracy fluctuation at segment granularity.*

2.2.2 Impact on the Selection of Compression Settings. Before streaming video frames to the edge/cloud server, we first need to decide the compression settings for each segment to satisfy the bandwidth constraint. Unfortunately, the encoded segments' bitrate is not solely decided by the compression settings but also by the video contents. Fig. 2 shows the variation of segment bitrate of all possible compression settings. It can be seen that the actual bitrate varies drastically across segments, which indicates *we need to consider the impact of varying video contents when we estimate the segment bitrate.*

We further observe a trade-off between QP and resolution. First, we evaluate their efficiency in trading accuracy for bandwidth saving. Specifically, we measured each segment's bitrate efficiencies, i.e., the ratio between bitrate saving and accuracy drop when tuning to a lower quality. Higher bitrate efficiency implies that the knob can trade less accuracy for the same bandwidth saving. Interestingly, as shown in Fig. 4, *QP generally outperforms resolution, proving its efficiency in compressing video volumes with minor accuracy drop.* We also noted significant performance fluctuations on both knobs, indicating a substantial impact from video content. Intuitive thought is that we can always choose to degrade QP instead of a resolution to preserve accuracy as much as possible. However, the *degrading resolution provides an additional opportunity to reduce the inference cost*, as shown in Fig. 5. As such, the trade-off between QP and resolution offers additional room for optimization.

3 DESIGN

3.1 System Overview

Fig. 6 shows the overall design of *Penance*. The proposed system has three major components:

Bitrate Estimator. When a new raw video segment is ready, the bitrate estimator predicts its bandwidth usage under all compression settings. Moreover, the bitrate estimator provides information about the current scene dynamics to the CRL adapter.

Performance Encoder. On the server side, the performance encoder utilizes the softmax probabilities to generate a performance embedding, which is then fed back to the IoT client to aid the decision-making process of the CRL adapter.

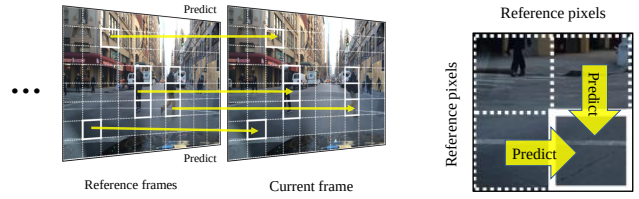
CRL Adapter. The CRL adapter is a DRL model that utilizes estimated bandwidth usage, historical configurations, and performance embedding to choose the compression settings and edge model version for the upcoming video segment. This selection is based on minimizing the inference cost while maintaining target accuracy within the constraints of the available bandwidth.

3.2 Bitrate Estimator

In this section, we revisit the prediction scheme of H.264/AVC codec. Then we elaborate on the design of the proposed bitrate estimator and training method.

3.2.1 Dive into H.264/ACV Prediction Scheme. We start by reviewing the encoding scheme of H.264/ACV. The encoder processes video frames in units of *macroblocks* consisting of 16x16 or 4x4 pixels. It forms the basic units of the *prediction* scheme of H.264/ACV, which exploits the content redundancy to save bitrates. As illustrated in Fig. 7, H.264/AVC adopts two types of prediction: *intra-prediction* and *inter-prediction*. Intra-prediction is used to predict a block of pixels within the same frame. It exploits the spatial correlation between neighboring pixels within the same frame to predict the current block. Inter-prediction is used to predict a block of pixels from a previously encoded frame and uses *motion estimation* to find a block of pixels in a previously encoded frame similar to the current block. Once the reference list between macroblocks is constructed, the encoder subtracts the prediction from the current macroblock to form a residual, which consumes much fewer bitrates after quantization and entropy encoding.

Above them, video frames can be categorized into *I-frames, P-frames, and B-frames*. I-frames contain only macroblocks encoded with intra-prediction, while P-frames and B-frames consist of both macroblocks with intra-prediction and inter-prediction. Accordingly, in Fig. 8, we plot the distribution of frame types for each frame



(a) Inter-prediction (b) Intra-prediction

Figure 7: Illustrations of H.264/AVC prediction scheme.

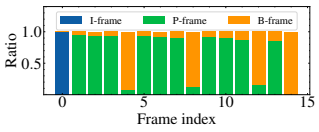


Figure 8: Distribution of frame types.

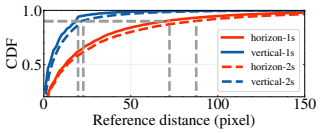


Figure 9: Distribution of motion range.

position with video segments encoded in §2.1.2. It shows that each frame position has a unique frame type distribution. It indicates that *each frame position has a different status in the H.264/AVC prediction scheme.* For instance, the first frame is always an I-frame, which means all of its macroblocks use intra-prediction. As opposed, the tailing frame is always B-frame, which suggests it mainly consists of macroblocks with inter-prediction.

We further investigate the motion estimation scheme, which involves searching for the best match within a search window around the current macroblock. Typically, the search window size is limited by a max iteration time [2] for efficiency. We plot the distribution of the motion range in the x and y axis of both 1-second and 2-second segments in Fig. 9. We can observe that the maximum motion range for both segment lengths is below **100** pixels in **95%** of the cases. It indicates that instead of adopting a deep model with a large reception field that covers the full picture (1280x720), *a relatively shallow network with a smaller reception field is sufficient for capturing the bitrate fluctuation brought by video motion.*

3.2.2 Model Design. Inspired by the above observations, we propose a lightweight model based on the convolutional neural network (CNN) that accurately predicts segment bandwidth usage with raw video frames. As depicted in Fig. 10, the bitrate estimator takes in the raw video frames down-sampled by half and predicts the segment bitrate under all compression settings. Our bitrate estimator has two unique designs. First, instead of using expensive 3D convolution or sliding a single 2D convolution layer across all frames, we use a group convolution layer (G Conv-N in Fig. 10, where N is the segment frame number) that *assigns dedicated weights to each frame* as the input layer. This design exploits the unique status of each frame position as mentioned in §3.2.1. The dedicated convolution kernel weights make learning distinct features from each frame position possible. Second, following the input layer are two MobileNet blocks [39], where we additionally use dilated convolution on the depth-wise convolution layers to expand the reception field. Then, an average pooling layer with a kernel size of 8 summarizes the features. Eventually, the output feature of the feature extractor has a receptive field of 166 pixels on the original video frame, sufficient to cover the motion estimation searching window.

The predict header is a stack of two fully connected (FC) layers that is further divided into two branches. The first branch predicts

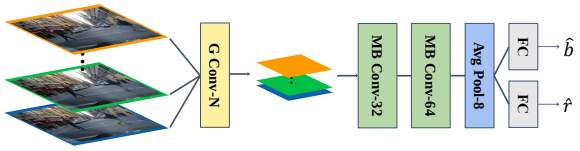


Figure 10: Architecture of the bitrate estimator

the *base bitrate* $\hat{b}$, i.e., the bitrate of the most expensive compression setting. The other branch predicts $\hat{r}$, the ratio between bitrates of the rest configurations settings with respect to the base bitrate.

3.2.3 Model Training. As the prediction of segment bandwidth usage is a regression problem, we use standard Mean Square Error (MSE) to train the model:

$$\theta_{BE} = \arg\max_{\theta} \frac{1}{N} \sum_{i} \left(b_i - \hat{b}_i\right)^2 + \left(r_i - \hat{r}_i\right)^2 \quad (1)$$

Where b_i and r_i are the ground-truth base size and size ratio of segment i, $\hat{b}_i$ and $\hat{r}_i$ are the corresponding network prediction, respectively.

3.3 Performance Encoder

Knowledge about the current segmentation performance is crucial for configuration adaptation in the runtime. Recent work [34] designed for object detection leverages a YOLO-based [38] CNN to predict the runtime accuracy function taking the raw video frames. However, unlike object detection, a sophisticated CNN feature extractor is required to obtain a semantic understanding of the current scenes, which cannot be deployed on general IoT devices only equipped with CPUs. Instead, *we shift this burden to the edge server by extracting features about runtime model performances directly from the edge model outputs of the last available prediction.*

An intuitive approach to implement the performance encoder is directly analyzing the pixel-level prediction confidence, i.e., maximum softmax probability. While this method works for classification [19], it does not correlate well with the mIoU of VSS predictions. As shown in Fig. 11, the upper frame has a mIoU of 0.63 while the lower frame has a mIoU of 0.80, even though they have similar pixel confidence distributions. This is because unlike pixel accuracy, which treats every pixel equally, *mIoU intrinsically weights each pixel according to the number of pixels within the class it belongs to.* The right side of Fig. 11 visually represents class IoU for several classes. It can be observed that the IoU of the "car" and "sky" classes significantly influence the overall mean IoU even though they contain fewer pixels than the class "road".

As such, instead of solely analyzing the pixel confidence, the performance encoder takes *softmax probabilities of all classes* as its input, as shown in Fig. 12. The rationale behind this is threefold. (i) It provides class information that affects the final mIoU, as mentioned above. (ii) It implies the object sizes in the scenes which helps to explore the opportunity of degrading resolution. (iii) It offers the encoder rich information about the candidate classes whose probability is only second to the largest. For example, if the difference between the probabilities of the first and second classes is small, it may be useful to consider both classes as possible predictions. Note that our method regards the edge VSS model as a black box; it only needs the final softmax probability of model outputs.

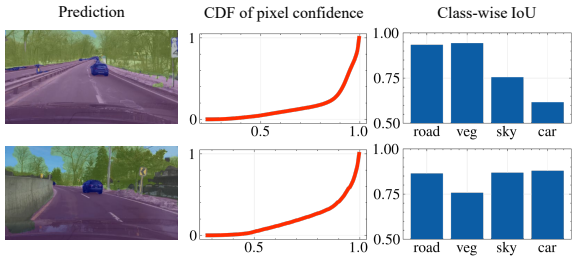


Figure 11: Example demonstrates two frames with comparable pixel confidence distributions but significantly different mean Intersection over Union (mIoU) scores.

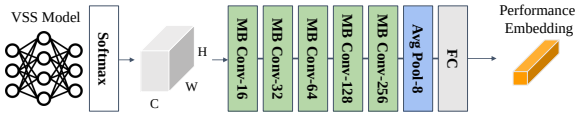


Figure 12: Architecture of the performance encoder

Thus it can be applied to various VSS architectures without further modifications.

We build the performance encoder with 5 MobileNet blocks followed by an average pooling layer as illustrated in Fig. 12. We rescaled the probabilities to Cx320x180 for all resolutions (C is the number of classes), leading to a low inference cost < 0.5G FLOPs. The flattened extracted features will be transmitted back to the IoT device as *performance embedding* to aid the configuration adaption process. This design leverages the rich GPU resource on the edge server and avoids imposing demanding feature extraction on IoT devices. The encoder is optimized end-to-end along with the CRL adapter, which will be introduced in section §3.4.

3.4 CRL Adapter

In this section, we elaborate on the design of the CRL Adapter. We first formulate the problem theoretically and then describe the details of our solution.

3.4.1 Problem Formulation. In *Penance*'s framework, the captured raw video frames are encoded and streamed in segments of T seconds, which is the basic unit of configuration adaptation. When segment i is ready, the goal of the CRL Adapter is to choose the best configuration that minimizes the edge inference cost of segment i under the constraints of bandwidth and accuracy. We formulate it into the following problem:

$$\begin{aligned} &\min_{r,q,v} C(r,v) \\ \text{s.t.,}\ &b_i(r,q) \le B_i \\ &\lambda_i(r,q,v) \ge A_i \end{aligned} \tag{2}$$

Where C is the cost function. b_i and λ_i are the segment bitrate and average mIoU across frames in segment i, respectively. B_i is the bandwidth of the following T second at the uploading time of segment i, and A_i is the target accuracy set by the user. r, q, v is the selected resolution, QP, and edge model version, respectively. If the device fails to upload segment i in T seconds, it will drop segment i and start to stream segment $i+1$ instead. This strategy avoids draining the on-device video buffer and limits the queuing delay [34]. The accuracy of the incomplete segment is zero, as the server cannot decode it.

In some extreme cases, there is no solution to problem (2). For instance, the target accuracy cannot be achieved when the bandwidth is too low. In this situation, we hope to optimize the following problem instead:

$$\begin{aligned} &\max_{r,q,v} \lambda_i(r,q,v) \\ \text{s.t.,}\ &b_i(r,q) \le B_i \end{aligned} \tag{3}$$

It will maximize the inference accuracy under the constraint of bandwidth instead of minimizing inference cost to ensure the quality of serving.

3.4.2 State Space Design. In DRL, the state space represents the environmental information that the policy can perceive at step i. In our solution, the state at step i is defined as:

$$s_i = (A_i, B_i, b_i, p_{i-1}, a_{i-1}, x_i) \tag{4}$$

Where A_i and B_i are the accuracy and bandwidth constraints. b_i is the estimated bitrate of segment i. p_{i-1} is the performance embedding extracted from the last frame of the latest predicted segment (segment $i-1$, for instance). a_{i-1} is the corresponding configuration of segment $i-1$. It represents the resolution, QP, and edge model version set to segment $i-1$. Including this term makes it resistant to the dynamic of final performance encoding caused by different configurations. Moreover, as the performance embedding is extracted from the last frame of segment $i-1$, its effectiveness in the decision of segment i depends on how much the video contents have changed in segment i. Thus, we further append x_i, the features before decision layers of the bitrate estimator, into the state space. As described in §3.2, the bitrate estimator captures the content dynamics of segment i, which makes the feature suitable as hints to inform the policy of the current content dynamics.

3.4.3 Policy Design. The policy is the core component of reinforcement learning. At step i, the policy π receives the current state s_i and produces corresponding action a_i. A policy can be either deterministic or stochastic. A deterministic policy map states a single action, while a stochastic policy map states a probability distribution over actions. We build a stochastic policy for its advantage in highly dynamic environments with imperfect information [41]. Specifically, the policy maps s_i to a_i, a probability distribution of all possible configurations. The action is sampled from the distribution during the training phase, and the action with the highest probability is selected during the testing phase.

Due to the high-dimensional continuous state space, we leverage the powerful approximation ability of neural networks and represent our policy with it. As shown in Fig. 13 we build the policy network with FC layers. Specifically, we first use separate FC layers to extract local features, which are further concatenated and fed to the following layers. The output of the policy network is a vector representing the probability of all configurations.

3.4.4 Reward Design with Constraints. At step, i, the predicted a_i configures the segment i. A corresponding reward r_i is generated by the reward function R, which evaluates the advantage of the move a_i on s_i. The rewards will be used as supervisory signals to optimize the policy network. In standard reinforcement learning

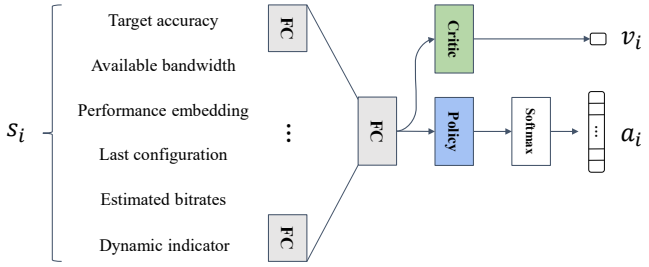


Figure 13: Architecture of the CRL adapter

settings, the optimization goal of the policy network is to maximize the expected sum of discounted rewards $J(\pi)$:

$$\max_{\pi} J(\pi) = \max_{a\sim\pi, s\sim P} \left[\sum_i \gamma^i r_i \right] \tag{5}$$

where P is probability of transitioning to state s^{i+1} from s_i when an action a_i is taken. γ is the reward discount factor and r_i is the reward on taking a_i given s_i. However, considering the constraints in (2) and (3), we need to extent Eq. (5) to the new form of constrained reinforcement learning:

$$\begin{aligned} &\max_{\pi} J(\pi) \\ &\text{s.t., } J^{c_i}(\pi) \le \beta_i \ \forall i \end{aligned} \tag{6}$$

where J^{c_i} and β_i represent the constraints to be met at each step i. A common way to solve Eq. (5) is to solve its Lagrangian relaxation instead [6, 36]. However, these methods require careful tuning of the Lagrangian parameters and are hard to optimize for complex tasks [30]. We thus extend the self-competition reward scheme in [30] to solve problem (2) and (3) and define the reward at step i as the following:

$$r_i = \text{sgn}(C_{EMA} - C_i)\delta_i - \xi_i \tag{7}$$

where

$$\begin{aligned} \delta_i &= \mathbb{1}_{U_{EMA}\le 0, O_{EMA}\le 0, O_i\le 0, U_i\le 0} \\ \xi_i &= \max\{\mathbb{1}_{U_i\ge U_{EMA}}, \mathbb{1}_{O_i\ge O_{EMA}}\} \end{aligned} \tag{8}$$

C_i is the server inference cost. U_i and O_i are $A_i - \lambda_i$ and $b_i - B_i$ that indicates if the constraints are met. It also maintains their exponential moving averages (EMA) as U_{EMA}, O_{EMA}, and C_{EMA}, respectively.

The intuition behind Eq. (7) is to force the policy beat its previous behaviors based on EMA. Specifically, δ_i judges if the accuracy and bandwidth constraints are satisfied. If so, the first term in Eq. (7) will minimize the inference cost by beating its historical performance C_{EMA}. Otherwise, the first term will be ignored, and the second term ξ_i will punish the policy if any constraints are violated. As such, Eq. (7) only allows the policy to optimize for inference cost when the constraints are consistently satisfied.

3.4.5 Policy Optimization. We have defined the state s_t, action a_t, and reward r_t. The next step is to optimize the policy. First, the policy will act as the CRL Adapter in *Penance* and roll out trajectories, i.e., states, actions, and rewards sequences. Then we use the actor-critic approach [31] to calculate the advantage of actions. As shown in Fig. 13, a parameterized value function V_ϕ named critic network, which shares the same architecture with the policy network, is used as a baseline. It takes the same input as the policy network and predicts $v_i = V_\phi(s_i)$ as the expected reward of s_i. The advantage of step i is then calculated as $A_i = r_i - v_i$, and it evaluates how good the action is compared to the average expected reward under s_i. This design reduces variance in the policy gradient and stabilizes the training process. A_i essentially replace r_i in Eq. (5).

Given the advantages, we then optimize the policy network and V_{θ_v} with Proximal Policy Optimization (PPO) algorithm [40] for its sample efficiency and stability. Different from vanilla policy gradient, PPO updates the policy network multiple times with the same batch of collected trajectories using the clipped surrogate objective function:

$$L^{policy}(\theta_k, \theta) = \min\left(R_i(\theta_k, \theta)A_i, \text{clip}\left(R_i(\theta_k, \theta), 1-\epsilon, 1+\epsilon\right)A_i\right) \tag{9}$$

where

$$R_i(\theta_k, \theta) = \frac{\pi_\theta(a_i|s_i)}{\pi_{\theta_k}(a_i|s_i)} \tag{10}$$

θ_k is the old policy parameter before updated, $R_i(\theta_k, \theta)$ is the probability ratio of action a_i between old and new policy given the same state s_i. Through the clipping operation, Eq. 9 carefully updates the policy π_θ while controlling the step by avoiding the behavior of the new policy getting too far from where it starts. Here ϵ is a hyper-parameter that controls the bound of clipping. Then the policy parameters are updated by maximizing the objective function on a collection of trajectories:

$$\theta = \arg\max_{\theta} \sum_i L^{policy} \tag{11}$$

At the same time, the value function V_ϕ is fitted by regression on the reward r_i:

$$\phi = \arg\min_{\phi} \sum_i \left(V_\phi(s_i) - r_i\right)^2 \tag{12}$$

4 EVALUATION

4.1 Experiment Setup

4.1.1 Dataset and Model Training. We randomly select 300 videos totaling 200 minutes to build our training dataset. Following the procedure in §2.1.2, we split the videos into 1-second segments with a frame rate of 15 fps and compressed them using H.264/AVC using 18 different settings. A test set of 60 random videos totaling 40 minutes is built and processed with the same procedure.

We first trained the bitrate estimator. We used a learning rate that gradually reduced from 1e-4 to 1e-6 through cosine annealing, set the weight decay to 1e-4, and continued with a 200-epoch training process. Subsequently, we proceeded to jointly train the frozen bitrate estimator, the CRL adapter, and the performance encoder. At the beginning of each epoch, we uniformly sampled the target accuracy and available bandwidth from $\{0.5, 0.55, \ldots, 0.75, 0.80\}$ and $[0.3, 1.0]$ MBps, respectively. We set the batch size to 32, ϵ to 0.2, and γ to 0.9, which means every action will be evaluated by the performance of the future ten segments. We set the exponential smoothing constant α to calculate EMA to 0.2. The learning rates of the policy and value function are both 1e-4. We trained the policy for 2 million steps.

4.1.2 Implementation. The edge server is an Ubuntu workstation equipped with Intel Xeon Gold 6226R CPU, 128 GB RAM, and a Geforce RTX 4090 GPU. We use a Raspberry Pi 4B [28] as the IoT device that embeds a 1.8 GHz quad-core CPU and 4 GB of RAM.

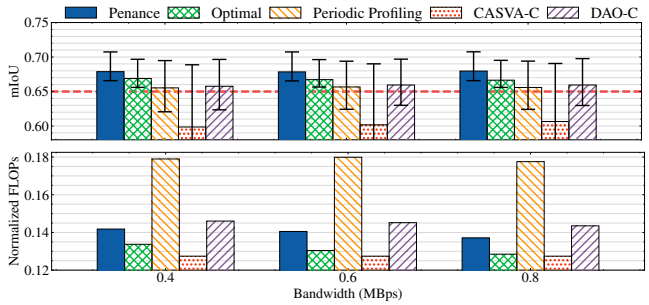


Figure 14: Comparing *Penance* with baseline methods under different bandwidths with target mIoU of 0.65.

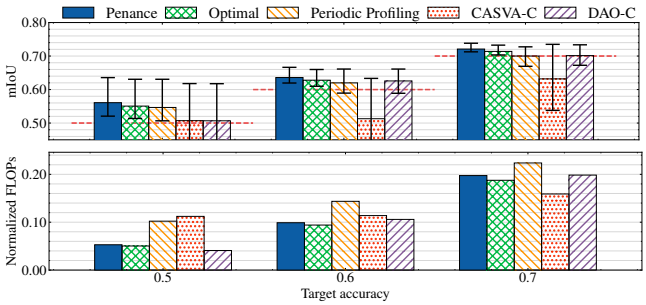


Figure 15: Comparing *Penance* with baseline methods by varying target mIoU with bandwidth limited to 0.6 MBps.

We recompile the H.264 codec in FFmpeg [42] to work with the onboard video-core to accelerate video encoding. Note that the onboard video-core is specially prepared for the codec, and all computations of *Penance* are conducted using CPUs only. We use the Traffic Control (TC) tool of Linux to control the bandwidth between IoT devices and the edge server.

4.1.3 Baselines. We compare *Penance* with the following baselines:

Periodically Profiling: This method streams the first frame of each 40-second video to the server and profiles the accuracies of all 90 configurations. Then it selects the best configuration for the rest of the segments through an exhaustive search. We let it access the ground-truth segment bitrate to cope with the bandwidth constraints.

CASVA-C: CASVA [52] is a DRL video analytic framework that maximizes inference accuracy under bandwidth constraints. It encodes each segment one more time and uses its bitrate as an indicator of video content dynamics. We modified it to cope with our cost-reduction task. It is similar to *Penance* but replaces the performance encoder with the method above. We named this approach as *CASVA-C* and trained it following the same procedure in §4.1.1.

DAO-C: DAO [34] uses a YOLO [38] based accuracy estimator to predict runtime accuracy function with the first frame of each segment. To cope with our settings, we instead trained a MobileNet-based semantic segmentation network [50] on the training set of BDD100k. Then we train the accuracy estimator with transfer learning following the procedure in [34]. At runtime, it uses the same strategy of *Periodic Profiling* to find the best configuration.

Optimal: A perfect model can access every segment's ground-truth accuracy and bitrate. It always chooses the optimal configuration and represents the upper bound of *Penance*'s performance.

4.2 Evaluation Results

4.2.1 Overall Performance. We first investigate the overall performance of *Penance* by fixing the target mIoU to 0.65 and analyzing the achieved mIoU under bandwidth ranging from 0.4 MBps to 0.8 Mbps. We plot the results in Fig. 14(a) where the bars report the *25th* percentiles, and the error bars show the *median* and *10th* to better describe the skewed accuracy distribution. As demonstrated, *Penance* is able to achieve the target accuracy under different bandwidth conditions, with an average failure rate (averaged ratio of segments that fail to achieve the target accuracy across bandwidths) of **4.1%**. Although the median accuracy of *Periodic Profiling* and *DAO-C* surpass the target accuracy, they failed in 36.2% and 34.4% cases, respectively, due to the inaccurate accuracy map estimation. While *CASVA-C* use the same DRL framework as *Penance*, it has a large average failure rate of 60.5%. This result suggests that the correlation between encoded segment size and runtime mIoU is relatively weak. While the segmentation difficulty of the video frames only relates to the current frame contents, the encoded segment size is additionally affected by the inter-frame similarity as illustrated in 3.2.1, making it noisy to represent the frame complexity.

Despite the accuracy assurance, *Penance* additionally consumes only **6.8%** FLOPs compared with *Optimal* as illustrated in Fig. 14(b), where the bars represent the averaged segment FLOPs normalized by the cost of the most expensive model with full resolution inputs. Though only one frame is profiled for every 40-second video, *Periodic Profiling* is still very expensive and spends 36.7% more FLOPs than the *Optimal*. While *CASVA-C* instead consumes fewer FLOPs than *Optimal*, it violates the accuracy constraint. Though *DAO-C* performs slightly better than the other two baselines in maintaining accuracy, it consumes 10.7% more FLOPs than *Optimal*. It validates the effectiveness of the feedback design of our performance encoder. We can also observe that the FLOPs of *Penance* degrade with the increasing bandwidth, indicating the internal balancing between bandwidth consumption and inference cost reduction.

4.2.2 Varying Target Accuracy. We further evaluate the performance of *Penance* given different accuracy requirements. In this experiment, we fix the bandwidth to 0.6 MBps, vary the target mIoU from 0.5 to 0.7, and plot the results in Fig. 15(a), where the bars report the *25th* percentiles, and the error bars show the *median* and *10th*. As shown in the plot, *Penance* successfully catches up with the varying target accuracy, with the failure rate of 3.6%, 3.9%, and 7.3%, respectively, while the other baselines either fail to meet the target accuracy or consume a lot more FLOPs than *Penance*. Interestingly, for the target mIoU of 0.5, *DAO-C* reduces the inference cost aggressively, leading to a failure rate of 40.8%, while *CASVA-C* becomes too conservative that it spends 122.9% more FLOPs compared with *Optimal*. This result can be explained by the fact that with the degradation of frame quality and model complexity, the relationship between configurations and runtime edge model accuracy is even harder to predict as the image noise and model imperfection start to exert more influence on the model outputs. On the contrary, by directly analyzing the final output

Table 1: Performance of segment bitrate estimation

Setting	Algorithm	Mean	$P_{25\%}$	Median	$P_{75\%}$
1s-15fps	AR	25.07%	16.39%	23.71%	30.41%
	Ours	**17.22%**	**6.24%**	**12.96%**	**24.31%**
1s-30fps	AR	28.67%	19.39%	27.16%	35.87%
	Ours	**16.72%**	**6.76%**	**12.19%**	**22.18%**
2s-15fps	AR	31.25%	14.38%	24.73%	37.19%
	Ours	**20.54%**	**6.50%**	**14.00%**	**21.77%**

Table 2: Overhead of *Penance* on Raspberry Pi 4B

	Bitrate Estimator	CRL Adapter	Total
Runtime (ms)	202.44	20.69	223.13
FLOPs (M)	45.79	5.50	51.29

probabilities of edge models, *Penance* can still effectively monitor the runtime performance.

4.2.3 Bitrate Estimation Performance. We then evaluate the performance of the bitrate estimator. Here, we compare it with the solution proposed in recent work [27]. Specifically, it predicts the base bitrate with an autoregression (*AR*) model and uses an offline-learned ratio table to derive the bitrates of the rest compression settings. We adopt the first-order autoregressive model ($AR(1)$) as it outperforms other order settings and regresses the ratio table on the same training set used by the bitrate estimator. We demonstrate the relative bitrate error statistics in the first row of Table 1. The results show that our method outperforms the baseline method. This is because the AR method cannot faithfully forecast the future bitrate when large dynamics exist in the video scenes. Instead, our method predicts per-segment bitrate consumption directly based on the raw video frames aware of the prediction mechanism of H.264/ACV codecs, making it robust to content dynamics.

To further investigate the generalization ability of the bitrate estimator, we additionally train and evaluate its performance by varying fps and segment duration and show the results in Table 1. As expected, our method maintains its performance in different encoding settings. Note that even when the segment duration is doubled, which means more scene changes are involved, the performance of our method does not change much. This result is aligned with our observation in §3.2 that our network design is sufficient to handle the bounded motion distance.

4.2.4 Computation Overhead on the IoT Device. This section evaluates our IoT device's computation overhead *Penance*. We measure the runtime and FLOPs of the two core parts in *Penance*, i.e., bitrate estimator and CRL adapter, on 1s-15 fps segments (the input data shape is 15x640x360). As illustrated in Table 2, *Penance* takes $\sim$ 50 M FLOPs, and the total runtime is below 300 ms. Considering that *Penance* runs once for each 1s segment, it is able to be deployed on general IoT devices.

5 RELATED WORKS

Video Semantic Segmentation: The most straightforward method for VSS is to apply image semantic segmentation models to each frame of the videos, as image semantic segmentation [44, 46, 54, 55] does. To fully exploit the relationship between frames, another stream of works uses optical flow or other motion cues to propagate information between frames to improve accuracy [10, 16, 23, 35] or inference speed [20, 26, 29, 56]. *Penance* is orthogonal to these works as it is agnostic to the details of the VSS model. One can easily conjunct *Penance* with any VSS model and use it as a plug-in to further reduce the inference cost.

Cost Reduction by Prediction Reusing: Much effort has gone into reducing the computation cost of video analytics on edge servers. One stream of works aims to filter frames without harming the overall accuracy of vision tasks. For instance, FilterForward [8] and Noscope [24] adopt cheap neural detectors to determine whether the frame should be offloaded. Reducto [25] leverages low-level image features to enable on-device filtering. Infi [48] proposes an end-to-end learnable filtering framework. Glimpse [12] propagates the cached bounding box on-device to future frames with optical flow. FoggyCache [18] exploits cross-device data similarity between IoT devices to minimize redundant computation. However, none of these works investigates pixel-level labeling tasks like VSS for severe accuracy degradation [51]. While the recent work EdgeIS [51] reused the instance segmentation results with contour calibration, it is optimized for Mask-RCNN and can only trace stationary objects. Instead, Penance is a general edge-assisted VSS framework agnostic to the underlying VSS model.

Joint Data and Model Adaptation: In the realm of image classification and object detection, a few works have been proposed to balance between data versions to minimize the inference costs while satisfying accuracy and bandwidth constraints. For instance, JCAB [43, 53] jointly optimizes bandwidth allocation and object detection model selection. It models the analytic accuracy as a periodically updated function of frame resolution and sampling rate. A^2 [22] minimizes the inference cost while meeting accuracy and latency requirements with an offline-generated accuracy map. Unfortunately, these works adopt fixed or heuristic accuracy functions specially designed for object detection or image classification, which can not be applied to handle the accuracy fluctuation of VSS models.

6 CONCLUSION

This paper presents *Penance*, a lightweight DRL-based low-cost edge-assisted VSS framework that adapts segment-level configurations to minimize edge inference cost while satisfying the accuracy and bandwidth constraints by leveraging output softmax probabilities and H.264/ACV encoding schemes. Experiment results showed the superiority of *Penance* over baseline methods.

7 ACKNOWLEDGMENTS

This work was supported in part by the Key R & D Program of Hubei Province of China under Grant No. 2021EHB002, the National Science Foundation of China with Grant 62071194, Knowledge Innovation Program of Wuhan-Shuguang, the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 101022280, Horizon Europe MSCA programme under grant agreement number 101086228, EPSRC with RC Grant reference EP/Y027787/1, and Project funded by China Postdoctoral Science Foundation 2023M731196.

REFERENCES

[1] [n. d.]. Video Analytics Market Size, Growth | Global Report [2022-2029]. https://www.fortunebusinessinsights.com/industry-reports/video-analytics-market-101114

[2] 2023. encoder/me.c · master · VideoLAN / x264 · GitLab. https://code.videolan.org/videolan/x264/-/blob/master/encoder/me.c

[3] Anil Chandra, Naidu Matcha. [n. d.]. A 2021 guide to Semantic Segmentation. https://nanonets.com/blog/semantic-image-segmentation-2020/

[4] Bhakti Baheti, Shubham Innani, Suhas Gajre, and Sanjay Talbar. 2020. Eff-UNet: A Novel Architecture for Semantic Segmentation in Unstructured Environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

[5] Saheba Bhatnagar, Laurence Gill, and Bidisha Ghosh. 2020. Drone Image Segmentation Using Machine and Deep Learning for Mapping Raised Bog Vegetation Communities. *Remote Sensing* 12, 16 (Jan. 2020), 2602. https://doi.org/10.3390/rs12162602 Number: 16 Publisher: Multidisciplinary Digital Publishing Institute.

[6] Dan A Calian, Daniel J Mankowitz, Tom Zahavy, Zhongwen Xu, Junhyuk Oh, Nir Levine, and Timothy Mann. 2020. Balancing constraints and rewards with meta-gradient d4pg. *arXiv preprint arXiv:2010.06324* (2020).

[7] Yigit Baran Can, Alexander Liniger, Danda Pani Paudel, and Luc Van Gool. 2021. Structured Bird's-Eye-View Traffic Scene Understanding From Onboard Images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 15661–15670.

[8] Christopher Canel, Thomas Kim, Giulio Zhou, Conglong Li, Hyeontaek Lim, David G Andersen, Michael Kaminsky, and Subramanya R Dulloor. 2019. Scaling Video Analytics on Constrained Edge Nodes. In *SysML*. 12.

[9] Anirudh S. Chakravarthy, Soumendu Sinha, Pratik Narang, Murari Mandal, Vinay Chamola, and F. Richard Yu. 2022. DroneSegNet: Robust Aerial Semantic Segmentation for UAV-Based IoT Applications. *IEEE Transactions on Vehicular Technology* 71, 4 (April 2022), 4277–4286. https://doi.org/10.1109/TVT.2022.3144358 Conference Name: IEEE Transactions on Vehicular Technology.

[10] Siddhartha Chandra, Camille Couprie, and Iasonas Kokkinos. 2018. Deep spatio-temporal random fields for efficient video segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8915–8924.

[11] Bike Chen, Chen Gong, and Jian Yang. 2019. Importance-Aware Semantic Segmentation for Autonomous Vehicles. *IEEE Transactions on Intelligent Transportation Systems* 20, 1 (Jan. 2019), 137–148. https://doi.org/10.1109/TITS.2018.2801309 Conference Name: IEEE Transactions on Intelligent Transportation Systems.

[12] Tiffany Yu-Han Chen, Lenin Ravindranath, Shuo Deng, Paramvir Bahl, and Hari Balakrishnan. 2015. Glimpse: Continuous, Real-Time Object Recognition on Mobile Devices. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. ACM, Seoul South Korea, 155–168. https://doi.org/10.1145/2809695.2809711

[13] Konstantin Ditschuneit and Johannes S. Otterbach. 2022. Auto-Compressing Subset Pruning for Semantic Image Segmentation. https://doi.org/10.48550/arXiv.2201.11103 arXiv:2201.11103 [cs, stat].

[14] Kuntai Du, Ahsan Pervaiz, Xin Yuan, Aakanksha Chowdhery, Qizheng Zhang, Henry Hoffmann, and Junchen Jiang. 2020. Server-Driven Video Streaming for Deep Learning Inference. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*. ACM, Virtual Event USA, 557–570. https://doi.org/10.1145/3387514.3405887

[15] Di Feng, Christian Haase-Schütz, Lars Rosenbaum, Heinz Hertlein, Claudius Gläser, Fabian Timm, Werner Wiesbeck, and Klaus Dietmayer. 2021. Deep Multi-Modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges. *IEEE Transactions on Intelligent Transportation Systems* 22, 3 (March 2021), 1341–1360. https://doi.org/10.1109/TITS.2020.2972974 Conference Name: IEEE Transactions on Intelligent Transportation Systems.

[16] Raghudeep Gadde, Varun Jampani, and Peter V Gehler. 2017. Semantic video cnns through representation warping. In *Proceedings of the IEEE International Conference on Computer Vision*. 4453–4462.

[17] José Francisco Guerrero Tello, Mauro Coltelli, Maria Marsella, Angela Celauro, and José Antonio Palenzuela Baena. 2022. Convolutional Neural Network Algorithms for Semantic Segmentation of Volcanic Ash Plumes Using Visible Camera Imagery. *Remote Sensing* 14, 18 (Jan. 2022), 4477. https://doi.org/10.3390/rs14184477 Number: 18 Publisher: Multidisciplinary Digital Publishing Institute.

[18] Peizhen Guo, Bo Hu, Rui Li, and Wenjun Hu. 2018. FoggyCache: Cross-Device Approximate Computation Reuse. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking (MobiCom '18)*. Association for Computing Machinery, New York, NY, USA, 19–34. https://doi.org/10.1145/3241539.3241557

[19] Dan Hendrycks and Kevin Gimpel. 2018. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. https://doi.org/10.48550/arXiv.1610.02136 arXiv:1610.02136 [cs].

[20] Ping Hu, Fabian Caba, Oliver Wang, Zhe Lin, Stan Sclaroff, and Federico Perazzi. 2020. Temporally distributed networks for fast video semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8818–8827.

[21] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodik, Siddhartha Sen, and Ion Stoica. 2018. Chameleon: scalable adaptation of video analytics. In *SIGCOM (SIGCOMM '18)*. Association for Computing Machinery, New York, NY, USA, 253–266. https://doi.org/10.1145/3230543.3230574

[22] Jingyan Jiang, Ziyue Luo, Chenghao Hu, Zhaoliang He, Zhi Wang, Shutao Xia, and Chuan Wu. 2021. Joint Model and Data Adaptation for Cloud Inference Serving. In *2021 IEEE Real-Time Systems Symposium (RTSS)*. 279–289. https://doi.org/10.1109/RTSS52674.2021.00034 ISSN: 2576-3172.

[23] Xiaojie Jin, Xin Li, Huaxin Xiao, Xiaohui Shen, Zhe Lin, Jimei Yang, Yunpeng Chen, Jian Dong, Luoqi Liu, Zequn Jie, et al. 2017. Video scene parsing with predictive feature learning. In *Proceedings of the IEEE International Conference on Computer Vision*. 5580–5588.

[24] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. 2017. NoScope: optimizing neural network queries over video at scale. *Proceedings of the VLDB Endowment* 10, 11 (Aug. 2017), 1586–1597. https://doi.org/10.14778/3137628.3137664

[25] Yuanqi Li, Arthi Padmanabhan, Pengzhan Zhao, Yufei Wang, Guoqing Harry Xu, and Ravi Netravali. 2020. Reducto: On-Camera Filtering for Resource-Efficient Real-Time Video Analytics. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication (SIGCOMM '20)*. Association for Computing Machinery, New York, NY, USA, 359–376. https://doi.org/10.1145/3387514.3405874

[26] Yule Li, Jianping Shi, and Dahua Lin. 2018. Low-latency video semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5997–6005.

[27] Shengzhong Liu, Tianshi Wang, Jinyang Li, Dachun Sun, Mani Srivastava, and Tarek Abdelzaher. 2022. AdaMask: Enabling Machine-Centric Video Streaming with Adaptive Frame Masking for DNN Inference Offloading. In *Proceedings of the 30th ACM International Conference on Multimedia (MM '22)*. Association for Computing Machinery, New York, NY, USA, 3035–3044. https://doi.org/10.1145/3503161.3548033

[28] Raspberry Pi Ltd. [n. d.]. Raspberry Pi 4 Model B specifications. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/

[29] Behrooz Mahasseni, Sinisa Todorovic, and Alan Fern. 2017. Budget-aware deep semantic video segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1029–1038.

[30] Amol Mandhane, Anton Zhernov, Maribeth Rauh, Chenjie Gu, Miaosen Wang, Flora Xue, Wendy Shang, Derek Pang, Rene Claus, Ching-Han Chiang, Cheng Chen, Jingning Han, Angie Chen, Daniel J. Mankowitz, Jackson Broshear, Julian Schrittwieser, Thomas Hubert, Oriol Vinyals, and Timothy Mann. 2022. MuZero with Self-competition for Rate Control in VP9 Video Compression. https://doi.org/10.48550/arXiv.2202.06626 arXiv:2202.06626 [cs, eess].

[31] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*. PMLR, 1928–1937.

[32] Yujian Mo, Yan Wu, Xinneng Yang, Feilin Liu, and Yujun Liao. 2022. Review the state-of-the-art technologies of semantic segmentation based on deep learning. *Neurocomputing* 493 (July 2022), 626–646. https://doi.org/10.1016/j.neucom.2022.01.005

[33] Nur Atirah Muhadi, Ahmad Fikri Abdullah, Siti Khairunniza Bejo, Muhammad Razif Mahadi, and Ana Mijic. 2021. Deep learning semantic segmentation for water level estimation using surveillance camera. *Applied Sciences* 11, 20 (2021), 9691.

[34] taslim murad, Anh Nguyen, and Zhisheng Yan. 2022. DAO: Dynamic Adaptive Offloading for Video Analytics. In *Proceedings of the 30th ACM International Conference on Multimedia (MM '22)*. Association for Computing Machinery, New York, NY, USA, 3017–3025. https://doi.org/10.1145/3503161.3548249

[35] David Nilsson and Cristian Sminchisescu. 2018. Semantic video segmentation by gated recurrent flow propagation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 6819–6828.

[36] Santiago Paternain, Luiz Chamon, Miguel Calvo-Fullana, and Alejandro Ribeiro. 2019. Constrained reinforcement learning has zero duality gap. *Advances in Neural Information Processing Systems* 32 (2019).

[37] Sébastien Piérard, Anthony Cioppa, Anaïs Halin, Renaud Vandeghen, Maxime Zanella, Benoît Macq, Saïd Mahmoudi, and Marc Van Droogenbroeck. 2023. Mixture Domain Adaptation To Improve Semantic Segmentation in Real-World Surveillance. 22–31. https://openaccess.thecvf.com/content/WACV2023W/RWS/html/Pierard_Mixture_Domain_Adaptation_To_Improve_Semantic_Segmentation_in_Real-World_Surveillance_WACVW_2023_paper.html

[38] Joseph Redmon and Ali Farhadi. 2018. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767* (2018).

[39] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4510–4520.

[40] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. In *arXiv:1707.06347 [cs]*. http://arxiv.org/abs/1707.06347 arXiv: 1707.06347.

[41] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[42] Suramya Tomar. 2006. Converting video formats with FFmpeg. *Linux Journal* 2006, 146 (2006), 10.

[43] Can Wang, Sheng Zhang, Yu Chen, Zhuzhong Qian, Jie Wu, and Mingjun Xiao. 2020. Joint Configuration Adaptation and Bandwidth Allocation for Edge-based Real-time Video Analytics. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*. 257–266. https://doi.org/10.1109/INFOCOM41043.2020.9155524 ISSN: 2641-9874.

[44] Haochen Wang, Xiaolong Jiang, Haibing Ren, Yao Hu, and Song Bai. 2021. SwiftNet: Real-time Video Object Segmentation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Nashville, TN, USA, 1296–1305. https://doi.org/10.1109/CVPR46437.2021.00135

[45] Xuedou Xiao, Juecheng Zhang, Wei Wang, Jianhua He, and Qian Zhang. 2022. DNN-Driven Compressive Offloading for Edge-Assisted Semantic Video Segmentation. In *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*. 10.

[46] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. 2018. BiSeNet: Bilateral Segmentation Network for Real-time Semantic Segmentation. 325–341. https://openaccess.thecvf.com/content_ECCV_2018/html/Changqian_Yu_BiSeNet_Bilateral_Segmentation_ECCV_2018_paper.html

[47] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. 2020. BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[48] Mu Yuan, Lan Zhang, Fengxiang He, Xueting Tong, and Xiang-Yang Li. 2022. InFi: end-to-end learnable input filter for resource-efficient mobile-centric inference. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking (MobiCom '22)*. Association for Computing Machinery, New York, NY, USA, 228–241. https://doi.org/10.1145/3495243.3517016

[49] Ben Zhang, Xin Jin, Sylvia Ratnasamy, John Wawrzynek, and Edward A. Lee. 2018. AWStream: adaptive wide-area streaming analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '18)*. Association for Computing Machinery, New York, NY, USA, 236–252. https://doi.org/10.1145/3230543.3230554

[50] Eric Zhang. 2023. Fast Semantic Segmentation. https://github.com/ekzhang/fastseg original-date: 2020-07-22T22:11:27Z.

[51] Jialin Zhang, Xiang Huang, Jingao Xu, Yue Wu, Qiang Ma, Xin Miao, Li Zhang, Pengpeng Chen, and Zheng Yang. 2022. Edge Assisted Real-time Instance Segmentation on Mobile Devices. In *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*. 537–547. https://doi.org/10.1109/ICDCS54860.2022.00058 ISSN: 2575-8411.

[52] Miao Zhang, Fangxin Wang, and Jiangchuan Liu. 2022. CASVA: Configuration-Adaptive Streaming for Live Video Analytics. In *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*. IEEE, London, United Kingdom, 2168–2177. https://doi.org/10.1109/INFOCOM48880.2022.9796875

[53] Sheng Zhang, Can Wang, Yibo Jin, Jie Wu, Zhuzhong Qian, Mingjun Xiao, and Sanglu Lu. 2022. Adaptive Configuration Selection and Bandwidth Allocation for Edge-Based Video Analytics. *IEEE/ACM Transactions on Networking* 30, 1 (Feb. 2022), 285–298. https://doi.org/10.1109/TNET.2021.3106937 Conference Name: IEEE/ACM Transactions on Networking.

[54] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. 2018. ICNet for Real-Time Semantic Segmentation on High-Resolution Images. https://doi.org/10.48550/arXiv.1704.08545 arXiv:1704.08545 [cs].

[55] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. 2017. Pyramid Scene Parsing Network. https://doi.org/10.48550/arXiv.1612.01105 arXiv:1612.01105 [cs].

[56] Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, and Yichen Wei. 2017. Deep feature flow for video recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2349–2358.