Enabling Edge-Cloud Video Analytics for Robotics Applications

Yiding Wang[1], Weiyan Wang[1], Duowen Liu[1], Xin Jin[2], Junchen Jiang[3], Kai Chen[1,4]
[1]iSING Lab, Hong Kong University of Science and Technology
[2]Peking University [3]University of Chicago [4]Pengcheng Lab

***Abstract*—Emerging deep learning-based video analytics tasks demand computation-intensive neural networks and powerful computing resources on the cloud to achieve high accuracy. Due to the latency requirement and limited network bandwidth, edge-cloud systems adaptively compress the data to strike a balance between overall analytics accuracy and bandwidth consumption. However, the degraded data leads to another issue of poor *tail accuracy*, which means the extremely low accuracy of a few semantic classes and video frames. Autonomous robotics applications especially value the tail accuracy performance but suffer using the prior edge-cloud systems.**

We present Runespoor, an edge-cloud video analytics system to manage the tail accuracy and enable emerging robotics applications. We train and deploy a super-resolution model tailored for the tail accuracy of analytics tasks on the server to significantly improves the performance on hard-to-detect classes and sophisticated frames. During online operation, we use an adaptive data rate controller to further improve the tail performance by instantly adjusting the data rate policy according to the video content. Our evaluation shows that Runespoor improves class-wise tail accuracy by up to 300%, frame-wise 90%/99% tail accuracy by up to 22%/54%, and greatly improves the overall accuracy and bandwidth trade-off.

I. Introduction

Deep learning-based video analytics applications are flourishing and powering a growing number of traditionally challenging applications for scene understanding. Such applications adopt computer vision (CV) techniques including multiple object detection [1], semantic segmentation [2], instance segmentation [3] and panoptic segmentation [4]. To obtain adequate inference accuracy, these tasks often require (i) computation-intensive deep learning (DL) models, (ii) powerful computation resources, and (iii) high-fidelity data. Also, to enable responsive and high-precision robotics applications such as autonomous vehicles and unmanned aerial vehicles (UAVs), video data should have both a high frame rate and high resolution.

Let us start with an example of robotics applications that can benefit from edge-cloud computing: autonomous delivery vehicles [5]. They are small-sized electric vehicles and designed to be deployed on a large scale. But during the COVID-19 lockdown when they are needed the most, "the technology is not ready at scale to deploy", said the president of a leading startup, and the service fee could be higher than delivery riders [6]. This is because unlike high-end autonomous cars that install expensive hardware to run complex DL inference locally [7], they are budget and battery constrained. They run at slower speeds on sidewalks or bike lanes, and are monitored remotely by operators via cellular video streams and intervened if necessary [8], thus do not have a latency requirement as stringent as high-end autonomous cars. Nevertheless, they use the same fundamental technologies: they heavily utilize cameras and deep neural networks to understand the surroundings for planning and control [9]. Currently, autonomous delivery vehicles run DL video inference locally as full-size cars, which could raise some concerns for the future *performance* and *scalability*: as they perform more challenging tasks, DL computation also increases, so does the requirement for computation resources; in the meantime, they still need to keep the compactness and cost under budget to scale.

Off-loading the heavy DL inference tasks to the cloud is a promising solution: it can significantly reduce the hardware requirements of edge devices, enable these applications to scale, and reuse the bandwidth for remote monitoring. The cloud (datacenters and edge clusters) has abundant compute and network resources to provide high accuracy for fast video analytics tasks [10]–[14]. However, the downside of cloud is also apparent: the limited bandwidth between the cloud and the edge could cause long transmission delay [15] when streaming high-quality video or inferior inference performance with low resolution or frame rate. It is critical for autonomous delivery vehicles to make decisions fluently and identify obstacles on the road correctly. There is a trade-off between overall accuracy and bandwidth consumption [15]–[17].

However, we find that managing this trade-off is not sufficient to enable edge-cloud robotics applications since it tends to cause poor *tail accuracy*. Although tuning data knobs (e.g., reducing video resolution) can balance bandwidth consumption and overall accuracy to some extent, it will also cause poor *class-wise tail accuracy* (i.e., low accuracy of specific semantic classes like traffic lights and motorcycles) and *frame-wise tail accuracy* (i.e., a few frames with extremely low accuracy over a period of time). This is because video data degradation will lead to the loss of detailed information, thus hurt the accuracy of these hard-to-detect small regions (e.g., traffic signs and riders rather than sky and buildings) and sophisticated frames [18], [19]. Issues caused by poor tail accuracy, including the mislabeling of specific classes and short periods of low accuracy inference, would hurt real-world performance and operations [20], [21]. However, in a conventional bandwidth-accuracy trade-off, the overall accuracy cannot reflect the low tail accuracy, because tail accuracy is covered by well-classified classes which are robust to data degradation (class-wise) and by the long periods of good operations (frame-wise) (§ II-B).

Motivated by robotics applications, we argue that for such edge-cloud video analytics systems, the real challenge is the *three-way trade-off* between bandwidth consumption, overall accuracy, and *tail accuracy*. Runespoor addresses this challenge with: (i) analytics-aware super-resolution (ASR) and (ii) content-aware adaptive controller (CAC).

Analytics-aware super-resolution extends super-resolution (SR), which is an effective DL technique that learns a mapping from low-resolution frames to high-resolution frames. We use the DL models for CV tasks to train ASR to reconstruct high-resolution frames tailored for the tail accuracy performance of video analytics tasks with augmented details from compressed data. ASR explicitly considers the DL inference performance in tail-related regions as the learning signals in addition to the conventional reconstruction similarity target (e.g., PSNR [22] and SSIM [23]) of SR. In this way, ASR significantly improves the accuracy of detailed regions, thus improves the accuracy of hard-to-detect classes and sophisticated frames (§ III-A).

Content-aware adaptive controller specifically improves the inference accuracy of sophisticated frames that heavily relies on detailed information by sending specialized higher-resolution frames. In edge-cloud systems, reducing data rate regarding the bandwidth constraints leads to the loss of detailed information. Moreover, the fast-changing scenes in robotics applications make it challenging to detect and fix tail accuracy during the online operation. Based on our tail-aware offline profiling, CAC learns to detect the frames that lead to low tail accuracy using the inference results of CV tasks and instantly decides the data configuration for subsequent frames to improve tail accuracy without extra DL computation or network overhead (§ III-B).

We implement and evaluate Runespoor on two CV tasks for modern robotics applications: (i) road-driving semantic segmentation and (ii) drone-view object detection. Our evaluation shows that Runespoor significantly outperforms prior work on tail accuracy. Runespoor improves frame-wise 90% and 99% accuracy by 18%-22% and 35%-54% for semantic segmentation, respectively. Runespoor improves class-wise 50% to 100% accuracy averagely by 0.9%-79.4% for semantic segmentation, and 14%-300% for object detection. Runespoor with CAC is an efficient online end-to-end system that adapts to real-world changing scenes. Runespoor also improves overall accuracy and saves bandwidth consumption.

Our contributions are: (i) exposing and defining the important tail accuracy problem in edge-cloud video analytics; (ii) analytics-aware super-resolution (ASR) that fixes tail accuracy by focusing on detailed information reconstruction; (iii) content-aware adaptive controller (CAC) that adapts to fast-changing scenes with DL outputs in an end-to-end system.

II. Motivation

Runespoor is motivated by video analytics tasks for emerging robotics CV applications, e.g., autonomous delivery robots and UAVs. They use fresh videos and are deployed on a large scale in complex scenes. They are equipped with high-definition cameras and require scene understanding capabilities for planning and control. They require powerful DL models, computation resources, and high-quality data [24] to ensure high inference accuracy and fast reaction. Due to limited computation resources at edge, many applications send videos to the cloud to run heavy DL inference tasks [10], [15], [16]. State-of-the-art video analytics systems work efficiently on general tasks, but are limited for our target applications (§ II-D).

We find that although current video degradation techniques in edge-cloud analytics systems (e.g., AWStream [15] and NAS [25]) can reduce the bandwidth consumption and maintain a relatively high overall inference accuracy, e.g., mIoU (mean of intersection over union) for semantic segmentation and mAP (mean average precision) for object detection, they cannot fix poor *tail accuracy*, which is caused by data degradation and the requirement of detailed information in CV tasks (§ II-B).

A. What is tail accuracy?

We conduct preliminary evaluations on data reconstruction methods in previous edge-cloud video analytics systems and find that the tail accuracy has two forms: (i) *class-wise* tail accuracy and (ii) *frame-wise* tail accuracy.

Class-wise tail accuracy refers to the huge accuracy loss of some semantic classes in the scene understanding DL inference tasks (e.g., segmentation and detection) on degraded data.

Let us start with an example: under bandwidth constraints, the edge device (e.g., an autonomous delivery robot) sends downsampled 512×256 frames (from Cityscapes [24] validation set) to the cloud for semantic segmentation inference. Before inference, to fit the input resolution of the DL model, frames are upsampled to the original 2048×1024 resolution by (i) Bilinear algorithm and (ii) the state-of-the-art SR model [26]. Bilinear algorithm is the default upsampling method in major ML frameworks (e.g., TensorFlow and PyTorch). SR proves useful in video systems [25], [27], [28]. In Figure 1, the class-wise accuracy is normalized to the fraction of the inference accuracy on original high-resolution frames, because we want to compare the accuracy loss caused by data degradation.

As expected in Figure 1, upsampling the compressed frames achieves higher inference accuracy with the same received data and bandwidth consumption. What is more, there exists significant class-wise accuracy inequality on all three series. The accuracies of classes like road, wall, sky, and buildings are higher than the overall accuracy (mIoU, three red lines) and close to 1, which means almost no loss on degraded data. However, other classes (e.g., motorcycles, bicycles, riders, traffic lights/signs, and buses) suffer from much more accuracy loss. Overall accuracy (e.g., mIoU and mAP) does not reflect the huge accuracy loss of those classes (tail), because most regions and classes are robust to the video resolution degradation (e.g., sky, road, cars, and buildings).

Frame-wise tail accuracy refers to the extremely low tail accuracy over a temporal series of video frames in inference tasks. Emerging applications such as autonomous vehicles require fluent decision-making. Every frame matters for such applications, but some frames suffer from more accuracy loss on degraded data. Figure 2 uses the same setting as Figure 1. It

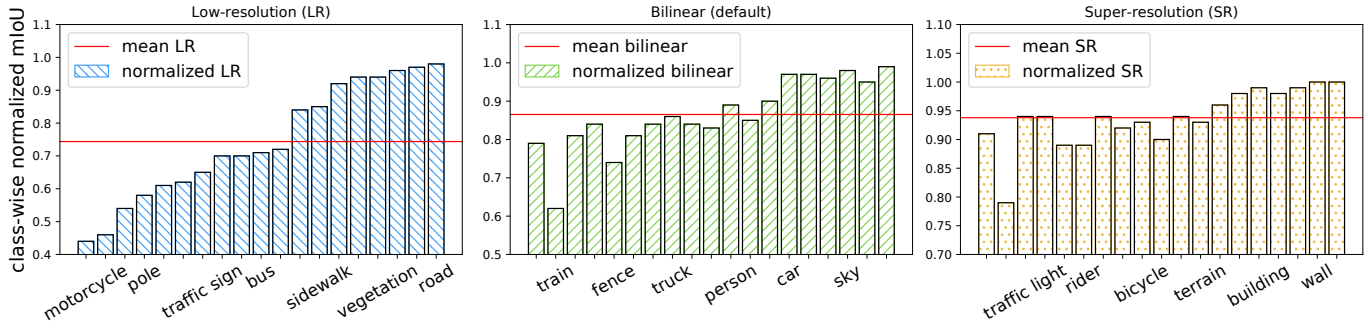


Fig. 1: Normalized semantic segmentation class-wise accuracy after compression. The normalization baseline is the accuracy on original data. Bilinear (mid) and super-resolution (right) still obtain degraded tail accuracy, although they improve overall accuracy. Subfigures have different Y-axises. The 19 X-axis class labels are printed in Round-Robin for space limitation. (Complete ordered labels please see Figure 7.)

shows the accuracy distribution of frames that are processed by Bilinear algorithm and the standard SR model. The accuracy of the hardest frames (90% and 99% accuracy) can be 19% to 37% worse than the mean accuracy of frames. The accuracy distribution indicates that during a period of operation, some frames can suffer from significantly low accuracy, which is not pleasant for applications. However, the overall accuracy of all frames over a long period of time cannot capture tail accuracy.

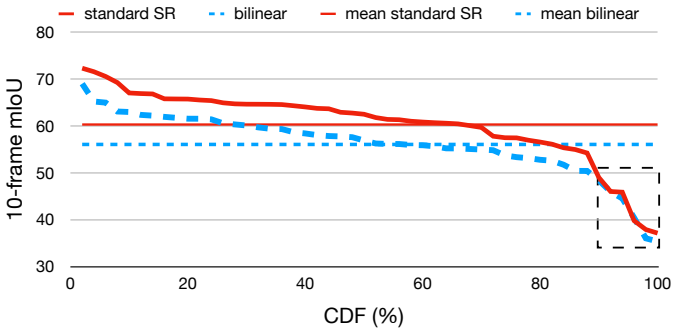


Fig. 2: Frame-wise accuracy distribution (averaged on every 10 frames). Using SR does improve the overall accuracy over Bilinear (two horizontal lines), but they both suffer from poor *tail accuracy* (the highlighted 90+ percentiles).

B. Why does tail accuracy exist?

The fundamental cause of the poor tail accuracy is that the detailed information, including small regions/objects and boundaries, is more sensitive to the degraded input data (downsampled frames) in DL inference tasks. The smaller or the lower resolution a region is, the harder it is to detect its semantic label by a neural network [29], as shown in the evaluation of class-wise inference accuracy and region size in Table I. This is why scene understanding applications require high-resolution videos to ensure high accuracy. This has been well discussed in the CV community [18], [19], but ignored in designing edge-cloud systems for video analytics.

As shown in Figure 1, upsampling low-resolution frames does improve the overall accuracy, and even achieves 94% of the best mIoU results with SR. However, the accuracy of those tail classes are still not satisfactory because they are generally small and sensitive to detailed quality loss.

The same reason also causes frame-wise tail accuracy. During a time series of video frames of moving robotics applications, the content of the video scenes constantly shifts, unlike fixed traffic cameras [16]. On the frames that are sensitive to detailed information, the inference accuracy of those classes will drop drastically on degraded data, and the frame-wise accuracy will drop accordingly. Naturally, the tail frames occur when the scene changes. The worst frames are not frequent, but they lead to poor tail accuracy performance.

C. Why does tail accuracy matter?

The tail accuracy issue would result in two problems in real-world scenarios: (i) Class-wise tail accuracy would harm the accuracy of some important classes, which are critical to the application performance. (ii) Frame-wise tail accuracy would hurt fluent decision-making and impair operations.

Class-wise tail accuracy is an important issue because those most affected classes are usually critical to robotics applications. The importance of an object often appears in small sizes in videos in practice [20]. For example, autonomous vehicles should pay more attention to the regions that are closely related to driving safety than those that are not crucial for vehicle operation. In other words, the system should focus on major obstacles/risks (e.g., pedestrians, riders, vehicles, and bicycles) and traffic information (e.g., traffic signs/lights). It does not need to pay the same attention to processing less critical regions such as sky, vegetation, and buildings. It would be very dangerous to mis-detect a person or rider in the scene. In Figure 1, it is clear that some of the tail classes are more important for applications.

Frame-wise tail accuracy in a time series of frames is critical to robotics applications for two reasons. First, the operation of autonomous systems relies on the DL inference results of CV tasks [9]. Robotics applications require correct DL inference to support the following decision-making. It is important to avoid extremely low tail accuracy on degraded data in edge-cloud systems. Second, the impact of tail accuracy would be magnified and significant for the overall service quality during a long time of operation and under a large-scale deployment.

D. Missing pieces in prior work

We argue that the real challenge for such edge-cloud video analytics systems is to handle the three-way trade-off between

bandwidth consumption, overall accuracy, and tail accuracy. Let us review how prior work handles this challenge.

1) Skipping non-relevant/stale frames at edge can save bandwidth for fixed cameras with many similar frames [16], [17], [30]. However, robotics applications require a fast reaction to the constantly changing environment [7]. We analyze our video datasets and find that 88%-94% of frames are essential to applications: they contain vehicles, people, traffic signs and actively change. So the room for saving bandwidth by skipping frames is limited here.
2) Splitting the CV model to both edge and cloud and only sending the intermediate CNN output to the cloud [31] can reduce bandwidth consumption, compared with sending raw videos. However, robotics applications require complex CNN models on powerful hardware to provide high inference accuracy. Constrained edge devices can only run cheaper models to extract features, which may not achieve the best accuracy. For example, for semantic segmentation, using the edge-friendly MobileNetV3_Small [32] as backbone takes 1% computation of the heavy Xception-71 [33], but loses 14% accuracy [34], which is crucial for demanding applications. Also, the rapid development of CNN often changes backbone architectures, which will break the edge-cloud splitting configuration [35]. We do not dispute the work on the edge and DL accelerators [36], but present a new design considering these limitations.
3) The systems using compression/downsampling [15] and quality recovery with SR [25] reduce bandwidth consumption with acceptable overall accuracy, and utilize the cloud computation resource. However, such data degradation methods lead to the *tail accuracy* issue.

III. Design

Runespoor is an edge-cloud video analytics framework especially for robotics applications. We propose two major components: *analytics-aware SR* (ASR) and *content-aware adaptive controller* (CAC). Figure 3 illustrates the workflow of Runespoor. The edge device adaptively compresses the video captured by the high-definition camera with CAC and streams it to the cloud server. On the server, it reconstructs the received data with ASR, then runs DL inference for CV tasks. CAC uses the DL inference results for the edge-side data rate control.

A. Analytics-aware super-resolution

Inspired by applying SR in video systems [25], [27], [28], we find that SR can be an effective method to strike a balance between bandwidth consumption and overall inference accuracy, however the tail accuracy issue still remains (§ II-D). An SR model trained on the same datasets as the vision analytics tasks (NAS [25]) is naturally an enhancement for downsampling-based systems (e.g., AWStream [15]) and a strong baseline.

To address the three-way trade-off between bandwidth consumption, overall accuracy, and tail accuracy, especially the limitations of video compression techniques that lead to the tail accuracy issue (§ II-B), Runespoor uses analytics-aware SR (ASR), which focuses on detailed information reconstruction

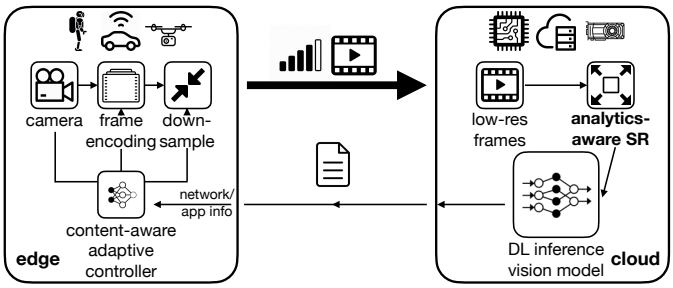


Fig. 3: Runespoor framework overview. The two design components are cloud-side ASR and edge-side CAC.

for analytics tasks. ASR improves the DL inference accuracy on reconstructed frames, especially on the small regions and objects that are quality-sensitive to be correctly detected.

To train ASR, we first train a base SR model using the dataset for CV tasks (e.g., semantic segmentation and object detection), then use the *analytics-aware loss function* to fine-tune the SR model to improve the DL inference accuracy. We use this two-step arrangement instead of training from scratch because of the different nature of SR and DL tasks. First, data labeling is prohibitively labor-intensive [37], thus Cityscapes dataset only labels 1 out of 30 frames. Analytics-aware training requires data annotation for DL tasks, while the base SR training can use much more unlabeled frames besides the training dataset. Second, it allows us to apply different training techniques. For example, SR and DL tasks both use randomly cropping as data augmentation for generalization [38]. We can use small patches (64×64) for the base SR training to reconstruct small details, and large patches (720×720) for analytics-aware fine-tuning to identify regions of any scale to improve the training quality.

The key design of ASR is the *analytics-aware loss function* that maximizes tail accuracy. Equation 1 illustrates its general idea. l_s denotes the structural similarity loss, which measures the frame reconstruction quality and stabilizes loss convergence. NAS [25] only uses l_s to train the standard SR. l_a denotes the task analytics loss, which measures the accuracy of DL inference tasks. hr and lr are the original high-resolution and downsampled low-resolution frames from the same datasets for training video analytics tasks. `asr_model` is the ASR model to train. `inf_model` denotes the video analytics model, which is pre-trained on high-resolution datasets for CV tasks. Runespoor does not require any customization on `inf_model` for generality in real-world deployment. We compute gradients of the loss function on both `asr_model` and `inf_model` whose parameters are frozen. α and β are weight parameters. Through experiments, we recommend a larger α for robust convergence. In this way, we utilize the powerful `inf_model` architecture to make `asr_model` analytics-aware.

$$\begin{aligned}\text{loss} = \alpha{\cdot}l_s(\text{asr\_model}(lr), hr) + \\ \beta{\cdot}l_a(\text{asr\_model}, \text{inf\_model}, lr).\end{aligned} \tag{1}$$

To implement Equation 1, an alternative method is using l_a to minimize the difference between the inference results of hr and reconstructed lr. CloudSeg [27], which is the workshop version

of this work, uses this method. Its rationale is treating HR (and the inference results on HR) as the target, which is intuitively inherited from the traditional SR training. We find this method inefficient, because a well-trained DL model can still make wrong predictions and brings inaccurate supervisory signals to our loss and affects `asr_model`. This method limits the accuracy performance (evaluated in § V-A1), and even makes tail accuracy worse by augmenting false predictions, as we find in evaluations.

Instead of the CloudSeg method, to minimize tail accuracy, we use the classification error of the DL inference tasks on small regions as the analytics loss l_a, as shown in Equation 2 and Figure 4. We bring the label of the analytics tasks to ASR training, which CloudSeg does not utilize. More importantly, l_a (e.g., cross entropy loss for semantic segmentation) in Equation 2 concentrates on the loss signals from small/detailed regions instead of the whole frame to improve tail accuracy by targeting the root cause (§ II-B). We select small or detailed (`dtl.`) regions by the relative size of a region to the frame. The threshold is different for each dataset. As a result, the large and stable areas like the sky are not considered in l_a.

$$\begin{aligned}\text{loss} = \alpha{\cdot}l_s(\text{asr\_model}(lr), hr)+\\ \beta{\cdot}l_a(\text{inf\_model}(\text{asr\_model}(lr))_{\text{dtl.}}, \text{label}_{\text{dtl.}}).\end{aligned} \tag{2}$$

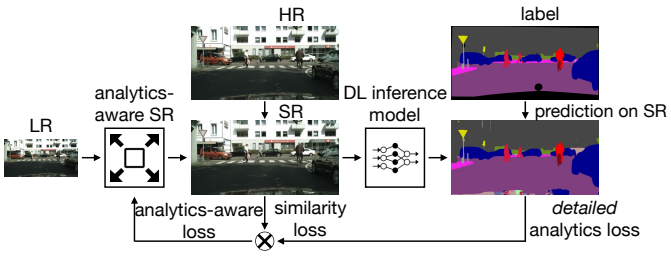


Fig. 4: We use DL inference results of detailed regions as the analytics-aware optimization targets (Equation 2).

B. Content-aware adaptive controller

Towards the bandwidth constraint in the online operation of edge-cloud systems, conventional wisdom [15], [39] suggests three knobs to adjust data rate: resolution, frame rate, and encoding quality and use a data rate controller to adapt to network conditions. For example, AWStream [15] learns the knob configurations for tasks by *offline profiling* and periodically updates the controller to handle *model drift* (less efficiency when the environment changes) by *online profiling*. Its online profiling works as steps 1 to 3 in Figure 5: (i) periodically profiling the current DL inference performance with extra raw data, and (ii) updating the data rate strategy, to (iii) make the corresponding adjustments in video knobs.

However, we find that for video analytics in robotics applications, the conventional data rate controller [15] is incapable of handling the tail accuracy issue. First, the conventional controller learns the data rate policy for a dataset of similar scenes under varying bandwidth constraints. Its design goal is to handle network changes instead of *frequent scene changes*. Thus it does not detect and react to the few frames that lead to low tail accuracy (§ II-B). Second, periodically online profiling (step 1) requires extra bandwidth consumption (to send raw data as a reference) and DL computation (for baseline accuracy). Thus it takes tens of seconds for AWStream [15] to react to significant scene changes only like camera movement. Such a slow process cannot catch "tail frames" in robotics applications because the newly-updated policy has already become stale.

Considering these two issues and our goal, we propose *content-aware adaptive controller* (CAC) to use the video analytics results to instantly detect the frames that could lead to extremely low accuracy on degraded data ("tail frames") and apply specialized video knobs configurations to improve tail accuracy. Figure 5 shows the workflow of CAC. Two key designs of CAC are: (i) *tail frame detection* with scene understanding predictions; and (ii) *tail-aware offline profiling* to obtain tail-specific data rate configurations.

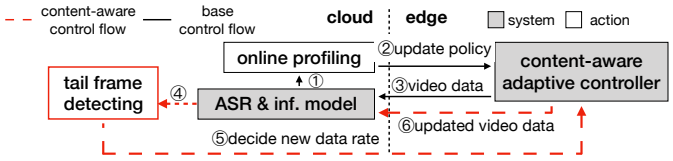


Fig. 5: Online workflow of content-aware adaptive controller (CAC). Steps 4 to 6 are content-aware. Steps 1 to 3 follow AWStream [15].

Tail frame detection requires a *low latency* and a *high recall rate* to catch all potential tail frames in fast changing environments. Lower precision, i.e., mis-detecting simple frames as tail frames, will not hurt the final accuracy because simple frames are quality-robust for DL tasks (§ II-B) under the same bandwidth constraint, as evaluated in § V-C. We use a threshold of the ratio of small regions to detect tail frames: checking the DL inference result (e.g., semantic segmentation) of a frame, if the size ratio of all small regions in the frame is below the threshold, we consider it the tail frame. This is because with fewer small regions, the accuracy of hard-to-detect classes tends to be volatile and easily affected by wrong predictions, so does the frame-wise accuracy. CAC applies specialized data configurations to send the subsequent frames to improve the DL inference accuracy of these tail frames. The small region here is defined the same as in Equation 2.

Finding this threshold is done together with *tail-aware offline profiling*. For generalization, we use a linear regression model to determine the threshold for tail frame detection using the DL profiling results of a dataset. The threshold we obtain for Cityscapes dataset is 1%. We analyze the tail frame detection results on Cityscapes validation dataset in Figure 6. The accuracy (mIoU) is calculated per frame, thus lower than the cumulative overall accuracy. The red crosses depict 95% to 100% tail accuracy. Using the 1% threshold of the ratio of small regions to detect tail frames, the recall rate is over 90%, which means we can catch almost all tail frames. The precision is between 25% and 35% on three scales, but we do not require high precision. By utilizing DL inference results of analytics

tasks, the online computation overhead of CAC is finding small regions by depth-first search on downsampled prediction maps (segmentation) or directly adding small bounding boxes (detection), which takes <1ms (unnoticeable) per frame. The detection is accurate even with low-resolution (8×) frames.

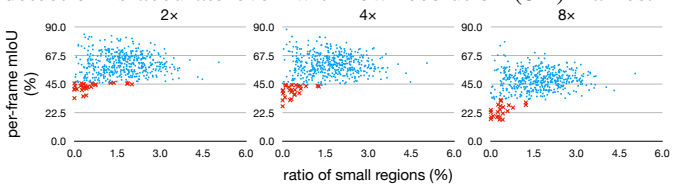


Fig. 6: Using a 1% threshold of small region ratio catches tail frames with a high recall rate. ×8 is the lowest resolution.

Tail-aware offline profiling in CAC extends prior work AWStream [15] to optimize tail accuracy. For a CV task, offline profiling on the training dataset with different knobs (resolutions and encoding qualities) processed by ASR will find the best data rate configurations for high accuracy regarding different bandwidth constraints. We separate offline profiling to find (i) the *tail-specific* knob configurations on frames with 90% to 100% accuracy (tail frames) and (ii) the standard configurations on the rest of the dataset. As discussed before, tail frames need a higher resolution to be correctly detected. The tail-specific configurations we obtain have a higher resolution, lower encoding quality, and the same data size. This is because tuning encoding quality within a range can reduce the data rate without hurting accuracy [15], [39]. This brings a significant tail accuracy improvement and almost no loss for other frames. Offline profiling is a one-time process, which takes tens of minutes for Cityscapes dataset with dozens of configurations.

In summary, CAC in Figure 5 works as follows. At the server, in addition to periodically online profiling as AWStream (steps 1 to 3) for standard performance monitoring, CAC gathers the video analytics results to detect tail frames (step 4). If tail frames are detected, CAC will adaptively apply a specialized data configuration (step 5) learned at the tail-aware offline profiling. At the edge, CAC instantly (∼500ms, § V-C) applies the specialized configurations for the subsequent similar frames (step 6) to improve tail accuracy, compared with ∼10s of AWStream. During the online operation, CAC will adjust data rate configurations according to the video content, video analytics and network performance. CAC requires no extra DL computation and raw data to obtain accuracy baselines online, and has the same offline profiling workload as AWStream.

IV. Implementation

Framework. We implement Runespoor prototype as Figure 3. We build video frames and control messages transmission with ZeroMQ between edge and cloud, and DL inference with PyTorch at the cloud. To achieve real-time video analytics, we use *per-frame inference* [40], and implement CV and encoding operations with OpenCV. Multiple ASR models are loaded via Flask at the cloud for different resolutions of received data. ASR models are not memory-consuming comparing with DL inference models. We build two applications which are widely used in real-world autonomous systems [41]: semantic segmentation and object detection.

Dataset. We implement semantic segmentation on urban road driving dataset Cityscapes [24], and object detection on drone-view dataset VisDrone2019 [42].

Cityscapes dataset consists of 8-bit RGB video frames in 2048×1024 resolution and a 17 fps frame rate, which are captured by the front camera of vehicles. It contains 2975 training frames, 500 validation frames, and more unlabeled video clips which are used in ASR training and CAC evaluation.

VisDrone2019 dataset consists of frames collected by drones, mostly in cities, and the resolution is up to 2000×1500. We use 1000 high-quality frames with a minimum resolution of 1920×1080 to provide high inference accuracy, especially for small objects. They are split into 700 and 300 images for ASR training and object detection evaluation, respectively.

To evaluate DL inference accuracy, we use standard metrics: intersection over union (IoU) for semantic segmentation, and average precision (AP) for object detection.

Training ASR and CAC. We implement ASR with PyTorch based on CARN [26], which is an efficient SR architecture. To train the base SR models (§ III-A), we use the training datasets plus unlabeled frames. The base training uses the default L1 loss function. After the base training, we get a set of SR models as a strong baseline of NAS [25].

We use SwiftNet [2] and YOLOv3 [1] to implement our analytics-aware training for semantic segmentation and object detection, respectively. They both consume high-resolution data to achieve state-of-the-art inference accuracy in real-time (§ V-C). We use the model weights provided by authoritative implementations [43], [43] which are pre-trained on the target datasets. They will not be modified in ASR training.

SwiftNet takes 2048×1024 input, and YOLOv3 takes 832×832, to exploit high-resolution data. To train ASR with the analytics-aware loss function (Equation 2), we set α and β to 0.95/0.05 and 0.9/0.1 through experiments. For semantic segmentation, we use cross entropy as the analytics loss l_a. For object detection, l_a uses cross entropy and focal loss, which calculates the classification and bounding-box location error.

To find the standard and tail-specific knob configurations in CAC, offline profiling includes resolutions from ×2 to ×8, JPEG encoding quality from 40 to 90, and processed by ASR regarding different bandwidth targets. Usually, tail-specific configurations have one level higher of resolution.

V. Evaluation

We show the evaluation results of Runespoor from:

§ V-A Runespoor handles *tail accuracy*. ASR improves 50% to 100% class-wise accuracy by up to 300%, and frame-wise 90% and 99% accuracy by up to 22% and 54%.

§ V-B For the traditional goal, ASR reduces *bandwidth consumption* and also improves *overall inference accuracy*.

§ V-C During *online operation*, CAC instantly reacts to the hard-to-detect and complex frames and further improves *tail accuracy*. The end-to-end Runespoor system is efficient.

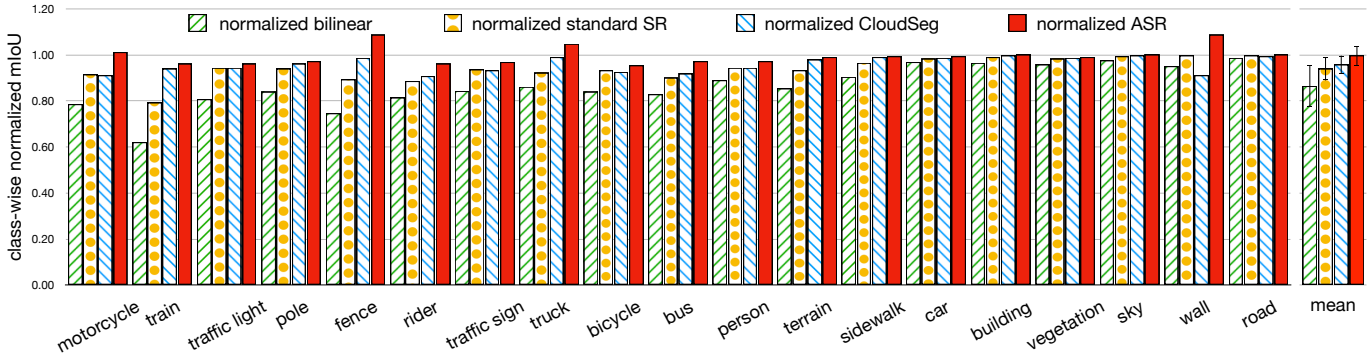


Fig. 7: ASR effectively improves the class-wise tail accuracy, and has the lowest standard deviation with a higher mean accuracy.

A. Tail accuracy improvements

Edge-cloud video analytics systems send compressed data to save bandwidth but lead to low tail accuracy. We compare ASR of Runespoor with three baselines which improve the accuracy of scene understanding DL inference tasks: (i) Bilinear algorithm (used in AWStream [15]) and (ii) the standard SR (used in NAS [25]). (iii) CloudSeg [27] described in § III-A. The standard SR models are well trained and achieve their optimal performance in PSNR [22], which shows the reconstruction performance to keep similarity. As a reference, we show the DL inference results on the original high-resolution data. The original resolution of Cityscapes validation dataset for semantic segmentation is 2048×1024 (HR), and ×2 means 1024×512 and so on. The VisDrone validation dataset has original frames in 1920×1080 and 2000×1500.

1) Class-wise tail accuracy: Figure 7 shows the class-wise DL inference results when receiving 512×256 (×4) frames for semantic segmentation, using three baselines and ASR. The per-class accuracy is normalized to the HR inference accuracy. ASR improves explicitly the accuracy of those semantic classes that perform poorly on degraded data, e.g., motorcycles, riders, buses, and fence, while CloudSeg [27] shows little or even negative improvements. Some classes having normalized accuracies that are higher than 100% is reasonable: training ASR employs architectures of SR plus the segmentation model and improves under-trained classes (which seldom appear in the dataset) in the pre-trained CV model. With Runespoor, all 19 classes achieve higher accuracy than the overall accuracy of other baselines (right). This shows that ASR is effective to fix the poor class-wise tail accuracy issue.

In detail, we compare with prior work on the average of 50% to 100% class-wise tail accuracy, which includes the worst 9 classes ("tail classes") out of 19 in Figure 7. Table I shows the tail accuracy breakdown of semantic segmentation. Comparing with the standard SR (NAS [25]), Runespoor exceeds on the average of 50% to 100% class-wise tail accuracy by 8.4% (from 61.73% to 66.92%), which is larger than the overall accuracy improvement (5.5% from 71.15% to 75.04%). Comparing with Bilinear algorithm (AWStream [15]), Runespoor exceeds on the tail accuracy by 23.7% (from 54.11% to 66.92%), which is also larger than the overall improvement(13.9%).

class IoU(%)	Bilinear	standard SR	CloudSeg	**ASR**	HR	size(%)
train	45.45	58.12	68.95	70.41	70.41	0.11
fence	41.83	50.09	55.49	61.12	56.20	0.82
motorcycle	45.96	53.38	53.22	59.02	58.44	0.08
traffic light	54.54	63.80	63.82	65.01	67.65	0.20
rider	47.34	51.64	52.87	55.96	58.28	0.22
bus	70.69	76.71	78.33	82.66	85.17	0.39
pole	52.73	59.05	60.54	61.24	62.91	1.48
bicycle	63.57	70.78	70.23	72.25	75.77	0.71
traffic sign	64.91	71.96	71.84	74.60	76.95	0.67
tail classes	54.11	61.73	62.72	**66.92**	68.30	4.68
normalized	0.79	0.90	0.92	**0.98**	1.00	in total
overall mIoU(%)	65.87	71.15	72.46	**75.04**	75.35	—

TABLE I: Class-wise tail accuracy breakdown at ×4 of semantic segmentation. ASR largely improves the class-wise tail accuracy performance comparing with prior work.

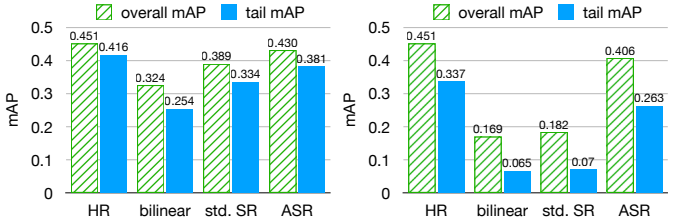
Table II shows the performance of ASR on all resolution scales. Comparing with Bilinear and the standard SR, ASR improves the average of 50% to 100% class-wise tail accuracy by 4.5%/0.9%, 23.7%/8.4%, 58.2%/20.3% and 79.4%/65.9% respectively on ×2, ×4, ×6 and ×8 compressed inputs.

scale	mIoU(%)	Bilinear	standard SR	**ASR**	HR
×2	**tail classes**	64.60	66.91	**67.52**	68.30
1024×512	overall	73.11	74.66	**75.12**	75.35
×4	**tail classes**	54.11	61.73	**66.92**	68.30
512×256	overall	65.87	71.15	**75.04**	75.35
×6	**tail classes**	38.17	54.77	**61.24**	68.30
341×170	overall	53.13	65.14	**70.81**	75.35
×8	**tail classes**	25.36	27.42	**45.50**	68.30
256×128	overall	40.49	45.40	**60.32**	75.35

TABLE II: Overall and class-wise tail accuracy of all scales for semantic segmentation.

We report the class-wise tail accuracy performance of object detection in Figure 8. ASR of Runespoor beats the two baselines at ×2 and ×4 inputs and obtains good performance compared with original data. The 50% to 100% tail classes of the VisDrone dataset include tricycles, bicycles, awning-tricycles, motors, and people, which are all small and hard to detect in the view of drones. ASR improves the class-wise tail accuracy of object detection by 14% to 300%, and overall accuracy by 11% to 140%. Let us take a close look. In this task, ×4 downsampled data hurts the accuracy significantly. For example, accuracy

(mAP) of tricycles detection on ×4 data with the standard SR only achieves 10% normalized accuracy. Impressively, ASR improves it to 91%. Figure 9 shows results on a real complex frame. We can see that the frame reconstruction of ASR which focuses on the detailed information lets the DL inference model detect the compact parked cars (left) and people on the street (right) much more accurately than the standard SR.



(a) ×2 downsampling (b) ×4 downsampling

Fig. 8: Average 50% to 100% class-wise tail accuracy of the object detection task. ASR improves class-wise tail accuracy especially on ×4 downsampled input.

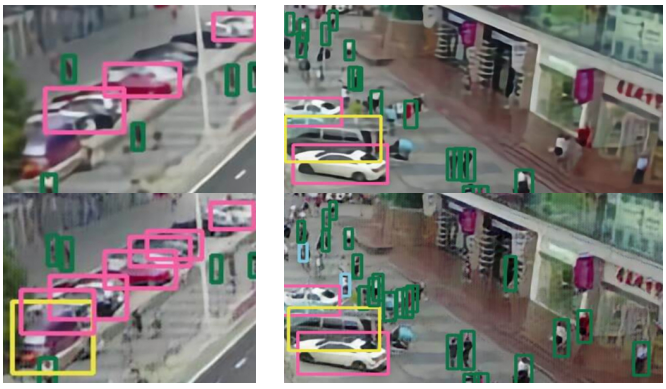


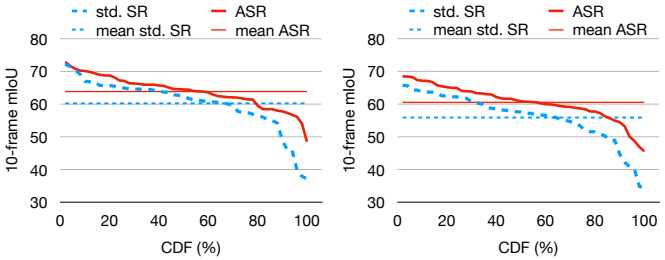
Fig. 9: Enlarged details of object detection on ×4 downsampling with standard SR (top) and ASR frames (down).

2) Frame-wise tail accuracy: Mitigating high-percentile accuracy over time is important to the operation of robotics applications (§ II). We show 90% and 99% accuracy on Cityscapes validation dataset for semantic segmentation in Figure 10. Each data point is an average accuracy on an episode of 10 frames. The horizontal lines for mean mIoU are different from the overall mIoU in Figure 7, because the overall mIoU is accumulated on the whole dataset, while the mean mIoU in Figure 10 is the average accuracy of all frame episodes.

Figure 10 shows the frame-wise accuracy distribution with ×4 and ×6 inputs. ASR improves all frames in general, especially the tail ones, thanks to its focus on detailed information. Table III shows the tail accuracy in detail. ASR improves 90% accuracy by 18% to 22%, and 99% accuracy by 35% to 54% comparing with the standard SR (NAS).

B. Overall accuracy and bandwidth saving

As we focus on the tail accuracy performance, ASR naturally improves the overall accuracy at the same bandwidth, which is also important for edge-cloud systems. We only present



(a) ×4 downsampling (b) ×6 downsampling

Fig. 10: Frame-wise accuracy distribution of semantic segmentation. ASR especially improves tail accuracy performance.

10-frame mIoU(%)		**90%**	**99%**	mean
×4 512×256	standard SR	49.01	37.93	60.29
	ASR	**57.83**	**54.03**	63.88
×6 341×170	standard SR	44.75	34.86	55.93
	ASR	**54.51**	**46.92**	60.58

TABLE III: 90%/99% frame-wise tail accuracy (segmentation).

evaluations on semantic segmentation due to space limitation, while object detection shows similar results.

Table IV compares the overall accuracy of semantic segmentation across these schemes. We can see that ASR improves the overall accuracy by 3% to 49% compared with Bilinear algorithm (AWStream [15]) and 1% to 33% compared with the standard SR (NAS [25]). Figure 11 shows the accuracy-bandwidth comparison of Runespoor, NAS, and AWStream. From a bandwidth saving perspective, to achieve the optimal overall accuracy (75% mIoU), ASR outperforms NAS by 3.5× and AWStream by 14.3×. To achieve 70% mIoU, ASR can still save 2.1× and 6.9× bandwidth consumption.

scale	metric	Bilinear	standard SR	**ASR**	HR
×2	mIoU(%)	73.11	74.66	**75.12**	75.35
	PSNR	36.72	40.13	39.91	—
	SSIM	0.981	0.991	0.989	
×4	mIoU(%)	65.87	71.15	**75.04**	75.35
	PSNR	31.79	35.34	35.06	—
	SSIM	0.946	0.972	0.968	
×6	mIoU(%)	53.13	65.14	**70.81**	75.35
	PSNR	29.34	33.30	32.75	—
	SSIM	0.917	0.955	0.950	
×8	mIoU(%)	40.49	45.40	**60.32**	75.35
	PSNR	27.81	30.62	30.57	—
	SSIM	0.897	0.929	0.924	

TABLE IV: Overall accuracy comparison for semantic segmentation. Runespoor achieves higher accuracy than other baselines while exhibiting similar visual quality.

To show the image reconstruction quality, we use PSNR (peak signal-to-noise ratio) [22] and SSIM (structural similarity index) [23]. In Table IV, ASR shows a slightly lower reconstruction quality compared with the standard SR, because the standard SR is trained only towards the similarity target, while ASR is not. We design ASR to sacrifice some image quality (e.g., clouds in the sky) for higher DL inference accuracy on details that are critical to tail accuracy.

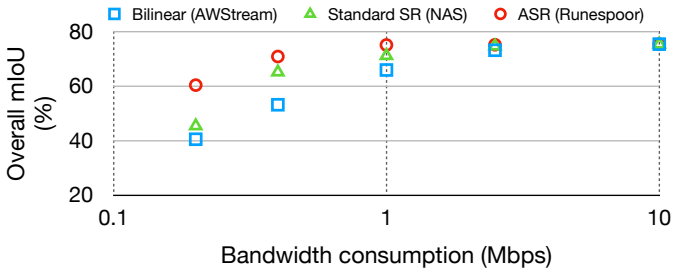


Fig. 11: Accuracy and bandwidth consumption at a same high-quality encoding. X-axis is in log scale. Runespoor achieves higher accuracy with the same bandwidth.

C. End-to-end performance with CAC

Content-aware adaptive controller (CAC) works in the online operation of edge-cloud video analytics of robotics applications. We use a Linux desktop as the edge device and a local server with 4 NVIDIA K40m GPUs as the cloud. We use the Linux `tc` utility to modify the outgoing bandwidth of the edge device to emulate the edge-cloud environment. Cityscapes only labels 1 frame in every 30 frames (1.8s), so does VisDrone. We choose Cityscapes dataset to evaluate CAC for its diversity of scenes. To simulate the real-world operation, we use a consecutive 20-second 17-fps *unlabeled* video clip (340 frames) to evaluate CAC on frame-wise tail accuracy handling. We use the DL predictions on original HR as the baseline to evaluate the relative frame-wise accuracy. We process the baseline and mark the ignored areas to make it in Cityscapes label format. Figure 12 shows the average relative accuracy of every second.

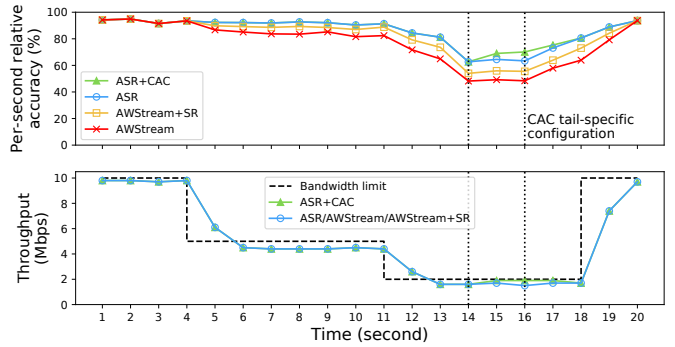


Fig. 12: CAC improves tail accuracy in the real-world video stream. Accuracy is relative to performance on HR.

We initially set the outgoing bandwidth to 10 Mbps (2048×1024, 17 fps, 90 encoding quality), which is an abundant 4G uploading bandwidth [44]. There are four stages in our evaluation: (i) before t=5s, set the bandwidth to 10 Mbps; (ii) at t=5s, limit the bandwidth to 5 Mbps, which is typical for uploading; (iii) at t=12s, reduce the bandwidth to 2 Mbps; (iv) at t=19s, reset the bandwidth to 10 Mbps. During 14s to 16s, the frames that lead to low tail accuracy occur.

Figure 12 shows that CAC takes ∼500ms of latency to detect and switch to the tail-specific knob configuration (higher resolution and lower encoding quality) for the subsequent frames, comparing with the ∼10 seconds of delay to detect scene changes in AWStream [15]. At around t=15s, Runespoor with CAC achieves a higher tail accuracy than ASR only. At t=17s, CAC switches back to the standard configuration. CAC, including the tail-specific configuration (14s-16s), has a similar throughput as other baselines, as shown in the lower figure. This is because we learn these configs through offline profiling regarding the same bandwidth targets, as discussed in § III-B.

We show the breakdown of end-to-end latency per frame in Table V. Each application has 410-530ms and 330-420ms latency. DL computations are distributed over multiple GPUs. Runespoor is efficient to support our target applications.

time (ms)	transmission+processing				ASR	inference
	×2	×4	×6	×8		
segmentation	144	40	20	14	68-80	316
detection	133	37	—	—	66-70	220

TABLE V: Breakdown of end-to-end latency per frame.

VI. Related Work

A. Edge-cloud video analytics systems

A common approach to manage the bandwidth-accuracy trade-off is to reduce the video quality at the edge, e.g., via pixel-level and frame-level downsampling. The degradation must ensure that sufficient information is retained so that the cloud can run video analytics on downsampled videos to produce sufficiently accurate results. AWStream [15] learns the Pareto-optimal policy for a task, and adaptively selects a video degradation strategy to balance accuracy and bandwidth. FilterForward [16], along with other filter-based frameworks [17], [30], selectively filters video frames at edge to reduce bandwidth consumption. The insight is that for fixed-view surveillance applications, there are a lot of non-relevant frames that can be skipped in DL video analytics.

B. Super-resolution for video analytics

Prior work in both system and CV has shown that SR is a promising approach to improving video streaming quality [25] and analytics accuracy [27], [28], [45] when only low-resolution videos are available. NAS [25] trains content-aware SR models and deploys them on the client device to improve the video streaming QoE with limited bandwidth. However, directly applying NAS to video analytics systems does not address the tail accuracy issue.

VII. Conclusion

We propose Runespoor to enable autonomous robotics applications to conduct advanced video analytics efficiently with edge-cloud computing. Runespoor improves *tail accuracy* with *analytics-aware SR* to reconstruct detailed information from compressed data, and *content-aware adaptive controller* to instantly adapt to fast-changing scenes. Our results show that Runespoor successfully improves the tail accuracy with minimal bandwidth consumption.

Acknowledgement: This work is supported in part by the Hong Kong RGC TRS [T41-603/20-R] and GRF-16215119. Kai Chen is the corresponding author of this paper.

REFERENCES

[1] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv Preprint arXiv:1804.02767*, 2018.

[2] M. Orsic, I. Kreso, P. Bevandic, and S. Segvic, "In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 607–12 616.

[3] M. Teichmann, M. Weber, M. Zoellner, R. Cipolla, and R. Urtasun, "MultiNet: Real-time joint semantic reasoning for autonomous driving," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018.

[4] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, "Panoptic segmentation," *arXiv Preprint arXiv:1801.00868*, 2018.

[5] M. Simon and A. Pardes. (2019) The prime challenges for Scout, Amazon's new delivery robot - WIRED. [Online]. Available: https://www.wired.com/story/amazon-new-delivery-robot-scout/

[6] A. Marshall. (2020) Delivery robots aren't ready when they could be needed most - WIRED. [Online]. Available: https://www.wired.com/story/delivery-robots-arent-ready-when-needed-most/

[7] S.-C. Lin, Y. Zhang, C.-H. Hsu, M. Skach, M. E. Haque, L. Tang, and J. Mars, "The architectural implications of autonomous driving: Constraints and acceleration," in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 2018, pp. 751–766.

[8] I. Boudwayn. (2020) Delivery robot operators are also working from home - Bloomberg. [Online]. Available: https://www.bloomberg.com/news/articles/2020-05-13/delivery-robot-operators-are-also-working-from-home

[9] L. Fridman, D. E. Brown, M. Glazer, W. Angell, S. Dodd, B. Jenik, J. Terwilliger, J. Kindelsberger, L. Ding, S. Seaman *et al.*, "MIT autonomous vehicle technology study: Large-scale deep learning based analysis of driver behavior and interaction with automation," *arXiv Preprint arXiv:1711.06976*, 2017.

[10] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman, "Live video analytics at scale with approximation and delay-tolerance." in *NSDI*, vol. 9, 2017, p. 1.

[11] K. Hsieh, G. Ananthanarayanan, P. Bodik, S. Venkataraman, P. Bahl, M. Philipose, P. B. Gibbons, and O. Mutlu, "Focus: Querying large video datasets with low latency and low cost," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 2018.

[12] H. Shen, L. Chen, Y. Jin, L. Zhao, B. Kong, M. Philipose, A. Krishnamurthy, and R. Sundaram, "Nexus: A gpu cluster engine for accelerating dnn-based video analysis," in *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, 2019, pp. 322–337.

[13] S. Hu, W. Bai, G. Zeng, Z. Wang, B. Qiao, K. Chen, K. Tan, and Y. Wang, "Aeolus: A building block for proactive transport in datacenters," in *Proceedings of the 2020 Conference of the ACM Special Interest Group on Data Communication*, 2020, pp. 422–434.

[14] J. Zhang, W. Bai, and K. Chen, "Enabling ecn for datacenter networks with rtt variations," in *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, 2019.

[15] B. Zhang, X. Jin, S. Ratnasamy, J. Wawrzynek, and E. A. Lee, "AWStream: Adaptive wide-area streaming analytics," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. ACM, 2018, pp. 236–252.

[16] C. Canel, T. Kim, G. Zhou, C. Li, H. Lim, D. G. Andersen, M. Kaminsky, and S. R. Dulloor, "Scaling video analytics on constrained edge nodes," in *2nd Conference on Systems and Machine Learning (SysML)*, 2019.

[17] T. Y.-H. Chen, L. Ravindranath, S. Deng, P. Bahl, and H. Balakrishnan, "Glimpse: Continuous, real-time object recognition on mobile devices," in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2015, pp. 155–168.

[18] H. Ding, X. Jiang, A. Q. Liu, N. M. Thalmann, and G. Wang, "Boundary-aware feature propagation for scene segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[19] S. Zhao, Y. Wang, Z. Yang, and D. Cai, "Region mutual information loss for semantic segmentation," in *Advances in Neural Information Processing Systems*, 2019, pp. 11 115–11 125.

[20] B. Chen, C. Gong, and J. Yang, "Importance-aware semantic segmentation for autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 1, pp. 137–148, 2018.

[21] Z. Wu, X. Wang, J. E. Gonzalez, T. Goldstein, and L. S. Davis, "Ace: Adapting to changing environments for semantic segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 2121–2130.

[22] A. Hore and D. Ziou, "Image quality metrics: PSNR vs. SSIM," in *2010 20th International Conference on Pattern Recognition*. IEEE, 2010.

[23] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli *et al.*, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[24] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.

[25] H. Yeo, Y. Jung, J. Kim, J. Shin, and D. Han, "Neural adaptive content-aware internet video delivery," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 2018.

[26] N. Ahn, B. Kang, and K.-A. Sohn, "Fast, accurate, and lightweight super-resolution with cascading residual network," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 252–268.

[27] Y. Wang, W. Wang, J. Zhang, J. Jiang, and K. Chen, "Bridging the edge-cloud barrier for real-time advanced vision analytics," in *11th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 19)*, 2019.

[28] D. Dai, Y. Wang, Y. Chen, and L. Van Gool, "Is image super-resolution helpful for other vision tasks?" in *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*. IEEE, 2016, pp. 1–9.

[29] D. Guo, L. Zhu, Y. Lu, H. Yu, and S. Wang, "Small object sensitive segmentation of urban street scene with spatial adjacency between object classes," *IEEE Transactions on Image Processing*, vol. 28, 2018.

[30] T. Zhang, A. Chowdhery, P. Bahl, K. Jamieson, and S. Banerjee, "The design and implementation of a wireless video surveillance system," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, 2015, pp. 426–438.

[31] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGARCH Computer Architecture News*, 2017.

[32] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for mobilenetv3," *arXiv Preprint arXiv:1905.02244*, 2019.

[33] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[34] TensorFlow, "TensorFlow DeepLab Model Zoo," https://github.com/tensorflow/models/blob/master/research/deeplab/g3doc/model_zoo.md.

[35] S. P. Chinchali, E. Cidon, E. Pergament, T. Chu, and S. Katti, "Neural networks meet physical networks: Distributed inference between edge devices and the cloud," in *Proceedings of the 17th ACM Workshop on Hot Topics in Networks*. ACM, 2018, pp. 50–56.

[36] J. Fowers, K. Ovtcharov, M. Papamichael, T. Massengill, M. Liu, D. Lo, S. Alkalay, M. Haselman, L. Adams, M. Ghandi *et al.*, "A configurable cloud-scale dnn processor for real-time ai," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2018, pp. 1–14.

[37] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "Labelme: A database and web-based tool for image annotation," *International Journal of Computer Vision*, vol. 77, no. 1-3, pp. 157–173, 2008.

[38] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, 2019.

[39] P. M. Grulich and F. Nawab, "Collaborative edge and cloud neural networks for real-time video processing," *Proceedings of the VLDB Endowment*, vol. 11, no. 12, pp. 2046–2049, 2018.

[40] Y. Liu, C. Shen, C. Yu, and J. Wang, "Efficient semantic video segmentation with per-frame inference," *arXiv:2002.11433*, 2020.

[41] Autopilot Tesla. [Online]. Available: https://www.tesla.com/autopilotAI

[42] P. Zhu, L. Wen, X. Bian, H. Ling, and Q. Hu, "Vision meets drones: A challenge," *arXiv Preprint arXiv:1804.07437*, 2018.

[43] M. Orsic, "SwiftNet," https://github.com/orsic/swiftnet, 2019.

[44] "USA Mobile Network Experience Report, July 2019," https://www.opensignal.com/reports/2019/07/usa/mobile-network-experience, 2019.

[45] M. Haris, G. Shakhnarovich, and N. Ukita, "Task-driven super resolution: Object detection in low-resolution images," *arXiv Preprint arXiv:1803.11316*, 2018.