ITSVA: Toward 6G-Enabled Vision Analytics over Integrated Terrestrial-Satellite Network

Miao Zhang*, Jiaxing Li*, Jianxin Shi†, Yifei Zhu‡, Lei Zhang§, Hengzhi Wang§
* School of Computing Science, Simon Fraser University, Canada
† College of Computer Science, Nankai University, China
‡ UM-SJTU Joint Institute, Shanghai Jiao Tong University, China
§ College of Computer Science and Software Engineering, Shenzhen University, China
{mza94, jla641}@sfu.ca, shijianxin@mail.nankai.edu.cn, yifei.zhu@sjtu.edu.cn, {leizhang, whz}@szu.edu.cn

***Abstract*—The sixth-generation (6G) mobile communications system is expected to integrate the terrestrial and low Earth orbit satellite networks (LSN) to provide seamless global Internet service. This will create new opportunities for edge-assisted mobile vision analytics (MVA), which offloads frames over networks to edge servers for analysis, thereby overcoming the local computational resource constraints. With the integrated terrestrial and LSN (ITLSN), edge-assisted MVA can reach its full potential in remote and maritime areas. Nevertheless, the proximity of LEO satellites to the Earth is a double-edged sword. It offers benefits in latency and data rates but also brings challenges like frequent satellite handovers and volatile channel conditions. To demystify the in-the-wild performance of ITLSN, we carry out large-scale measurements with a major LSN service provider. The measurement results confirm the highly asymmetric and dynamic network performance of today's ITLSN, which can present non-trivial challenges for MVA frame offloading. We thus propose an ITLSN-adaptive MVA offloading framework, `ITSVA`, to address the inherent dynamics brought by network conditions, video content, and the offloading strategy. Extensive trace-driven simulation experiments are further conducted to verify the effectiveness of `ITSVA`.**

***Index Terms*—mobile vision analytics, LEO satellites, 6G**

I. Introduction

Mobile vision analytics (MVA) [1]–[4], which enables machines to understand the physical world by analyzing videos captured by mobile devices in real time, is a critical technical enabler to realizing emerging extended reality (XR) applications. By analyzing video content with deep neural network (DNN) based vision models, MVA can help bridge the gap between the physical and virtual worlds. For instance, object detection models [5] can locate and classify objects appearing in the field of view of a camera, thus providing a better situational awareness; human keypoint detection models [6] can identify and locate keypoints or landmarks on a human body within an image or video frame, enabling better interaction between humans and virtual objects. Given the constrained computational resources and heat dissipation issues of mobile devices, existing MVA systems tend to offload heavy DNN inference workloads to edge servers so as to meet the stringent latency and tremendous resource demands [1], [7], [8].

Unfortunately, almost all edge-assisted MVA systems transmit video data over terrestrial networks, substantially limiting the potential in remote and maritime areas. The next generation of mobile system standard, 6G, is expected to integrate mega-constellations of low Earth orbit (LEO) satellites with terrestrial networks to provide seamless broadband Internet services [9], opening up exciting opportunities for edge-assisted MVA. Compared to their geosynchronous Earth orbit and medium Earth orbit counterparts, LEO satellites are much closer to the Earth, with altitudes typically below $2,000$ km. This proximity dramatically reduces signal travel distance, resulting in lower communication latency and higher data rates. For example, one major LEO broadband service provider, *SpaceX Starlink*, promises a latency of 25-60 ms and a download speed of 25-100 Mbps in the standard service plan.[1] The low latency and higher data rate make the integrated terrestrial and LEO satellite network (ITLSN) a potential fit for MVA applications in currently underserved areas.

As the LEO satellite network (LSN) evolves to global coverage, the ITLSN will seamlessly connect anyone, anywhere on the planet. This will also expand the reach of edge-assisted MVA applications to a broader audience. For example, tourists and adventurers will have access to augmented reality (AR) guides in remote and wilderness areas for a fun experience and improved safety. Global mobile users will be able to engage in diverse collaborative XR games, whether they are travelling on high-speed trains, flying in planes, or sailing on the ocean.

Despite the huge potential, it can be challenging for today's LSN to support efficient MVA offloading. First, inheriting the download-centric design principle of previous generations of mobile networks, LSN offers *highly asymmetric* performance, with the mean download throughput about 13.3× higher than the mean upload throughput according to our measurements. This is disadvantageous to edge-assisted MVA, which relies on uplink resources to stream high-quality frames to the edge server for analysis [3], [8]. Moreover, due to their low orbital altitudes, LEO satellites have a small coverage area and move faster than the Earth, causing frequent satellite handovers that drastically impair the network performance stability. This is further exacerbated by the fact that the LSN is more prone to be affected by various environmental factors (e.g., precipitation and temperature) [10]. Therefore, the ITLSN exhibits

[1] https://www.starlink.com/legal/documents/DOC-1400-28829-70, [Online; accessed Sep 20, 2023]

highly volatile network performance. For example, according to our observations, the upload throughput can fluctuate wildly between 27.6 Mbps and 2.6 Mbps within a minute.

In an ideal edge-assisted MVA scenario, the offloading delay of a frame should not be greater than a frame playback duration with the intent to maintain the post-processing and rendering at the capture frame rate. Yet, the scarce upload bandwidth of today's ITLSN makes high frame rate offloading an elusive goal. Periodically offloading at a fixed frame interval is also likely to suffer from the volatile uplink performance of the ITLSN, incurring unacceptable delays in the offloading pipeline. As such, network-adaptive offloading solutions that dynamically tune the offloading interval can be necessary.

However, adapting the frame offloading interval to the ITLSN's uplink conditions is not an easy job. A small offloading interval may cause network congestion and delay the offloading pipeline, ultimately rendering the returned remote inference result outdated. Nonetheless, a large offloading interval can also make the post-processing and rendering process persistently use stale analysis results. To make things worse, the performance of an offloading interval is also influenced by video content dynamics and the setting of the previous offloading intervals. Accurate network prediction can be helpful for the offloading interval setting. That being said, influenced by a variety of unobservable internal and external factors, e.g., unpredictable weather conditions, accurate uplink performance prediction of the ITLSN is hard to achieve.

In this paper, we propose a novel framework, called `ITSVA`, to address the challenges brought by the inherent dynamics of 6G-enabled MVA. Specifically, `ITSVA` achieves high accuracy and low network overhead through fine-grained network and video content adaptation. The main contributions of this paper can be summarized as follows.

- By conducting extensive *in-the-wild* measurements with a real-world setup, we reveal the uplink characteristics of today's ITLSN and further identify the key challenges toward building high-performance 6G-enabled MVA systems over the ITLSN.
- We propose `ITSVA`, an ITLSN-adaptive MVA offloading framework that can achieve both high accuracy and low network data transfer overhead. Particularly, `ITSVA` utilizes an optical flow-based local tracker to boost the accuracy when the remote inference results are unavailable. It also integrates a deep reinforcement learning (DRL) agent to make fine-grained frame offloading decisions that take into account the network variations, video content dynamics, and previous decisions.
- We collect large-scale ITLSN uplink network traces from the real world and design a trace-driven simulator to evaluate the performance of `ITSVA`. The results of extensive experiments show that `ITSVA` can achieve higher accuracy with lower network overhead than representative baselines.

II. Background and Motivation

In this section, we first provide a brief background introduction of edge-assisted MVA. We then investigate the performance characteristics of today's ITLSN through large-scale in-the-wild measurements.

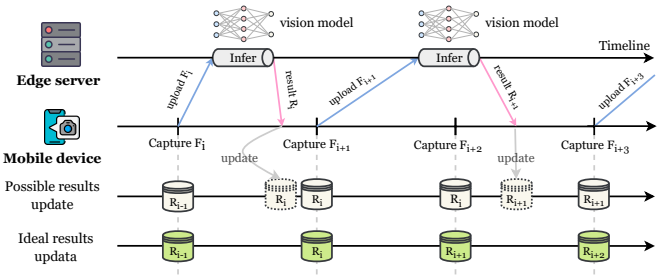


Fig. 1: Frame offloading workflow of edge-assisted MVA.

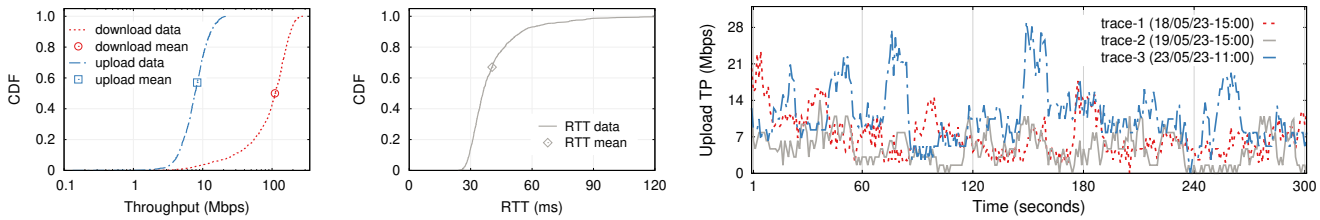
A. Edge-Assisted Mobile Vision Analytics

Fig. 1 shows a typical frame offloading workflow of edge-assisted MVA, where frames are offloaded to the edge server for analytics as they are captured by the mobile device camera. The edge server processes the offloaded frame with the vision DNN model and sends the inference results back to the mobile device for post-processing or rendering. Ideally, the entire offloading pipeline can run in *real time*, i.e., the mobile device can obtain the remote inference result of the frame F_i before the capture of the next frame F_{i+1} [3]. In this case, the per-frame offloading latency should not exceed 33.3 ms for a camera recording videos at a frame rate of 30 FPS. Although state-of-the-art vision models can run at a higher frame rate than 30 FPS [5] on a typical data center GPU, the network delays of transmitting high-quality frames will become the pipeline latency bottleneck, especially when the network conditions between the mobile device and the edge server are poor. If the frame offloading pipeline is delayed, the offloaded frame and its subsequent frames will have no choice but to use the stale inference result, which can significantly lower the overall accuracy.

A number of solutions have been proposed to address the network resource challenges of achieving the dual performance goals of low latency and high accuracy in edge-assisted MVA. Some strategies attempt to reduce the offloaded data size by applying video or image compression algorithms on the frames [1]. However, the reduced frame resolution or degraded image quality caused by various lossy compression algorithms can hurt the accuracy of server-side vision model inference. Others concentrate on periodically offloading frames to reduce the network data transfer overhead while utilizing diverse on-device approaches (e.g., compressed DNN models or local trackers) to compensate for the accuracy of the unoffloaded frames [3]. Nonetheless, determining when and which frames to offload for overall performance optimization remains an ongoing challenge.

B. Characteristics of ITLSN

To gain a deep understanding of the in-the-wild performance of ITLSN, we perform large-scale measurements with Starlink. We use a Raspberry Pi 4 Model B as the mobile device and



(a) TCP throughput statistics. (b) Network latency statistics. (c) Upload throughput varies over time.

Fig. 2: Characteristics of integrated terrestrial and LEO satellite network (SpaceX Starlink as a case study).

rent the edge server from the nearest AWS data center to the mobile device. With this setup, we use the `iPerf3` utility to measure the TCP upload and download throughput and the `Ping` utility to measure the round-trip time (RTT).

Fig. 2a and Fig. 2b demonstrate the statistical results over more than one thousand tests from different times of the day. Each test lasts for 10 seconds. As can be seen, although the mean download throughput achieves around 110 Mbps, the mean upload throughput is only about 8.3 Mbps, and 16.2% of the tests have an upload throughput even lower than 5 Mbps. The relatively low upload throughput of ITLSN indicates that transferring frames from mobile devices to edge servers may cause significant delays in the offloading pipeline. Moreover, the mean RTT of all tests is about 40.5 ms. Unfortunately, such a network latency cannot support a 30 FPS camera offloading each frame back to back in real time. This suggests that today's ITLSN still cannot support high frame rate offloading, and specialized designs are needed for 6G-enabled MVA.

We also collected longer upload network traces with a one-second granularity. Fig. 2c displays the upload throughput variations of three traces. As shown, the upload throughput of the ITLSN fluctuates dramatically over time. For instance, the upload throughput of *trace-3* can vary wildly from 27.6 Mbps to 2.6 Mbps within one minute. Also, the upload throughput fluctuations are somewhat random, with no apparent daily patterns observed (e.g., *tace-1* and *trace-2*). When the network conditions are poor, it is likely to observe service outages where the throughput drops to 0, e.g., from second 242 to second 244 of *trace-2*. Compared to other types of networks, ITLSN experiences more frequent serving satellite handovers and is more susceptible to changing weather conditions, leading to inherent fluctuations in network performance. This calls for network-dynamics-aware designs for 6G-enabled MVA to deliver consistent quality of experience.

III. System Design

In this section, we propose an edge-assisted MVA framework specifically designed for ITLSN, called `ITSVA`, which can overcome the unique network challenges of ITLSN and achieve high overall performance.

A. Framework Overview

In a typical Starlink residential setup, a router connects the user device to a dish (antenna) that communicates directly with the LEO satellites. User data are then transmitted to a ground station that acts as a gateway to the Internet. Fig. 3 presents an overview of `ITSVA`, where the mobile device connects to the Internet via LSN. `ITSVA` assumes that the analytics server is placed as close to the edge of the LSN (i.e., the ground station) as possible to reduce network latency.

Assume that the mobile device's camera captures frames at a constant rate of f FPS. At the beginning of every second t, the *offloading scheduler* of `ITSVA` makes the offloading decision for all frames captured between t and $t+1$. To be specific, if a frame is scheduled to be offloaded, it will be sent over the network to the remote server for inference. Note that the frames scheduled for remote inference are offloaded one by one in order of capture time. Meanwhile, the *local tracker* adapts the latest remote inference result, obtained from a local cache, to each newly captured frame. At the capture time of each frame F_{i+1}, the *post-processing and rendering* module queries the *offloading scheduler* for the remote inference result of the frame F_i. If the query is successful, the remote inference result will be used. If the remote inference result is unavailable due to offloading timeout or not being offloaded, the module will utilize the tracked result from the *local tracker* instead. Once a new remote inference result is returned, the *offloading scheduler* immediately updates the *local cache* with that result.

B. Optical Flow-Based Local Tracker

Our measurement results necessitate selective frame offloading of `ITSVA`, given the scarce and volatile uplink resources of the ITLSN. For unoffloaded frames, a straightforward solution is to reuse the inference result of the latest offloaded frame since the video content exhibits certain temporal continuity in successive frames. However, this solution provides high accuracy only if the video content has insignificant motion between two consecutive offloadings. In MVA, both the recorded objects and the camera can move, resulting in highly dynamic content. As a result, adapting the stale remote inference result to the newly captured content is necessary to improve the accuracy of unoffloaded frames. For the object detection task we focus on in this paper, a recent study [3] has verified that compared with other sophisticated designs, an on-device object tracker can dramatically boost the accuracy of unoffloaded frames at low resource costs. We thus integrate a *local tracker* into `ITSVA` to increase its resilience to varying network conditions and preserve accuracy.

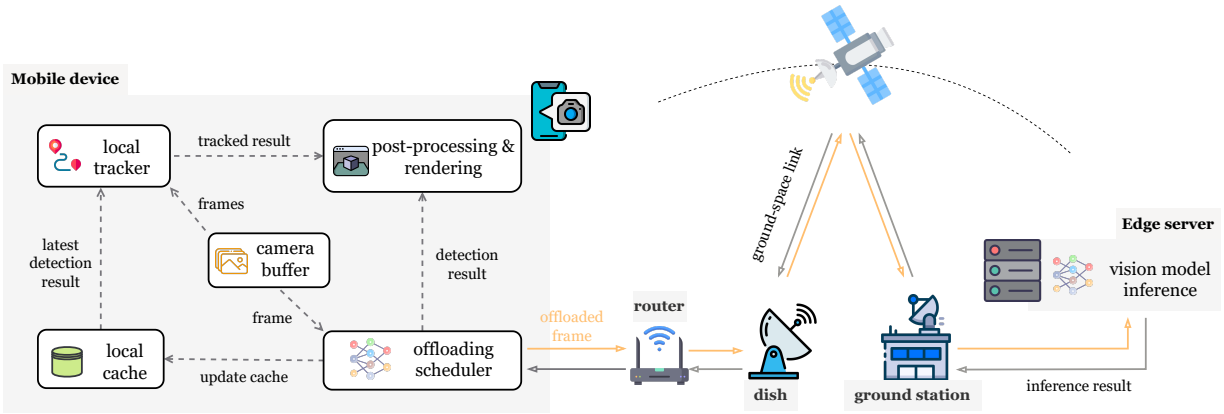


Fig. 3: Overview of ITSVA.

Optical flow is widely used to understand and quantify how objects move in a sequence of video frames, which is well suited for MVA object tracking. As such, ITSVA leverages the pyramidal Lucas-Kanade (LK) optical flow algorithm [11] to track the detected objects, which is able to address both the image object and camera motion. To achieve lightweight implementation, we take the bounding box coordinates of the latest detected objects as the input feature set to obtain the updated object coordinates in the subsequent frames.

C. DRL-based Offloading Scheduler

To allow fine-grained network adaptation while avoiding the high overhead caused by frequent decision-making, the *offloading scheduler* of ITSVA finds a compromise and makes offloading decisions once a second. Let $\mathcal{F}^t = \{F_1, F_2, \cdots, F_f\}$ denote all frames captured during second t. The *offloading scheduler* then tunes the offloading interval l_t to maximize the overall accuracy while minimizing the offloaded data amount over the network. For instance, if the offloading interval l_t is set to 2, frames $\{F_1, F_3, \cdots\}$ are scheduled to be offloaded. $l_t = 0$ represents a special case where no frames are considered for offloading.

Determining an appropriate offloading interval presents non-trivial challenges. A small interval can overload the uplink and cause increased delays for per-frame offloading. This further makes the inference result stale, reducing overall accuracy. Nevertheless, a large interval can also degrade overall accuracy because the local tacker can only be accurate for a short time horizon. Moreover, the performance of an interval is also related to the specific video content. Drastically varying video content may suggest a small interval to ensure accuracy, while a large interval may be beneficial if the video content remains roughly unchanged. In addition, the interval set for the second t can have a cascading effect on the performance of subsequent seconds. The reason is that different settings of l_t can lead to different levels of freshness for the locally cached detection result, thus affecting the subsequent tracking performance.

ITSVA relies on a deep reinforcement learning (DRL) [12] based *offloading scheduler* to address the inherent and complicated system dynamics brought by the network conditions, video content, and previous decisions. The DRL approach trains an agent through experience and feedback. It then uses the trained agent for online decision-making. We first describe the state s_t, which is the input to the DRL agent at the second t and consists of a set of observable variables representing the situations of the environment.

$$s_t = (\vec{n_t}, \vec{u_t}, \vec{d}_{t-1}, \vec{h}_{t-1}, \delta_t, \vec{p_t}) \tag{1}$$

where $\vec{n_t} = \{n_{t-k}, \cdots, n_{t-1}\}$ is the observed upload throughput for the past k seconds. $\vec{u_t} = \{u_{t-k}, \cdots, u_{t-1}\}$ is the mean per-frame offloading delay for the past k seconds. Both n_t and u_t are indicators of recent network conditions. $\vec{d}_{t-1}$ is the offloading decision vector of the last second.[2] $\vec{h}_{t-1}$ represents the offloading success vector of the last second, which reflects the quality of the last offloading decision. δ_t is the distance in frames between the the latest successfully offloaded frame and the latest captured frame, which indicates the freshness of the cached detection result. $\vec{p_t} = \{p_1, p_2, \cdots, p_f\}$ is the optical flow vector for frames in $\mathcal{F}^t$. Specifically, p_i is calculated as:

$$p_i = \sum_j \left(\frac{|x'_{i,j} - x_{i-1,j}|}{w} + \frac{|y'_{i,j} - y_{i-1,j}|}{h}\right) \tag{2}$$

where w and h are the frame width and height, respectively. $(x_{i-1,j}, y_{i-1,j})$ is the coordinate of feature point j in frame F_{i-1}, and $(x'_{i,j}, y'_{i,j})$ is the optical-flow tracked coordinate of the feature point in frame F_i. p_i accumulates the normalized displacements of all tracked features points and quantifies their motions. Based on this, $\vec{p_t}$ summarizes the most recent video content dynamics.

Methodology: For each input state s_t, the DRL agent selects an action a_t based on a trained policy $\pi_\theta(s_t, a_t) \rightarrow [0, 1]$, where θ is the policy parameter. For our problem, the action a_t corresponds to the offloading interval l_t. The policy $\pi_\theta(s_t, a_t)$ is typically represented by a neural network. In our design, a 1D convolutional neural network layer is applied to extract

[2]Note that not all frames scheduled for offloading will be successfully offloaded within a second. To avoid wasting resources in offloading stale frames, unoffloaded frames of $t-1$ will no longer be considered for offloading during second t, and any incomplete offloading of $t-1$ will also be aborted.

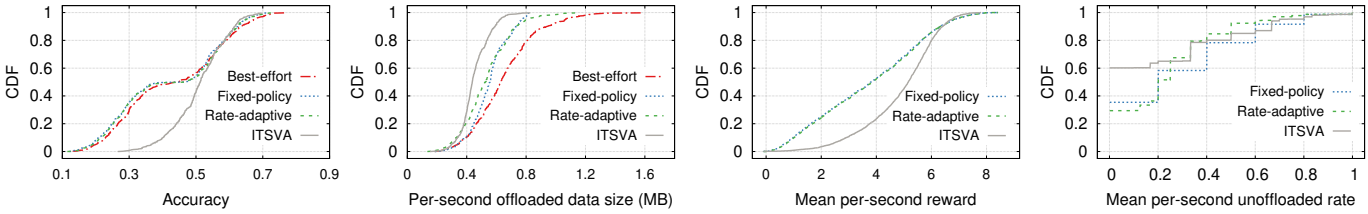


Fig. 4: Performance comparison of different solutions (statistics of all video-trace pairs in the test set)

features from each vector type input, including $\vec{n}_t$, $\vec{u}_t$, $\vec{d}_{t-1}$, $\vec{h}_{t-1}$, and $\vec{p}_t$. Meanwhile, a fully connected layer is used to process δ_t. All output features are then flattened and combined into a new layer, which is further fed into another fully connected layer and a softmax layer to obtain the final action probability distribution.

Once an action a_t is taken, the agent receives an immediate reward r_t from the environment, which indicates the quality of the chosen action a_t in state s_t. Since the performance goal of MVA is to maximize accuracy while minimizing offloading overhead, the reward function is defined as follows:

$$r_t = \alpha_1 A_t - \alpha_2 O_t - \alpha_3 U_t \tag{3}$$

where A_t is the overall analytics accuracy of all frames in $\mathcal{F}^t$; O_t is the actually offloaded data size, indicating the network traffic consumption for processing the frames in $\mathcal{F}^t$; U_t is the unoffloaded rate, which is calculated by dividing the number of frames unoffloaded by the number of frames scheduled to be offloaded. A high U_t may suggest an inappropriate offloading decision that overestimates the actual network conditions. α_1, α_2, and α_3 are positive weighting hyper-parameters that are used to strike a balance between these three factors.

The agent is then trained to maximize the discounted cumulative rewards $\sum_{t=0}^{\infty} \gamma^t r_t$ with respect to the policy parameter θ, where γ is the discounted factor. Particularly, we employ a popular policy gradient method, proximal policy optimization (PPO) [13], to train the policy neural network, and distributed parallel training is also used to accelerate the training process.

IV. Evaluation

We design and implement a trace-driven simulator to evaluate the performance of ITSVA. Real-world collected ITLSN network traces and pre-recorded videos are employed to drive the simulation. Video frames are fed into the simulator at their original capture frame rate, as if they were captured by a real-world camera.

A. Evaluation Setup

Network traces: We collected a large-scale network dataset incorporating 1,200 ITLSN upload traces using a similar real-world setup as introduced in §II-B. The traces are collected at different times throughout the day to ensure their representativeness. Each network trace has a length of 60 seconds with a granularity of 1 second. The dataset is partitioned, with 80% reserved for training the DRL agent and the remaining 20% for testing.

Video dataset and vision task: We use the high-quality videos with a resolution of 1920×1080 and a frame rate of 30 FPS from the MOTS dataset [14] for evaluation. Especially, 4 videos are used for DRL training, and 2 videos are used for testing. We focus on the object detection vision task, and the remote inference is executed with a pre-trained YOLOv7-w6 model [5]. The inference delay of the model with an NVIDIA GeForce RTX 3080 GPU is about 19 ms. We thus regard 19 ms as the server-side inference delay in the simulation.

Baselines: We compare ITSVA with the following baselines.

- *Best-effort:* This solution offloads frames one by one consecutively. It is unaware of the network conditions and simply offloads the latest captured frame once a remote inference result is returned. Frames that are not scheduled to offload or experience offloading timeout will utilize the cached latest detection result for post-processing and rendering.
- *Fixed-policy:* This solution offloads frames at a fixed interval, i.e., l_t is fixed for all seconds. Specifically, the value of l_t is chosen as the smallest interval that the mean throughput of all network traces can accommodate. This method can be considered as a coarse-grained network-adaptive method.
- *Rate-adaptive:* This solution dynamically adjusts l_t based on the most recent network observations. It predicts the upload throughput as the harmonic mean of the observed throughput values for the past 5 seconds. It then sets l_t to the minimum interval that will not make the offloaded data size exceed the predicted network capacity. This solution can be seen as a fine-grained, network-adaptive method that only considers future network conditions.

Evaluation metrics: The first evaluation metric we are interested in is *accuracy*. If all frames can be successfully offloaded within a frame duration, each frame will use its own remote inference result for post-processing and rendering, leading to the highest possible accuracy. Therefore, we consider the server-side inference result of each frame as its ground truth and compare the results obtained by different solutions with it to calculate their accuracy. In particular, we use the F1 score for the object detection task, and a true positive is accepted if the predicted and ground-truth bounding boxes have the same object category and their intersection over union is greater than 0.5. Another evaluation metric is *per-second offloaded data size*. It is calculated by dividing the total offloaded data size by the total offloading delay. It indicates the consumption of network resources. For solutions that make decisions every second, we also consider the *mean per-second reward* and the *mean per-second unoffloaded rate*.

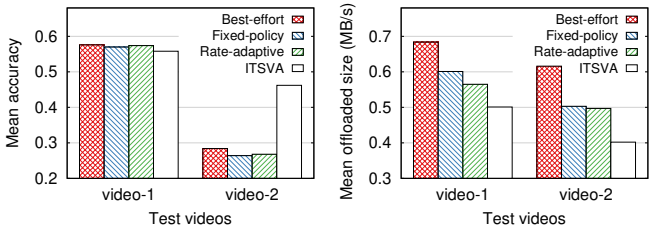


Fig. 5: Performance comparison on different test videos

Action and hyper-parameter settings: As the frame rate of all videos is 30 FPS, we consider all factors of 30 as the candidate intervals to ensure the independence of decisions made for each second, i.e., $l_t = \{0, 1, 2, 3, 5, 6, 10, 15, 30\}$. The look-back window k for network-related inputs is 5. For the policy network, the number of filters and kernel size for all 1D CNN layers are 128 and 3, respectively; the number of neurons for each fully connected layer is 128. The learning rate for the agent training is $1e^{-4}$, and the discounted factor γ is 0.99. The reward weights α_1, α_2, and α_3 are empirically set to 10, 0.1, and 0.01, respectively.

B. Evaluation Result

Fig. 4 displays the performance of different solutions on all test video-trace combinations. As can be seen, *fixed-policy* and *rate-adaptive* achieve similar accuracy and mean per-second reward. The former is unaware of the fine-grained network and video content dynamics. It cannot fully utilize the network resources when the bandwidth is abundant or save the network resources when the video content remains unchanged. On the other hand, the latter considers the changing network conditions but achieves only slight improvements. This can be attributed to the inaccuracy of its throughput prediction and the lack of knowledge of video content variations. In comparison, *Best-effort* achieves higher accuracy than them but offloads the most data. This solution aggressively offloads frames and cannot efficiently use network resources for frames that benefit accuracy the most.

Compared to the baselines, ITSVA attains the highest overall accuracy with significantly reduced network data transfer overhead. With the knowledge extracted from current network conditions, video content dynamics, and local cache status, ITSVA is able to make rewarding decisions. Furthermore, one can also observe that for about 60% video-trace combinations, ITSVA hits a mean per-second unoffloaded rate of 0. This confirms that the DRL agent is well-trained to select the most appropriate offloading interval.

Fig. 5 further shows the performance of the solutions on each test video. Note that video-1 is captured by a stationary camera, while video-2 is captured by a moving camera. This means that video-2 has higher content dynamics than video-1. As shown, the baselines attain decent accuracy on video-1 at the cost of high data offloading overheads. Yet, they perform poorly on video-2 despite the still high network consumption. By contrast, ITSVA achieves competitive accuracy on video-1 and much higher accuracy on video-2 with drastically reduced network overheads. This verifies the robustness of ITSVA in dealing with dynamic video content.

V. Conclusion

Edge-assisted MVA, a key technology for bridging the gap between the physical and virtual worlds, should be specifically optimized for the ITLSN in 6G to realize its full potential. Offloading video frames directly from space still faces severe performance challenges, such as the scarce uplink resources and the wildly fluctuate network throughput. Motivated by the insights of in-the-wild measurements, this paper overcomes the challenges through a carefully designed framework, ITSVA. It takes the inherent dynamics in the network uplink, video content, and local cache status into consideration to make robust offloading decisions. Extensive evaluation results validate the effectiveness of ITSVA in achieving the goals of high accuracy and low network overhead.

References

[1] L. Liu, H. Li, and M. Gruteser, "Edge assisted real-time object detection for mobile augmented reality," in *Proceedings of The 25th Annual International Conference on Mobile Computing and Networking (MobiCom'19)*, 2019, pp. 1–16.

[2] W. Zhang, Z. He, L. Liu, Z. Jia, Y. Liu, M. Gruteser, D. Raychaudhuri, and Y. Zhang, "Elf: Accelerate high-resolution mobile deep vision with content-aware parallel offloading," in *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking (MobiCom'21)*, 2021, pp. 201–214.

[3] J. Meng, Z. J. Kong, Y. C. Hu, M. G. Choi, and D. Lal, "Do we need sophisticated system design for edge-assisted augmented reality?" in *Proceedings of the 5th International Workshop on Edge Systems, Analytics and Networking (EdgeSys'22)*, 2022, pp. 7–12.

[4] N. Wu, F. X. Lin, F. Qian, and B. Han, "Hybrid mobile vision for emerging applications," in *Proceedings of the 23rd Annual International Workshop on Mobile Computing Systems and Applications (HotMobile'22)*, 2022, pp. 61–67.

[5] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv preprint arXiv:2207.02696*, 2022.

[6] S. Kreiss, L. Bertoni, and A. Alahi, "Pifpaf: Composite fields for human pose estimation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 11 977–11 986.

[7] D. G. Morín, P. Pérez, and A. G. Armada, "Toward the distributed implementation of immersive augmented reality architectures on 5g networks," *IEEE Communications Magazine*, vol. 60, no. 2, pp. 46–52, 2022.

[8] M. Ghoshal, Z. J. Kong, Q. Xu, Z. Lu, S. Aggarwal, I. Khan, Y. Li, Y. C. Hu, and D. Koutsonikolas, "An in-depth study of uplink performance of 5g mmwave networks," in *Proceedings of the ACM SIGCOMM Workshop on 5G and Beyond Network Measurements, Modeling, and Use Cases (5G-MeMU'22)*, 2022, pp. 29–35.

[9] X. Lin, S. Cioni, G. Charbit, N. Chuberre, S. Hellsten, and J. Boutillon, "On the path to 6g: Embracing the next wave of low earth orbit satellite access," *IEEE Communications Magazine*, vol. 59, no. 12, pp. 36–42, 2021.

[10] S. Ma, Y. C. Chou, H. Zhao, L. Chen, X. Ma, and J. Liu, "Network characteristics of leo satellite constellations: A starlink-based measurement from end users," *arXiv preprint arXiv:2212.13697*, 2022.

[11] J.-Y. Bouguet, "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," *Intel corporation*, vol. 5, no. 1-10, p. 4, 2001.

[12] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction, second edition*. MIT press, 2018.

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[14] P. Voigtlaender, M. Krause, A. Osep *et al.*, "Mots: Multi-object tracking and segmentation," in *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 2019, pp. 7942–7951.