Resource Efficient Edge Computing Infrastructure for Video Surveillance

Pavana Pradeep, Amitangshu Pal, Sanjeev Sondur, Krishna Kant
Computer and Information Sciences, Temple University, Philadelphia, USA
Email: {pavana.pradeep, amitangshu.pal, sanjeev.sondur, kkant}@temple.edu

***Abstract*—The emerging edge computing applications often involve multiple co-located or closely located edge devices that generate continuous data streams for consumption by an edge controller (EC). The EC performs online analytics and drives decision making and query processing. High definition cameras are increasingly used to capture video streams that provide an unprecedented understanding of the changing situation in surveillance and related applications. However, such systems suffer from limited energy (and hence limited computing power) at the edge devices and limited bandwidth available to the EC that often varies dynamically. Also, the deployment environments and costs of EC's themselves require high energy efficiency. In this paper, we comprehensively address this problem in the context of vehicular traffic monitoring and develop a scheme that has two components: YLLO and BATS. YLLO is a lightweight object recognition algorithm that exploits computing power on the edge devices to substantially reduce the frame rate sent to and processing at the EC. BATS adapts the transmissions to the available bandwidth by taking advantage of further redundancy in the video stream in both single and multi-camera scenarios.**

We show that these mechanisms together can maintain object identification accuracy of above 95%, while transmitting just ~5-10% of all the frames recorded by the cameras.

***Index Terms*—Edge Computing, Energy Efficiency, Video Analytics, Bandwidth Management**

I. Introduction

Edge computing is a rapidly developing field [1] with numerous real-time analytics applications that support smart cyber-physical systems, such as traffic management, surveillance, patient care, smart manufacturing, etc. Quick analytics of live video streams are essential to ensure safety, planning, and security. In general, video analytics typically require intensive computations, the high energy consumption of end devices, and a further transit latency to the cloud. An extensive cyber-physical system requires multiple Edge Controllers (EC), each receiving data streams from many edge devices and is responsible for temporary data storage and real-time data analytics (possibly in collaboration with other ECs), to derive intelligence. Each EC generally covers a single physical region, and the edge-devices (ED) in that region will typically associate with that EC as shown in Fig. 1.

Most edge computing deployments must deal with limitations such as cost, size, energy consumption, bandwidth, storage capacity, etc. Despite these constraints, EDs continue to advance rapidly in computational capabilities. Although individual devices are restricted, the set of devices connected to the EC collectively represents a significant computational resource that can be exploited to reduce the cost of EC deployments. The ECs themselves are typically deployed in large numbers and a hostile environment (e.g., hanging off a pole in a small enclosure and little cooling); therefore, an EC is constrained by energy consumption, computational resources, and cost. Thus, a suitable monitoring solution must consider the attributes of the entire system.

In this paper, we address energy consumption and computational resource constraint by devising a solution that consists of two components. First, we design a lightweight and energy-efficient video-stream analytics algorithm called **YLLO** that runs on individual edge devices and substantially reduces the video frames sent to the EC. By exploiting the edge computing power, *YLLO* reduces both the frame transmission bandwidth requirements to the EC and further processing needs at the EC. Next, we present another algorithm called **BATS**, that exploits the overlaps between the views of multiple cameras and adapt to the available bandwidth dynamically.

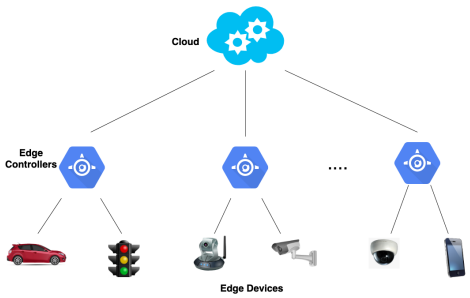


Fig. 1: Hierarchical architecture of edge computing.

Our proposal includes a mechanism for efficiently monitoring the road traffic using fixed road-side cameras as EDs, which interact with ECs through a wireless network. We assume that the cameras are deployed densely to capture the traffic videos from multiple angles with potential overlaps. For example, a telephone or light pole may have 2-3 cameras mounted with different and possibly overlapping fields of view. Similarly, the cameras on poles on opposite sides of the road may have overlapping views. The ECs are likely to be deployed close-by depending partly on the effectiveness of the wireless technology. An obvious target is an already deployed cellular technology, including 4G-LTE and the emerging 5G, and in that case, the ECs could potentially be bundled with base stations. Other techniques include TV whitespace (i.e., the unused gap between TV channels in 470-790MHz range) and even WiFi. We do not assume a specific technology; different technologies have their unique characteristics in terms of range, available bandwidth, and impairments.

This research was support by NSF grant CNS-1527346

YLLO stands for "**Y**ou **L**ook **L**ess Than **O**nce" since it avoids full processing all frames by exploiting the redundancy in a video stream. YLLO borrows basic functions from other popular single-step algorithms such as variants of the popular YOLO algorithm [2]. It provides speed improvements over YOLO which stands for "You Only Look Once". YLLO does not attempt to devise new algorithms for the basic functions such as bounding box or object detection. BATS stands for "**B**andwidth **A**daptive **T**ransmissions of **S**treams" and performs bandwidth adaptation assuming that component deals with the adaptation of frame stream transmission to the transmission channel capacity to the EC is shared among multiple cameras in the same or nearby locations.

For evaluation purposes, we use High Definition Video Streams (HDVS) at 30FPS that were captured both for real road traffic and for specially set up a scenario of toy cars imitating traffic patterns on the road. Both the video streams were captured using smartphone cameras. We assume that the computing power of the deployed devices will be similar to a laptop, and the purpose of the processing is to identify vehicles in the video stream. The selected frames are sent to an EC to identify the license plates using the open-source Automated License Plate Reader (ALPR) software [3]. It is challenging to correctly read the license plate because of its fine grain feature. Using license plate recognition accuracy as an evaluation metric, we show that the combination of YLLO and BATS can maintain an object identification accuracy of over 95%, by only sending ~5-10% of all the recorded frames. In the process, we show that our mechanism saves at least 60% energy at the ED. The savings at the EC are expected to be even higher because of the less number of frames sent to ECs that are marked with object IDs, thereby further reducing the work for EC.

The remainder of this paper is organized as below. Section II discusses object identification in a video stream, including the popular YOLOv2 algorithm that forms the basis for our object detection framework-YLLO. Section III presents our proposed YLLO algorithm and compares its performance against that of the original YOLOv2. In section IV, we formulate an object identification optimization problem in a multi-camera environment, and our solution using the proposed BATS algorithm. Experimental evaluation and results are summarized in section V. Related art is briefly discussed in section VI. Section VII concludes the paper including future work.

II. Device Based Object Recognition

Avoiding the transmission of all video frames from edge devices to the EC requires that the device be able to analyze the frames for the presence of "interesting" objects so that the other frames can be either discarded or stored locally and sent later for archival purposes (if necessary). Thus, the critical challenge for edge devices is the real-time recognition of evidence of relevant objects in the video stream.

A. Feature vs. Object Detection

In general, there are two ways to identify frames with interesting objects: (a) look for features that relate to the presence of relevant objects, or (b) directly detect and classify the objects in the stream. From the perspective of real-time operation on low-end devices, option (a) appears quite attractive. We initially explored option (a) using state of the art feature detection methods such as FAST, BRIEF, SIFT, and ORB [4]. Of these, the combined algorithm "Oriented FAST and rotated BRIEF" (ORB) is the most advanced and designed for real-time use.

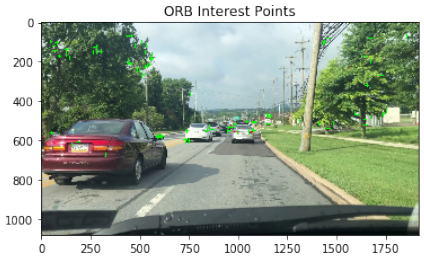


Fig. 2: Keypoint based characterization of a frame (ORB).

Fig. 3: Object based characterization of a frame (YLLO).

Key points are low-level features which do not use a hierarchical layer-wise representation. Fig. 2 shows the key-point characterization of a frame using ORB. The accuracy of key points is affected by the image noise, the similarities between the foreground and the background, and high texture repetition. Under such conditions, it is challenging to achieve discriminative features across the frames using ORB or any other traditional feature detectors. Our experimentation showed that ORB could indeed work reasonably fast (frame rate of 26/sec on a somewhat dated desktop). However, the object identification accuracy of ORB failed to classify objects like cars of different types accurately. Therefore, after considerable experimentation, we abandoned option (a) and focused on (b). Fig. 3 shows the detection of the object with a bounding box, which is of high importance in our research.

An accurate object recognition requires deep learning, such as a convolutional neural network (CNN) with hundreds of layers. The complexity of CNN depends on the variety of objects to be recognized (e.g., vehicles, people, animals, bikes, etc. in the traffic context) and the discrimination desired such as vehicle type (e.g., car, minivan, small truck, etc.), color, and other features. Running such a CNN model on an ED using HDVS at 30 frames/sec requires high computing power and not practical given the constraints (explained above).

We are interested in finding frames if the scene has changed sufficiently for a new object added, to limit the number of frames sent to the EC. Thus a CNN with a smaller number of layers may be useful provided that it does not miss interesting objects or changes to the scene. In particular, we should have a very low false-negative rate for recognizing the objects in the scene, but a higher false-positive rate is acceptable. In addition, the new approach should reliably predict the new

objects and/or the new position of the objects in successive frames, to skip other frames intelligently. We shall discuss in detail these issues after an introduction to the primary approach.

B. Object Detection Algorithms

A variety of algorithms have been proposed in the literature for object detection [5], [6], but most of them are not appropriate for the real-time video-stream use that we are targeting. Table V in [5] supports our above argument and highlights the *low* accuracy and frame rate for various algorithms. The existing object detection algorithms can be classified into two categories: (A) a two-step process that generates region proposals at first and then classifies each proposal into different object categories, and (B) a single-step procedure that regards object detection as a regression or classification problem for the entire frame. The first category includes R-CNN, SPP-net, Fast R-CNN, Faster R-CNN, R-FCN, FPN, and Mask R-CNN. The second category comprises MultiBox, AttentionNet, G-CNN, YOLO, SSD, YOLOv2, DSSD, YOLOv3, DSOD, RetinaNet, etc. For brevity, we do not discuss these; a comprehensive review of most of these is presented by Zhoa [7] and in [8]. Generally, the type (A) methods are inappropriate for real-time use because of their computational costs and hence will not be considered.

TABLE I: Comparison of Various Algorithms [7]

Algorithm	**Accuracy**	**Speed**
Faster R-CNN	73.2	7
SSD300	74.3	46
SSD512	76.8	19
YOLO	63.4	45
YOLOv2 288x288	69.0	91
YOLOv2 416x416	76.8	67
Tiny YOLO 416x416	57.1	207

Table I list the performance of some of the algorithms known for speed and accuracy. For comparison, we also include Faster R-CNN, which is among the fastest algorithms of type (A). Note that the numbers here are for a certain combination of CPU and GPU and on a specific dataset (PASCAL VOC 2007+2012); therefore, *the FPS numbers especially have no meaning beyond the comparisons shown in the Table.* A modification of YOLOv2 [9], called YOLOv3 [2] improves accuracy but at the cost of much higher number of layers (106 vs 19). It is less appropriate for edge device applications because of its computational cost and memory requirements. A lighter version of YOLO, Tiny YOLO still has a higher FPS but its accuracy is compromised. The Single Shot MultiBox Detector (SSD) [10] is rather fast and was specifically designed for mobile devices. However, unlike YOLO, SSD outputs the low resolution version of the classified image which may be unsuitable for sending to EC directly for license plate recognition. We experimented with a few set of videos with SSD (300×300) and the accuracy dropped to 20%, because of the low resolution images processed by the ALPR software. However, for general mobile applications that do not require fine-grain object identification, SSD may be preferable.

C. An Overview of YOLO and SSD

In this section, we provide a brief overview of the two most promising lightweight algorithms, namely YOLOv2 and SSD, mainly from the perspective of their appropriateness for our application. As stated earlier, we use the bounding box, and object detection features of such algorithms, and thus could, in principle, use any single-step algorithm.

YOLOv2 is a structured neural network that has a rather small number of convolutional layers (19) and only 5 max-pooling layers. It divides the image into a grid of size 13x13. Each such grid cell predicts B (typically 5) bounding boxes (BBs) and computes a set of confidence scores for them. Each BB is characterized by five parameters (x, y, h, w, c) where (x, y) is the position of the BB center, (h, w) are the height and the width of the BB, and c is the *box confidence score*. Both the position and BB sides are normalized for the whole image. The box confidence score is the overall confidence of detecting an object in the BB, computed as IOU (intersection-over-union) involving the actual box and the predicted box (see Fig. 4).

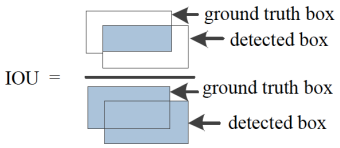


Fig. 4: Calculation of IOU.

YOLOv2 outputs a sequence of "class probabilities", the probability that the objects detected in the grid cell (considering all of the B bounding boxes) and each box has a confidence score which is the probability of an object inside the grid and $\mathbf{C}$ conditional class probabilities, one per class for the likeliness of the 80 object classes [11]. The confidence score is the maximum value of the element-wise product of box confidence and class probabilities. Non-maximum suppression (NMS) is a post-processing technique that ensures that a single bounding box with a high confidence score for each detected object is found among multiple boxes identified for the same object. NMS discards a detection box when the degree of overlap between the detection box and the selected high-scoring detection box exceeds a default threshold value of 0.5.

Most algorithms, including YOLOv2, convert the input frames to a specified resolution before working on them. For YOLOv2, the default internal resolution is 416x416, which is what we have exploited. As shown in Table I, a higher performance can be achieved by reducing the resolution. However, each resolution level requires a separately trained model. For our purposes we use the default resolution and pre-trained models on the PASCAL VOC 2007 [12] and VOC 2012 [13] data sets (also used by results in Table I). The output frame generated by YOLOv2 includes the labels of the detected objects. The resolution of the output frame is 1280x720, which is high enough so that it can be sent directly to the EC for further processing of fine-grain features. In our experiments, we let the EC extract license plate numbers from the detected vehicles by using the ALPR software. It is important to note that although EC could analyze from scratch to extract license plates, the tagging of car objects by the edge devices makes its job substantially simpler and thus energy efficient.

There are numerous variants of YOLOv2/v3 algorithms for

detecting specific types of objects such as cars, pedestrians, etc. [14], [15]. Such specialization is unlikely to be useful in an edge computing environment where a wide variety of plausible objects should be detectable. For example, in the road traffic context, in addition to the vehicles, we need to be able to quickly detect people, animals, bicycles, trucks, tractors, etc.

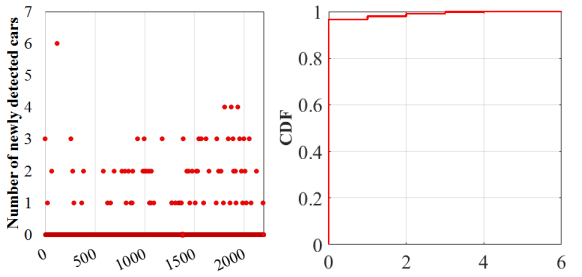
YOLO has difficulty in dealing with small objects in groups and struggles to generalize to objects in new/unusual aspect ratios/configurations. The Single Shot MultiBox Detector (SSD) [10], attempts to address this by discretizing the output space of bounding boxes and fuses predictions from multiple feature maps with different resolutions.

SSD starts with the VGG16 backbone architecture and adds several feature layers to the end of the network, which is responsible for predicting the offsets to default boxes with different scales and aspect ratios and their associated confidences. The network is trained with a weighted sum of localization loss (e.g. Smooth L1) and confidence loss (e.g. Softmax), as explained in [7]. Final detection results are obtained by conducting NMS on multi-scale refined bounding boxes.

III. YLLO: An Energy Efficient Algorithm for Edge Computing

A. Motivation for YLLO Algorithm

As stated earlier, our proposed YOLO algorithm is designed specifically for continuous video-streams, rather than individual snapshots. Thus our focus is only exploiting redundancies in the video-stream to quickly recognize the essential frames for transmission to the edge controller. Although we do use YOLOv2's basic building blocks for this purpose, the mechanism can be applied to any single-step algorithm.



(a) Time series (b) Cumulative distribution function

Fig. 5: New car detection per frame during a rush hour traffic.

Fig. 5 shows the distribution of the number of new cars detected in a video that was captured on the road during a typical rush hour. As seen from this figure, no new objects (vehicles) are detected in most of the frames (~97%). This phenomenon is true for many surveillance applications, including occupancy detection in an office building, or face detection within a cafeteria, etc.

B. Description of YLLO

In the traffic monitoring application, many fast-moving vehicles may enter and exit the scene frequently. YLLO uses the underlying object recognizer to identify the objects and tracks them over the successive frame. By doing object recognition, we not only reduce the computational load on the edge device but also reduce the bandwidth used for frame transmission to the EC. Depending on the application requirements, the untransmitted frames may either be discarded or stored locally by the edge device for some time (e.g., a few hours) and retrieved by the EC as-needed basis.

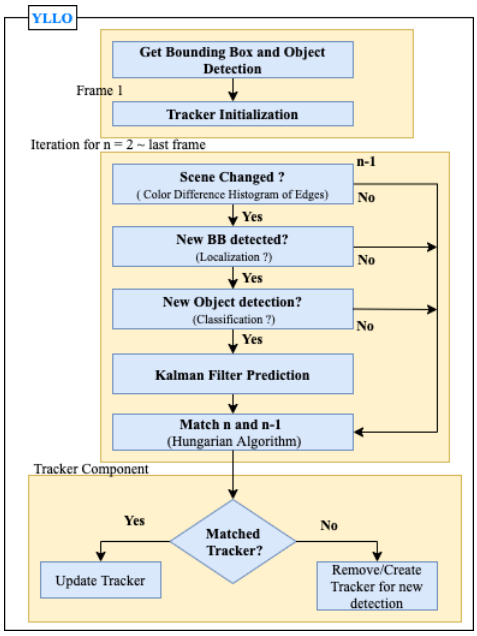


Fig. 6: Structure of YLLO Algorithm.

The overall structure of YLLO is shown in Fig. 6. YLLO algorithm uses the coordinates of the detected object and initializes the object tracker when it receives the first frame. The object tracker associates each detected object between successive frames using the bounding box coordinates. The tracking component uses Kalman Filter [16] and a Hungarian assignment [17] algorithm. The Kalman Filter is a popular filter for building a stable tracking system. It predicts the location of objects for the next frame by using successful detection from the current frame. A multi-object tracking system requires a data assignment process between detected bounding boxes and tracker bounding boxes. The Hungarian algorithm matches the detections by finding IOU (Intersection Over Union) between the detected bounding box and the tracker bounding box. The new objects that are successfully associated with a tracked object will inherit the existing object ID, otherwise a new unique ID is assigned to the object and the tracker is updated. If none of the detected bounding boxes can be associated with the tracked bounding box, the tracker component is deleted Bewley [18]. For the successive frames, YLLO extracts simple feature vector and performs scene change detection by analyzing inter-frame differences. In most videos, a scene changes typically slowly, but occasionally abrupt changes do occur. The scene change detection should consider local (a portion of the region in a frame) and global (the whole frame) scene changes. In our scheme, we combine the edge orientation and histogram-based difference for scene change detection.

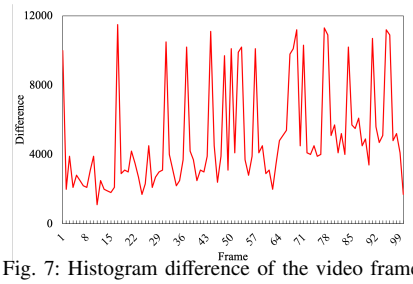


Fig. 7: Histogram difference of the video frames.

Edge orientation can represent the object boundaries and texture structures, thereby providing more semantic information about the object shape than the flat regions. The Laplacian of an image is the second order derivative of pixel intensity values and a high value indicates area of rapid change in density. These second order derivative filters are susceptible to noise. We address the sensitivity issue by smoothing the spatial images before using the Laplacian filter. In particular, we take the approach of combining *Gaussian Smoothing* along with *Laplacian Filter* for edge detection as in Paris [19]. These filters are utilized for identifying sudden intensity changes by locating irregular sharp edges. The Laplacian-Gaussian edge detection framework convolves the images with a mask to detect zero crossings of the calculated second derivatives. The pixels that define local maximum gradient are considered as edges by the edge detector.

To apply this methodology, we generate color difference histograms H_n for edge orientations. Such differences are primarily insensitive to moving cameras, object motions, and changes in illumination. The edge orientations are uniformly quantized into 18 bins, which corresponds to angle intervals of 20 degree (similar to [20]). In our experiments, given a video frame (color image), we quantize the luminance (brightness) channel into 10 bins and the two chrominance channels (U and V components of YUV system) into 3 bins each. The result is a 10×3×3+18 or 108-dimensional color feature vector. The distance metric for measuring the similarity between histograms is the L1 norm defined as:

$$d_n = \sum_{i=1}^{108} \mid H_n(i) - H_{n-1}(i) \mid \quad (1)$$

where H_0(i) = 0 for 0 ≤ i ≤ 108. Finally, the scene change S is calculated using a threshold T of the distance of histograms:

$$S = \{n \mid d_n \geq T, n = 1, 2, 3...N\} \quad (2)$$

where N is the number of frames.

Fig. 7 shows the histogram difference between successive frames. Experimentally, we determined the appropriate scene change detection threshold 'T' to be 8500 over several videos. Figs. 8 shows some of the frames that are sent to the EC because of scene change detection.

C. Comparison of YLLO and YOLOv2

Dataset: We investigated the performance of our YLLO mechanism in terms of speed and accuracy. We evaluated the accuracy and performance of our model to detect vehicles on two data sets. A data set by Longyin Wen [21], which is a real-world multi-object detection and multi-object tracking benchmark, consisting of 10 hours of videos captured at 25 frames per seconds (fps), with resolution of 960 x 540 pixels. This data set has more than 140,000 frames with 8250 vehicles that are manually annotated. Another dataset by Fisher Yu [22] comprising of over 100,000 images with rich annotations such as image level tagging, object bounding boxes, drive-able areas, lane markings, and full-frame instance segmentation.

Ground Truth: In both the datasets, the objects like cars, trucks, bikes, persons, traffic lights etc. were annotated in the video by the authors. Therefore, we assess the performance of our model by comparing results to the ground truth.

Metrics: The primary performance metric for YLLO is detection accuracy. We use detection accuracy as the percentage of correctly predicted examples out of all predictions, formally known as $\frac{TP+TN}{TP+FP+TN+FN}$. The metric parameters are defined as:

- YLLO is said to generate a True Positive (TP) when there is an object and the model detects it with an IOU against ground truth box above the threshold. When multiple boxes detect the same object, the box with the highest IOU is considered as TP. All other boxes are considered as False Positives (FP). We set the IOU threshold to 0.5.
- YLLO is said to generate a False Negative (FN) if the object is present and the predicted box has an IOU < threshold with ground truth box.
- If the object is not in the image, yet the model detects one then the prediction is considered as False Positive (FP).

We considered an additional metric called Multiple Object Tracking Accuracy (MOTA) as defined by Bernadin. [23] to measure accuracy of YLLO. This metric denotes the characteristics of a tracking system in terms of its accuracy in recognizing object configurations like localizing the object and their ability to track objects across the frames consistently. Table II shows the comparison of the performance of YOLOv2 along with our proposed YLLO scheme.

TABLE II: Comparison of Original YOLOv2 with YLLO.

	Detection Algorithm	**Speed (FPS)**	**Detection Accuracy**	**MOTA Accuracy**
Dataset1 [21]	YOLOv2	3-4 FPS	79.7 %	78.45 %
	YLLO	15-16 FPS	84.35 %	86.44 %
Dataset2 [22]	YOLOv2	3-4 FPS	78.06 %	79.3 %
	YLLO	14-15FPS	84.14 %	86.75 %

Compared to YOLOv2 model, YLLO has higher accuracy with the introduction of a scene change detection between successive frames before performing classification, which reduces the overall number of false positives. The experiments for calculating the accuracy tests were run on a small form factor Dell Optiplex 7050 machine, with Intel core i5 (generation 7) processor, using 8 GB memory and 512 GB of HDD. These experiments on a somewhat dated desktop processor are only to provide an idea of how much faster YLLO can run as compared with YOLOv2; the actual performance will depend on the precise deployment, and could well be better than listed here, since GPUs are now being integrated even in mobile and low-end CPU platforms.

On both datasets, YLLO achieves ~3-4× speed as compared to YOLOv2. In particular, YLLO achieves a frame rate of ~14-16 FPS which is near-real time, without compromising the accuracy. Note that this happens on a desktop machine with a

(a) Frame 1 - YLLO (b) Frame 17 - YLLO (c) Frame 31 - YLLO

Fig. 8: Frames for object detection with scene change detected.

somewhat dated processor (generation 7 or KabyLake) without the advantage of any GPU and thus represents capabilities similar to an edge device.

IV. BATS: Bandwidth Adaptive Transmissions of Streams

Next, we discuss the second component of our scheme, namely, an intelligent transmission of the video stream frames that accounts for the available transmission bandwidth. When multiple cameras are covering the same scene, it takes advantage of the redundancy in deciding which frames to transmit. YLLO selectively transmits frames whenever the cameras detect a new object; however, the available bandwidth may not allow transmission of all such frames. For this, we exploit the overlapping coverage areas of multiple cameras to transmit selective frames, while maximizing the level of coverage.

In a multi-camera system, there may be a significant amount of coverage area overlaps in between the cameras. For example, a pair of cameras at both sides of a highway will capture approximately the same number of cars passing the highway. Similarly, two cameras on the same pole on different angles may have overlapping coverage. Thus, if a user query asks to capture all the cars passing the highway within a time interval, then sending frames from one of these cameras may be sufficient. A set of such cameras monitoring a single geographic area with a significant amount of overlapping views form a cluster. BATS provides an *intra-cluster* algorithm, which will only transmit the selected frames after eliminating the redundant/overlapping ones when the wireless channel has limited channel capacity.

A. Determining the primary cameras

We consider a set of cameras as *primary cameras* whose frames are essential for sufficient coverage purposes. For simplicity, we divide the region into several small grids. Next, mark the grids that are covered by these cameras, and the associated weights based on the quality of coverage of the grids by the corresponding cameras. Assume that γ_g is a binary variable if grid-g is covered by at least one camera and zero otherwise. Assume that C_{cg} is a binary variable which is one if camera-c covers grid-g, and the coverage quality of camera-c corresponding to grid point g is q_{cg}. x_c is a binary variable if camera-c is chosen as a primary camera and zero otherwise, whereas w_c is its weight. Thus to ensure that none of the viewpoints are missed, we need to at least accumulate camera-frames such that the amount of coverage (i.e the number of grids covered) is maximized, whereas the number of cameras in the primary set is minimized, while improving the overall quality of coverage. This results in the following optimization problem:

$$\begin{aligned} \text{Maximize} \quad & \sum_g \gamma_g - \alpha \sum_c x_c/w_c + \beta \sum_g Q_g \\ \text{subject to} \quad & \mathbf{C_1}: \gamma_\mathbf{g} \geq \zeta \sum_\mathbf{c} \mathbf{C_{cg} x_c} \\ & \mathbf{C_2}: Q_g = \max_c \; q_{cg} x_c \end{aligned} \tag{3}$$

where α, β, ζ are small numbers. A small α ensures that for two solutions with same coverage, the solution with fewer cameras and higher weights is chosen. When all these are identical, then the solution with better overall quality is chosen, which is ensured by the variable $\beta << \alpha$.

In constraint $\mathbf{C_1}$, the parameter ζ ensures that γ_g is one if grid-g is covered by at least one camera. Because γ_g is binary, it can be at most one irrespective of the number of cameras covering it. The constraints $\mathbf{C_2}$ records the variable Q_g, which is the quality of coverage of a grid-g by at least one of the primary cameras.

It is easy to show that the above problem is NP-Hard. We propose a simple heuristic that flows as follows. We first mark all the grids as *uncovered*. At , the camera with maximum coverage (calculated in terms of the total number of uncovered grids) is chosen to be a primary camera. If multiple cameras have equal coverage, then ties are broken by selecting the camera with (a) higher weight, and (b) better quality of coverage. After choosing the first primary camera, the grids that are covered by the camera is marked as *covered*. Then the same process is repeated for choosing the next set of cameras that maximizes the number of uncovered grids. The iteration stops when adding more cameras does not result in exploring any more uncovered grids.

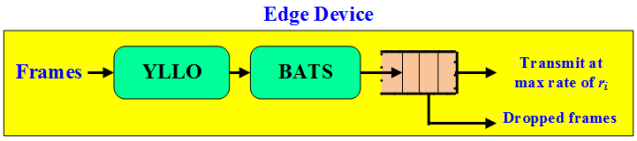


Fig. 9: Illustration of an edge device operations flow.

B. Collaborative rate adaptation in multi-camera system

We now formulate an optimization problem to maximize the amount of information availabile from a cluster at the controller end, with the constraints that (a) the channel capacity is not overshot, and (b) at the same time the controller can receive frames from the cameras at a certain minimum rate. Let us assume that there are C cameras, and the channel capacity is assumed to be $\mathbb{C}$. The controller can dynamically measure the channel capacity by periodically measuring the

packet transmission rate and their corresponding losses. Thus if the time averaged packet loss rate is L and the physical-layer bit rate is R, then $\mathbb{C} = R(1-L)$ [24]

Considering these factors, the **weighted proportional fairness** within a cluster can be achieved by modeling the utility function of node i as $U_i(r_i) = \alpha_i.\log(r_i)$, where α_i is the normalized weight of the window i. The weights may be assigned by the controller based on their locations and area of coverage, orientations and can be time-dependent. Our objective is to maximize the overall utility of the cluster, i.e. $\sum_{i=1}^{N} U_i(r_i)$, after satisfying the required constraints. We also assume that each frame consumes F bps, thus to avoid the channel to be overloaded, i.e. $\sum_{i=1}^{N} r_i.F \leq \mathbb{C}$, or $\sum_{i=1}^{N} r_i \leq \frac{\mathbb{C}}{F} = \mathbb{M}$. Intuitively we can think that the cameras in a cluster work as a *single virtual camera* that reports at a maximum rate of $\mathbb{M}$ frames/interval. $\mathbb{M}$ is a controlling parameter that controls the overall sampling rate of the coalition, i.e. if the controller wants to receive the frames more frequently, it increases $\mathbb{M}$ and vice versa.

Let us denote the set of primary cameras after solving problem (3) is S. Thus each of these cameras need to transmit at a rate of $\mathcal{R}/|S| = \mathbb{S}$ to ensure that any point is tracked at least at a rate of $\mathcal{R}$, i.e. $r_i \geq \mathbb{S}, \forall i \in S$. For all the other cameras can transmit at least at a rate of some minimum rate of R_m. We assume that all the cameras need to transmit at least at a rate of R_m to ensure that the controller can do some analytic in regards to the surveillance. Thus the overall optimization problem becomes:

$$\begin{aligned} \text{Maximize} \quad & \sum_{i=1}^{N} U_i(r_i) \\ \text{subject to} \quad & \mathbf{C_1} : \sum_{i=1}^{N} r_i \leq \mathbb{M}, \\ & \mathbf{C_2} : r_i \geq \mathbb{S}, \forall i \in S, \\ & \mathbf{C_3} : r_i \geq R_m, \forall i \in \bar{S}, \end{aligned} \tag{4}$$

Algorithm 1 Collaborative Multi-Camera Rate Allocation scheme

1: INPUT : Maximum sampling rate $\mathbb{R}_i$, utility weights α_i and $\mathbb{M}$.
2: OUTPUT : Sampling rates r_i $\forall i \in \{1, 2, ..., N\}$.
3: $r_i = \mathbb{S}$, $\forall i \in S$;
4: $r_i = R_m$, $\forall i \in \bar{S}$;
5: $Tot = \sum_i r_i$;
6: $\mathbb{M} = \mathbb{M} - Tot$;
7: **for** each node $i = \{1, 2, ..., N\}$ **do**
8: $\quad r_i = r_i + \frac{\alpha_i}{\sum_{i \in U} \alpha_i} \mathbb{M}$;
9: **end for**
10: return r_i $\forall i$;

We propose an algorithm to solve this problem, which is presented in the Algorithm 1, and is addressed centrally at the edge controller. Algorithm 1 first assigns the minimum required frame transmission rate for each camera (line 3-4), and then the total rate is calculated (line 5). The remaining $\mathbb{M}$ is calculated in line 6. After this step the constraints $\mathbf{C_2}-\mathbf{C_3}$ are satisfied.

We then construct the modified optimization problem after eliminating constraints $\mathbf{C_2}-\mathbf{C_3}$ as follows:

$$\text{Maximize} \sum_{i=1}^{N} U_i(r_i) \quad \text{subject to} \mathbf{C_1} : \sum_{i=1}^{N} r_i \leq \mathbb{M} \tag{5}$$

As log is a concave function, the above problem becomes a convex optimization problem. Thus by solving the KKT conditions of problem (5), we obtain $r_i = \frac{\alpha_i}{\sum_{i \in U} \alpha_i} \mathbb{M}$.

Thus in line 7-9 of Algorithm 1, the remaining $\mathbb{M}$ is then shared proportionately among all the cameras, i.e $r_i = r_i + \frac{\alpha_i}{\sum_{i \in U} \alpha_i} \mathbb{M}$. The calculated sampling rates then sent to the corresponding edge devices.

C. Queue Management

Fig. 9 shows the overall block diagram of the set of steps executed in an edge device. The selected frames sent by the YLLO are stored in an outgoing queue, whose maximum transmission rate is determined by Algorithm 1. Thus, after getting the maximum allowable transmission rate from the controller, the individual edge devices set this as the maximum service rate of their queues. Typically any queue management policy can be adopted; however, for our experiments we have adopted a simple packet dropping policy. When the queue is filled up by a certain threshold Γ, then we drop ξ percentage of equally spaced packets from the queue. We believe that dropping the equally spaced packets is suitable for the streaming applications, rather than dropping some packets in succession, which may hurt the object identification accuracy. For our experiments, Γ and ξ are assumed to be 65% and 20% respectively.

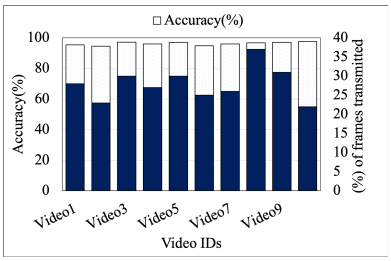
Fig. 10: Example of a frame where ALPR cannot identify one of the license plate **(i.e. red car)** due to lack of clarity.

V. Experimental Evaluation

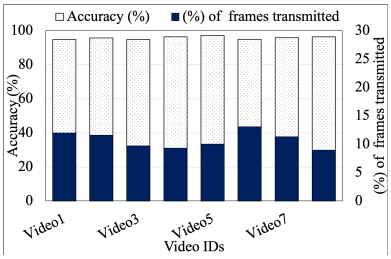
In this section, we evaluate the performance of our schemewith the real video traces. The effectiveness of our schemeis measured using these metrics: (a) object identification accuracy, (b) the fraction of total packets transmitted and, (c) energy consumed. For measuring the object identification accuracy, we pass the frames received at the controller end to the ALPR software and find out the percentage of vehicle nameplates identified correctly for the objects that are identified as true positives by YLLO. Fig. 10 shows an example of a transmitted frame to the EC for detecting the license plates. In this figure, one of the license plates is not identified due to clarity. Unless otherwise mentioned, the frame resolution is kept as 720p (1280×720) which is a recommended resolution level specified by the ALPR software.

A. *Performance of transmitting frames independently*

We first evaluate the performance when each camera transmits the frames independently to the EC. We collected video traces from four different sites around the city of Philadelphia, which consists of residential areas around King of Prussia, interstates near I-76, Center City, and south Philadelphia areas. We have recorded a total of 10 video traces with ~45,000 image frames/video for moving camera and 8 video traces for static camera. The ED connected to the camera transmits the relevant frames to the EC, whenever it detects a new object as described in section III. The EC then passes these frames to the ALPR software to record the vehicle number-plates.



(a) Video captured from car dashboard



(b) Video captured from static cameras

Fig. 11: Vehicle identification accuracy and % of frame transmissions for independent transmissions.

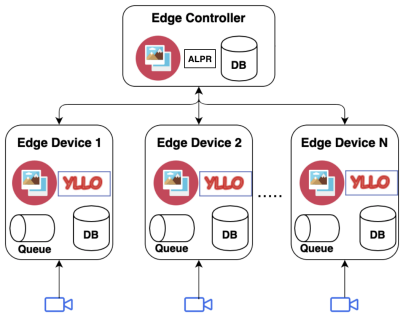


Fig. 12: Experimental setup.

For videos captured using a moving camera (see Fig. 11(a)), the object identification accuracy is more than 95% for all video, and the frame transmitted to EC is around ~20-37%. The inaccuracy of 5% arises mainly because new frames are sent to the EC whenever the cameras detect a new object. This selective transmission can cause some frames to be incomplete or obscure views, thereby hindering the ALPR software incorrectly identifying the nameplate. For video from a static source (see Fig. 11(b)), the scheme achieves the same high accuracy level of around 95% by transmitting only ~5-10% of its captured frames. In both cases, the limited transmission *directly equates to energy saving* in the ED and the EC.

B. *Energy Consumption Analysis*

Energy consumption is a primary concern in video surveillance since video cameras collect a vast amount of data that must be transmitted to the edge controller over the wireless link.A significant part of the energy consumption in a smart camera is due to data transmission and the computation energy of local video frame processing. As part of **additional energy evaluation** of the proposed YLLO algorithm, tests were conducted with multiple HD video streams on two laptops. Machine 1 sports a recent (8th generation) 1.80GHz Intel i7 CPU, 16GB memory, 512GB SSD with Windows-10 OS. Machine2 is older (6th generation) 2.40GHz Intel i5 CPU, 32GB memory, 1TB SSD with Linux Ubuntu 18.4. We compared the test results with energy consumed by YOLOv2 at two frame rates of 30 and 15 FPS. In [25], authors have found that the frame sampling rate independently impacts accuracy. We used a single-phase watt-meter to record the average AC power consumption for both machines (with batteries removed). To minimize influence of other processes, all unnecessary background jobs and services were terminated. The power measurements reported exclude the idle power consumption when the machines are not running our scheme. It also excludes approximately the power consumption of the power adapter itself.Additionally ensuring that the metrics captured closely reflect the power consumption of YLLO only

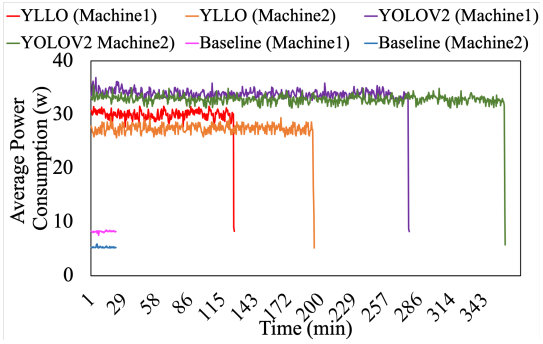
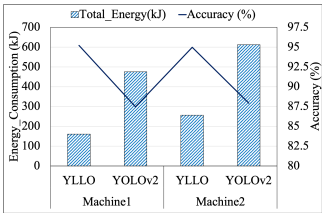
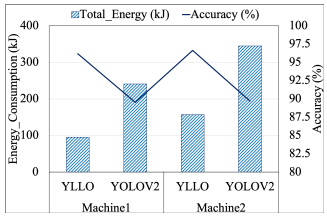


Fig. 13: Average computing power consumption ofor the video traces from moving camera.

1) Energy Consumption Analysis of Video Traces from Moving Camera: Fig. 13 shows the comparison of average power vs. time-taken by our scheme vs YOLOv2. It can be noted that our scheme in both the machines takes less than ~50% the time taken by original YOLOv2. This reduction is mainly due to intelligently avoiding unnecessary frame processing. Yet, the identification accuracy remains over 95% for all video traces, much better than the YOLOv2 accuracy. The reason for the inaccuracy in YOLOv2 is that sending redundant frames to the EC increases the chance of identifying false positives in license plate recognition. As a result, there is a reduction incorrectly identified license plates compared to YLLO. On the other hand, YLLO keeps track of objects with specific object IDs and does not send a frame if the object IDs stay the same for two consecutive frames.

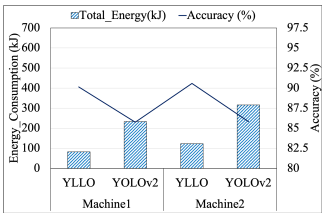


(a) Dashboard video

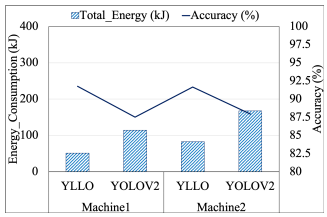


(b) Static Camera Capture

Fig. 14: Energy & accuracy of YLLO vs YOLOv2 at 30FPS.



(a) Dashboard video



(b) Static Camera Capture

Fig. 15: Energy & cccuracy of YLLO vs YOLOv2 at 15FPS.

Fig. 14 shows the energy consumption (in Joules on the left axis) and accuracy (as a percentage on the right axis) of YLLO for 30FPS over different video traces. The comparison is shown for both machine1 and machine2. Also, while Fig. 14(a) is for videos taken from car dashboard on busy streets, Fig. 14(b) is for videos taken from static camera in a residential street. It is seen that *YLLO consumes only about 40% of the energy of YOLOv2, and yet can provide substantially higher accuracy.* The accuracies are 95% (YLLO) vs. 87.5% (YOLOv2) in Fig. 14(a) and increase somewhat [96% (YLLO) vs. 89.5% (YOLOv2)] for both algorithms with statically placed cameras in Fig. 14(b). This is attributed to objects (e.g., buildings, parked cars, trees, etc.) remain static from frame to frame and are easily discounted in the processing.

A similar situation is seen in Fig. 15 where we eliminate alternate frames and thereby run the videos at 15 FPS. Both algorithms suffer accuracy loss in these cases, as expected.However, YLLO still consumes only about 40% of the energy of YOLOv2 and still manages to beat YOLOv2 in accuracy by 5 percentage points (90% vs. about 85%). At both frame rates, machine 1 has lower power consumption due to energy efficient CPU and DRAM, and faster SSD (newer model).

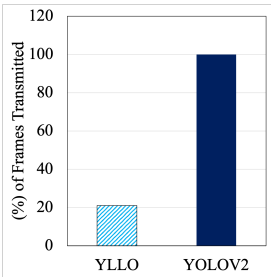


Fig. 16: Percentage of frames transmitted to EC.

The number of network calls to EC (for license plate reading) is another contributor to reduced energy consumption in YLLO. Fig. 16 shows the total network calls by YLLO (to EC) is reduced by 75% because of selective frame transmission (as explained earlier). In contrast, the number of network calls made in YOLOv2 is proportional to the number of frames recorded in the video frames. This reduction in network calls *directly translates* into lower EC computing power requirements and a decrease in network BW usage for communication between ED and EC. *That is, using our schemewill cut down the bandwidth needs to about 1/4th of the original.* This, in turn, can reduce the energy consumption of the device's radio, although the details very much depend on the communications technology used and energy saving aspects such as batching of transmissions, use of low power modes for the transmitter and receiver, etc.

2) Per-Frame Energy Consumption: To better understand the energy results, we further analyzed the energy consumption across various video streams and compared the per-frame energy consumption between YOLOv2 and YLLO. Fig.17 shows this behavior across the 10 dashboard videos, both with 30FPS and 15 FPS (with the line in the middle separating the 30 FPS case from 15 FPS). The graph shows the mean energy in KJ and the vertical bars show the variation across frames of each video. It is seen that the variation in the mean energy across videos is rather limited, and the variation across frames in a video is also quite small. Thus it is meaningful to speak of per-frame energy consumption in that the total energy consumption for a video of this type can be obtained by simply considering the total number of frames. We believe that this also applies to the frame processing in the EC, although we have not conducted detailed experiments directly.

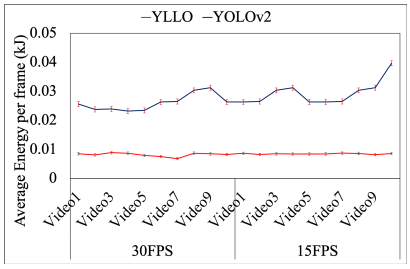


Fig. 17: Mean and variation of per frame energy.

C. Performance of Collaborative Frame Transmissions

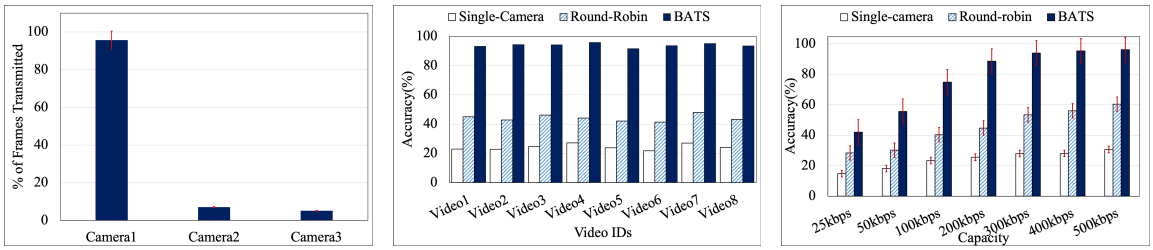
We evaluate the performance in a multi-camera network where there is a significant overlap of coverage areas between the cameras. The purpose of this experiment is to examine to what extent *BATS* can maintain the accuracy as wireless capacity becomes scare, by exploiting the overlaps in between the camera coverage.

We recorded video traces from 3 cameras deployed in our university, with a significant amount of coverage area overlap. Fig. 12 shows the experimental setup; the edge devices connected to the cameras send the frames to the controller for detailed analytics. Fig. 18 shows the views of these 3 cameras showing significant overlaps in their coverage areas. We consider camera-1 as primary camera whose frames are essential for identifying the vehicle nameplates. These cameras log video traces synchronously for around an hour with ~36,000 image frames.

We compare the accuracy of BATS with two approaches: (a) a round-robin approach that cycles through all the cameras within a cluster in a round-robin manner to upload the frames and (b) a single-camera approach that arbitrarily selects a single camera to upload the frames. The average frame size from the cameras is around ~2.5 MB.

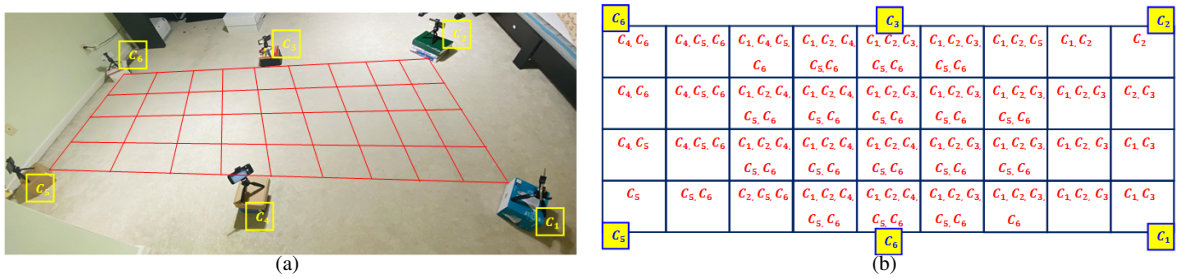
(a) Camera1 (b) Camera2 (c) Camera3

Fig. 18: (a)-(c) Videos captured by 3 cameras with significant overlap in coverage area.



(a) Frac of frames (b) Accuracy vs. Video ID (c) Accuracy vs. Channel capacity

Fig. 19: (a) Fraction of frames transmitted by different cameras, (b) Accuracy and (c) channel capacities of BATS vs. simpler mechanisms.



(a) (b)

Fig. 20: (a) The experimental setup in an indoor environment with six cameras, and their (b) coverage areas.

Fig. 19(a) shows the fraction of frames transmitted by different cameras, which confirms that the primary camera (i.e. camera 1) transmits more than 95% of the frames as compared to the secondary cameras. In Fig. 19(b),we assume that the channel capacity to be equal to 300 kbps. Across all videos (i.e video 1 to video 8) in Fig. 19(b) clearly shows the object detection accuracy of BATS increases by ∼4× compared to the single-camera and ∼2× compared to round-robin schemes.

Next, using different channel capacities (x-axis), we show in Fig. 19(c) the average number-plate identification accuracy (y-axis) of all the videos. These figures show the effectiveness of the collaborative frame transmission policy of BATS as opposed to the independent transmissions. In particular, the accuracy improves significantly as the channel capacity creases from 25 kbps to 300 kbps, but saturates after that.

D. Object Identification Accuracy with Different Frame Resolutions

We evaluate the object detection accuracy of our schemefor different frame resolutions. In Fig. 22, we vary the frame resolution from 1280 × 720 to 854 × 480, which results in an accuracy degradation of 97% to 20%. This is because the performance of the ALPR software degrades significantly with lower resolution. In particular, it goes below 80% with a resolution of less than 920 × 720.

However, frame resolution can be a tuning parameter that the EC can exploit in applications where it only needs to monitor the number of objects passing, rather than doing some deep analytics on the frames like the number-plate identification.

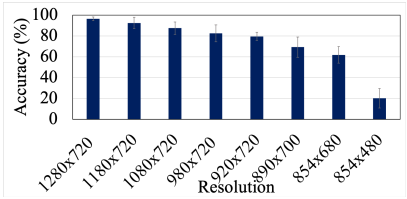
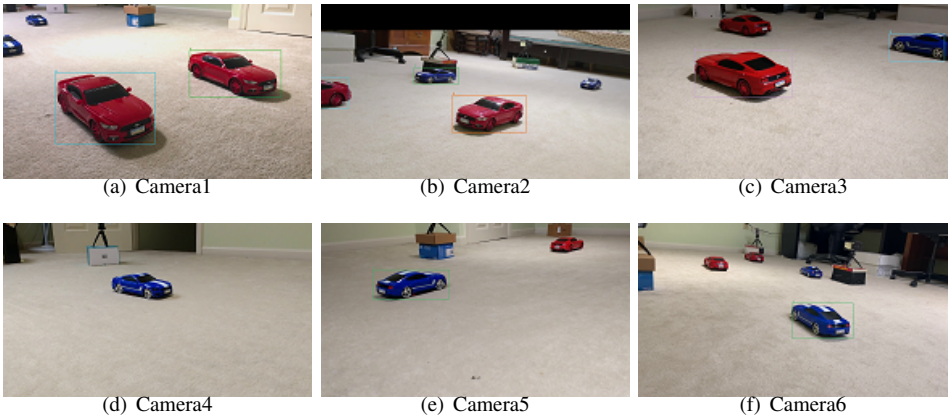


Fig. 22: Object identification accuracy with different frame resolutions.

E. Performance in Multi-Camera Environment

As suitable datasets in a multi-camera environment with partial area overlaps are relatively sparse, we experimented with toy cars setup in an indoor environment. To imitate the scenario of partial area coverage, we place six cameras as shown in Fig. 20(a) as if they are placed by the two sides of a road. We printed the license plates and stick them on both sides of these cars (putting license plates on front and rear end is a requirement in 31 states in USA). This ensured that their license plates could be read by cameras that can see the cars from any side (both front and rear).

We divide the coverage areas into 9×4 grids, and record the area of coverage by these cameras in Fig. 20(b). From Fig. 20(b) we can determine (C_1, C_2, C_5, C_6) as primary cam-



(a) Camera1 (b) Camera2 (c) Camera3

(d) Camera4 (e) Camera5 (f) Camera6

Fig. 21: Videos captured from 6 cameras showing effect of coverage area overlaps.

eras and (C_3, C_4) as secondary cameras. Fig. 21 shows the snapshot of the videos captured by these six cameras at any time instance, which clearly shows the effects of coverage area overlaps that can be exploited for suppressing redundancy.

Fig. 23 shows the fraction of frames transmitted by different cameras, where the channel capacity is assumed to be 200 kbps. It is evident that the primary cameras transmit ~22% of all the frames whereas the secondary ones on transmit a tiny fraction of ~5-6%. This confirms the ability of BATS in redundancy suppression for better channel utilization.

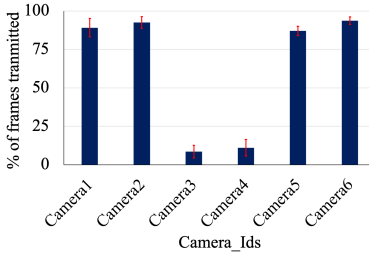


Fig. 23: Percentage of frames transmitted to teh EC.

Fig. 24 shows the accuracy of identifying the license-plates of BATS as opposed to a single camera and round-robin schemes, with varying channel capacities. BATS improves the license-plate detection accuracy by ~4 times as compared to the single camera scenario. As compared to the round-robin scenario, BATS enhances the accuracy by ~65-67%. The accuracy improves till the channel capacity is 300 kbps and saturates beyond that.

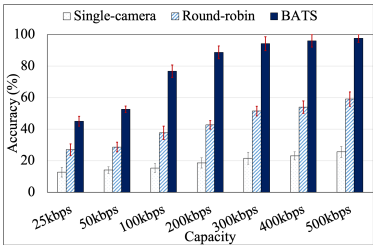


Fig. 24: Accuracy of our schemeagainst simpler mechanisms.

This reduction in the number of frames can also provide corresponding energy savings in the radio transmitters if suitable power saving mechanisms are available and engaged. For example, IEEE 802.11ah includes power saving mode using the Target Wake Time (TWT) mechanism [26], [27] that permits the transmitter to enter into a sleep state occasionally. During their sleep state, the incoming packets can be buffered and transmitted when the radios wake up. There is an energy-latency tradeoff here which needs to be adjusted depending on the application requirements.

VI. Related Work

Applying CNNs to object classification has been showing excellent performance improvements recently. In [28],authors used CNNs to detect the objects that demonstrate exceptional performance gains in real-time. Reference [29] compares the speed and accuracy of various CNN models. In [30] the authors have developed a cognitive assistance framework using Google Glass along with edge computing to provide satisfactory performance. A framework by Glimpse [31] analyzes data from multiple sources and presents a case study of filtering unnecessary frames for face detection. Wang [32] suggested an application using smartphone cameras to increase pedestrian safety. This application detects vehicles approaching the pedestrian using the rear camera and alerts them. CarSafe [33] presents an Android application that detects driver fatigue using the front-facing camera along with tracking road conditions using a rear-face camera. Live video analytics using edge computing has been increasingly popular in the research community. Ananthanarayanan [34] has discussed the feasibility of distributed edge computing architecture for meeting the real-time requirement of large-scale video analytics. Videostorm [35] is a video analytics platform that performs query processing on live video streams, optimizing both the query knobs and resource allocation. Similar to our work, Chameleon [25] presents a video analytics system that optimizes the inference accuracy and computational resources by exploiting the temporal and spatial correlation among video frames. Optasia [36] writes and process video queries in SQL, however, it does not address query processing across distributed machines. Deep-Decision [37] is a measurement-driven framework making smart decisions to choose the appropriate deep learning model under varying network conditions. Unlike our work, their model does not discuss object tracking mechanism across the video frames. VideoEdge [38] uses edge-based video analytics to maintain higher accuracy in real-world video queries over hierarchical structures by placing and merging the queries for processing. However, it does not discuss the reduction in bandwidth required when uploading the frames to the cloud for further analysis. Vigil [24] and LAVEA [39] also explored edge-based real-time video analytics system by partitioning the analytics between the edge and the cloud. In [40], [41] the authors have

proposed a roadmap for scaling video analytics using cross-camera correlations. Similar proposals of edge computing for video surveillance are studied in [42], [43], [44], [45]. Our scheme advances the state-of-art by performing a light-weight object detection scheme locally at the edge devices; thus, avoid transmitting redundant frames to the remote controller, whereas reserves the deep analytics work at the controller end.

VII. Conclusions and Future Work

Managing and exploiting the streaming video data efficiently for surveillance applications is likely to be the defining characteristic of edge computing. The critical issue is the large amount of raw data generated by IoTs, which must be processed and correlated in real-time by energy-constrained ED and ultimately by EC that too is energy constrained due to limitations of cost, space, cooling, etc. In this paper, we develop a lightweight object detection technique called YLLO that significantly reduces the number of frame transmissions to the ECs while still maintaining the object identification accuracy of more than 95%. In addition, our BATS scheme can further reduce the frame rate by exploiting coverage overlaps among multiple cameras under the control of the same EC. In future, we plan to take advantage of the redundancy across multiple ECs that occurs because similar objects are likely to be seen by cameras hosting on adjacent ECs. We will also further enhance the speed and energy efficiency of the YLLO algorithm by exploiting regularity conditions in a video stream.

References

[1] N. Abbas *et al.*, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2018.

[2] J. Redmon *et al.*, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[3] "Opensource Automatic License Plate Recognition," http://www.openalpr.com/cloud-api.html.

[4] E. Salahat *et al.*, "Recent advances in features extraction and description algorithms: A comprehensive survey," in *IEEE ICIT*, 2017, pp. 1059–1063.

[5] J. Han *et al.*, "Advanced deep-learning techniques for salient and category-specific object detection: A survey," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 84–100, Jan 2018.

[6] L. Weng, "Object detection part 4: Fast detection models," https://lilianweng.github.io/lil-log/2018/12/27/object-detection-part-4.html, Dec 2018.

[7] Z.-Q. Zhao *et al.*, "Object detection with deep learning: A review," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[8] Y. Li *et al.*, "Tiny-dsod: Lightweight object detection for resource-restricted usages," *arXiv preprint arXiv:1807.11013*, 2018.

[9] J. Redmon *et al.*, "Yolo9000: better, faster, stronger," in *IEEE CVPR*, 2017, pp. 7263–7271.

[10] W. Liu *et al.*, "SSD: single shot multibox detector," *CoRR*, vol. abs/1512.02325, 2015.

[11] T.-Y. Lin *et al.*, "Microsoft coco: Common objects in context," in *European conference on computer vision*, 2014, pp. 740–755.

[12] M. Everingham *et al.*, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results," http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[13] M. Everingham *et al.*, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[14] M. Putra *et al.*, "Convolutional neural network for person and car detection using yolo framework," *Journal of Telecommunication, Electronic and Computer Engineering*, vol. 10, no. 1-7, pp. 67–71, 2018.

[15] M. J. Shafiee *et al.*, "Fast yolo: a fast you only look once system for real-time embedded object detection in video," *arXiv preprint arXiv:1709.05943*, 2017.

[16] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[17] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[18] A. Bewley *et al.*, "Simple online and realtime tracking," in *IEEE ICIP*, 2016, pp. 3464–3468.

[19] S. Paris *et al.*, "Local laplacian filters: Edge-aware image processing with a laplacian pyramid." *ACM Trans. Graph.*, vol. 30, no. 4, p. 68, 2011.

[20] G.-H. Liu *et al.*, "Content-based image retrieval using color difference histogram," *Pattern recognition*, vol. 46, no. 1, pp. 188–198, 2013.

[21] L. Wen *et al.*, "Ua-detrac: A new benchmark and protocol for multi-object detection and tracking," *arXiv CoRR*, vol. abs/1511.04136, 2015.

[22] F. Yu *et al.*, "Bdd100k: A diverse driving video database with scalable annotation tooling," *arXiv preprint arXiv:1805.04687*, 2018.

[23] K. Bernardin *et al.*, "Multiple object tracking performance metrics and evaluation in a smart room environment," in *IEEE International Workshop on Visual Surveillance*, vol. 90, 2006, p. 91.

[24] T. Zhang *et al.*, "The design and implementation of a wireless video surveillance system," in *ACM MobiCom*, 2015, pp. 426–438.

[25] J. Jiang *et al.*, "Chameleon: scalable adaptation of video analytics," in *ACM SIGCOMM*, 2018, pp. 253–266.

[26] E. M. Khorov *et al.*, "A tutorial on IEEE 802.11ax high efficiency wlans," *IEEE Commun. Surv. Tutorials*, vol. 21, no. 1, pp. 197–216, 2019.

[27] E. M. Khorov *et al.*, "A survey on IEEE 802.11ah: An enabling networking technology for smart cities," *Comput. Commun.*, vol. 58, pp. 53–69, 2015.

[28] A. Krizhevsky *et al.*, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[29] J. Huang *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," in *IEEE CVPR*, 2017, pp. 7310–7311.

[30] K. Ha *et al.*, "Towards wearable cognitive assistance," in *ACM Mobisys*, 2014, pp. 68–81.

[31] S. Han *et al.*, "Glimpsedata: Towards continuous vision-based personal analytics," in *ACM WPA*, 2014, pp. 31–36.

[32] T. Wang *et al.*, "Walksafe: a pedestrian safety app for mobile phone users who walk and talk while crossing roads," in *ACM HotMobile*, 2012.

[33] C.-W. You *et al.*, "Carsafe app: Alerting drowsy and distracted drivers using dual cameras on smartphones," in *ACM Mobisys*, 2013, pp. 13–26.

[34] G. Ananthanarayanan *et al.*, "Real-time video analytics: The killer app for edge computing," *IEEE Computer*, vol. 50, no. 10, pp. 58–67, 2017.

[35] H. Zhang *et al.*, "Live video analytics at scale with approximation and delay-tolerance," in *USENIX NSDI*, 2017, pp. 377–392.

[36] Y. Lu *et al.*, "Optasia: A relational platform for efficient large-scale video analytics," in *ACM SoCC*, 2016, pp. 57–70.

[37] X. Ran *et al.*, "Deepdecision: A mobile deep learning framework for edge video analytics," in *IEEE Infocom*, 2018, pp. 1421–1429.

[38] C.-C. Hung *et al.*, "Videoedge: Processing camera streams using hierarchical clusters," in *IEEE/ACM SEC*, 2018, pp. 115–131.

[39] S. Yi *et al.*, "LAVEA: latency-aware video analytics on edge computing platform," in *ACM/IEEE SEC*, 2017, pp. 15:1–15:13.

[40] S. Jain *et al.*, "Scaling video analytics systems to large camera deployments," in *HotMobile*, 2019, pp. 9–14.

[41] A. Misra *et al.*, "Dependable machine intelligence at the tactical edge," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, T. Pham, Ed., vol. 11006, International Society for Optics and Photonics. SPIE, 2019, pp. 64 – 77.

[42] S. Y. Nikouei *et al.*, "Smart surveillance as an edge network service: From harr-cascade, SVM to a lightweight CNN," in *IEEE CIC*, 2018, pp. 256–265.

[43] J. Wang *et al.*, "Elastic urban video surveillance system using edge computing," in *SmartIoT*, 2017, pp. 7:1–7:6.

[44] J. Barthelemy *et al.*, "Edge-computing video analytics for real-time traffic monitoring in a smart city," *Sensors*, vol. 19, no. 9, p. 2048, 2019.

[45] H. Sun *et al.*, "*VU*: video usefulness and its application in large-scale video surveillance systems: an early experience," in *SmartIoT*, 2017, pp. 6:1–6:6.