MULTI-VIEW SUMMARIZATION AND ACTIVITY RECOGNITION MEET EDGE COMPUTING IN IOT ENVIRONMENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-view video summarization (MVS) lacks researchers attention due to their major challenges of inter-view correlations and overlapping of cameras. Most of the prior MVS works are offline, relying on only summary, needing extra communication bandwidth and transmission time with no focus on uncertain environments. Different from the existing methods, we propose edge intelligence based MVS and spatio-temporal features based activity recognition for IoT environments. We segment the multi-view videos on each slave device over edge into shots using light-weight CNN object detection model and compute mutual information among them to generate summary. Our system does not rely on summary only but encode and transmit it to a master device with neural computing stick (NCS) for intelligently computing inter-view correlations and efficiently recognizing activities, thereby saving computation resources, communication bandwidth, and transmission time. Experiments report an increase of 0.4 in F-measure score on MVS Office dataset as well as 0.2% and 2% increase in activity recognition accuracy over UCF-50 and YouTube 11 datasets, respectively, with lower storage and transmission time compared to state-of-the-art. The time complexity is decreased from 1.23 to 0.45 secs for a single frame processing, thereby generating 0.75 secs faster MVS. Furthermore, we made a new dataset by synthetically adding fog to an MVS dataset to show the adaptability of our system for both certain and uncertain surveillance environments.

1 INTRODUCTION

Surveillance cameras installed indoor and outdoor at offices, public places, and roads generate huge amount of video data on daily basis. This gigantic volume of data has two big issues: first one is storage consumption and second is huge computational complexity for its purposeful usage (Xu et al., 2019). Video summarization aims at these problems by condensing the data size via extracting key information from lengthy videos and suppressing the redundant frames. A video summary generated from a single camera is called single-view video summarization (SVS) (Mahasseni et al., 2017). On the other hand, a summary generated from a camera network is known as MVS (Panda et al., 2016a). SVS is intensively researched with applications to various domains including surveillance (Wang et al., 2017), sports (Tejero-de Pablos et al., 2018), and news videos (Wang et al., 2018). In contrast, MVS is not studied deeply because of several challenges such as computing inter- and intra-view correlations, overlapping regions among connected cameras, and variation in light conditions among different views. The basic flow of MVS includes input acquisition, preprocessing, feature extraction, post-processing, and summary generation. The mainstream MVS methods follow traditional machine learning approaches such as clustering along with low-level features extracted from entire frame with no focus on specific targets in surveillance.

The most important part of MVS is considering different objects in surveillance that can be useful for summary generation. However, the existing techniques do not focus on objects such as persons and vehicles while generating summary. Thus, the final summary may miss some important frames having persons or vehicles that need to be considered for MVS. Furthermore, all the existing techniques rely only on MVS with no further steps for analysis of the generated summary. For instance, the generated summary can be used for indexing, browsing, and activity recognition. The existing

methods are functional only in certain environments with no focus on uncertain scenarios (Min et al., 2019), making them inadequate in real-world environments. Finally, all the existing methods process data on local/online servers or personal computers with huge computation power. It requires extra processing time, power of transmission, and does not guarantee quick responsive action for any abnormal situations, if not handled on the edge. To ensure proper and quick responsive arrangements, activity recognition at edge is a necessary requirement of the current technological era. Activity recognition literature is mature, but with no focus on processing over the edge. Almost all the existing techniques classify activities over high computational local or cloud servers. Classifying activity on edge is an important task of surveillance in smart cities. Therefore, to tackle these challenges effectively, we present a novel framework applicable in both certain and uncertain environments for MVS and activity recognition over the edge.

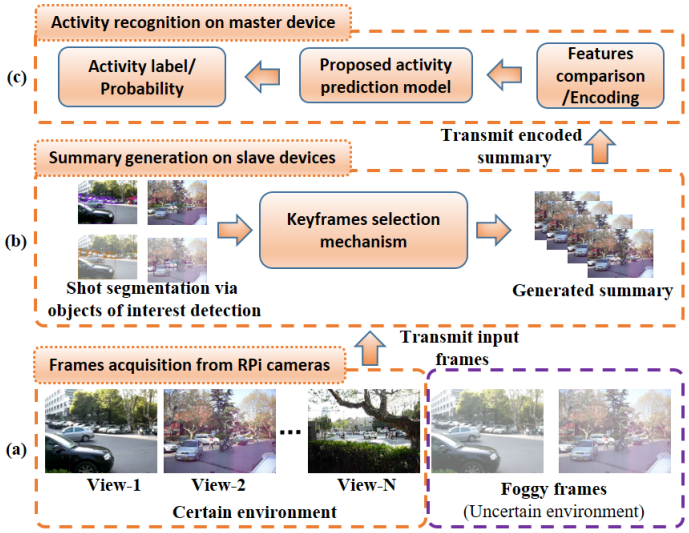

Figure 1: Input and output flow of our proposed framework. (a) Video frames (both certain and uncertain environment) from resource constrained devices. (b) Annotate frames by detecting objects of interest, apply keyframes selection mechanism, generate summary, encode and transmit it to master device. (c) Decode generated summary, perform features extraction, and forward it to activity prediction model at master device to get the output class with probability score.

The problems aimed in this paper are different from the schemes presented in existing literature. We integrated two different domains including MVS and activity recognition under the umbrella of a unified framework in an IoT environment. We presented interconnected resource constrained IoT devices working together to achieve several targets i.e., object detection, summary generation, and activity recognition as shown in Figure 1. The overall framework consists of numerous slaves and a master resource constrained device connected through a common wireless sensor network (WSN). The slave devices are equipped with a camera to capture multi-view video data, segment it into shots, generate summary, encode a sequence of keyframes, and transmit it to the master device. The master device is equipped with an INTEL Movidius NCS to classify the ongoing activity in the acquired sequence. INTEL Movidius is a modular and deep learning accelerator in a standard USB 3.0 stick. It has a Vision Processing Unit (VPU) that is functional with ultra-low power and better performance. It enables activity recognition with significantly lower power, storage, and computational cost. Further, a widely used concept of temporal point processing (Xiao et al., 2019) is utilized for activity classification, ensuring an effective recognition model. While addressing the problems in MVS and activity recognition over resource constrained devices, we made the following contributions.

- Employing an algorithm for MVS on resource constrained devices, reducing the time complexity compared to existing approaches with higher accuracy. The generated summary is further utilized to recognize the underlying activity of all the views through an auto-encoder and learned spatio-temporal features followed by different variants of SVM classifiers to demonstrate the efficiency and effectiveness of our proposed framework.

- Adding uncertainties such as fog to an outdoor MVS benchmark dataset to demonstrate the working of proposed framework in any type of scenario and introduce a new trend in MVS literature for researchers.
- The presented framework has high-level adaptability with special care for the capacity and traffic of WSN. It has many flavors with tradeoff among transmission time, quality of keyframes, and accuracy of activity recognition model with computationally different classifiers.

In the subsequent sections, Section 2 provides a literature review and Section 3 explains the presented framework in detail. In Section 4, experimental results for MVS and activity recognition are given, and Section 5 concludes the overall paper with future directions.

2 Related Work

This section is mainly divided into two sub-sections, covering the representative studies of MVS and activity recognition related to our work. The literature of MVS was initiated almost a decade ago in 2010 by (Fu et al., 2010; Li & Merialdo, 2010) with the first MVS indoor dataset. The employed MVS methods are computationally complex but they are not suitable for resource constrained devices. In contrast to MVS, activity recognition is a richer research field with a variety of techniques focusing on different applications.

2.1 Multi-view video summarization literature

Majority of the MVS methods are based on handcrafted-features integrated with traditional machine learning approaches for final summary generation. The initial MVS schemes (Fu et al., 2010; Li & Merialdo, 2010) utilized features such as SIFT descriptors, Gaussian and Laplacian difference, and motion features followed by statistical learning approaches such as K-means or other clustering techniques for summary generation. The next trend of MVS (Muramatsu et al., 2014; Ou et al., 2014b) used the same features by adding pre- or post-processing like background subtraction along with supervised learning such as SVM or unsupervised learning. The final summary in this trend is either uniform length or of length provided by user. The next MVS trend (Kuanar et al., 2015; Ou et al., 2014a) utilized mid-level features, motion-based shot boundary detection, spatiotemporal graphs, and low-level features such as color and edge histograms. The final summary is generated through the same statistical learning-based techniques. A breakthrough in MVS field is noticed after the usage of learned features in prerequisite steps for generating summaries. This trend is followed in 2016 (Panda et al., 2016a), where BVLC CaffeNet (Jia et al., 2014) and Spatiotemporal C3D (Tran et al., 2015) features are used for sparse coding and video representation, respectively. Besides clustering, template matching and sparse representative selection over learned embedding are used for generating final summaries. Panda et al. (Panda & Roy-Chowdhury, 2017) also used C3D features for video representation by computing inter- and intra-view similarities through sparse coefficients and summary generation using clustering.

2.2 Activity recognition literature

The recent success of deep learning based methods in activity recognition achieved high-level accuracies. These methods extract features from final layers of various deep learning models and apply sequential learning for activity recognition (Ramasinghe & Rodrigo, 2015; Liu et al., 2016). For instance, (Ullah et al., 2017) used features extracted from a pre-trained AlexNet, followed by bi-directional LSTM for human action recognition with better results. Following this, (Ullah et al., 2018) explored optical flow convolutional features with multi-layer LSTM for activity recognition. Their approach outperformed (Ullah et al., 2019) and other state-of-the-art, however, its huge running time restricts its adoptability in real-world surveillance networks. Similar to the previous methods, (Simonyan & Zisserman, 2014) used different types of CNN architectures and configurations for human action representation, considering both stationary and non-stationary environments. Till date, the activity recognition methods do not perform computation over resource constrained devices with significant accuracy. Thus, to tackle this problem effectively, we optimized VGG-19 model (Simonyan & Zisserman, 2014) to make it functional over resource constrained devices with details given in Section 3.

3 PROPOSED FRAMEWORK

The overall framework is divided into two major modules based on the functionalities of the resource constrained devices. The first module is related to multi-view video acquisition, shot segmentation, and summary generation. The second module performs computationally expensive processes involving deep features extraction, features comparison and encoding, and activity prediction. The whole framework contains a number of resource constrained devices (master and slave) connected to a WSN in IoT. This section provides details of the proposed system with major steps are given in appendix section A and visualized in Figure 2.

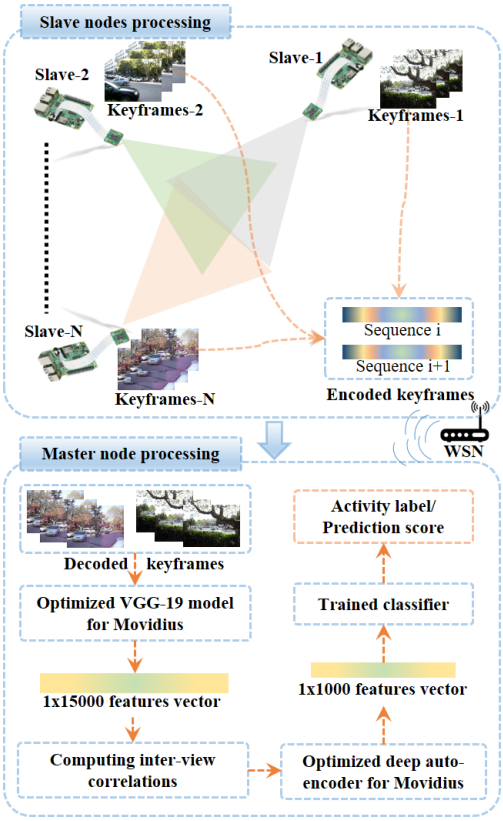

Figure 2: The conceptual diagram of our proposed framework, where slave devices with camera capture multi-view video data, apply shot segmentation, extract keyframes, encode and transmit them to master device for computing inter-view correlations and activity recognition.

3.1 OBJECTS OF INTEREST DETECTION FOR SHOT SEGMENTATION

Surveillance analysts are usually interested in activities of humans and vehicles; thus we consider them as objects of our interest. For this purpose, a tiny version of YOLO CNN model [25] is employed, which is much faster and has higher accuracy compared to other detectors. This tiny object detector works well in normal scenarios but it has poor performance when videos contain fog. To overcome this limitation, we used the pre-trained weights of YOLO tiny model and retrained them over foggy data as show in experimental results section 3. Fog is applied on the entire image globally without targeting our objects of interest only. Adding fog globally has an advantage of better learning, because in real-life surveillance scenarios, the fog is always observed throughout the image. The foggy data is generated from the COCO dataset (Redmon & Farhadi, 2018) by adding synthetic fog to all the images using MATLAB script with different levels of fog, ranging from 0 to 255. We conducted experiments with different values of fog parameter (f_p) as shown in experimental section and selected $f_p = 220$ as optimal. The process of foggy data creation is given in Algorithm A in appendix section.

Training data for object detection specifically in foggy/normal environment has an additional advantage of reliability. Most of the available multi-view datasets are related to indoor such as Office and Bl-7f. Thus, we conducted experiments on the available multi-view outdoor datasets only, such as Road (Fu et al., 2010; Li & Merialdo, 2010). The trained model is used over each slave device for object detection. The attached camera of device transmits frames to our customized object detection model. The frames having persons or vehicles with confidence score greater than 0.9 are stored and considered for further steps while the remaining frames are discarded.

3.2 KEYFRAMES SELECTION MECHANISM

A summary is generated from the segmented frames with objects, because events occur due to their interaction. Therefore, we considered those shots that can possibly contain events such as sitting on a chair and entering a room etc. in MVS datasets. These events are possible only due to human presence. Due to this reason, we segmented those shots in which humans are detected. The shots with desired objects are larger in number, containing redundant frames and thus cannot be considered in the final summary. To avoid the redundancy and provide users a very compact representation, we compute mutual information between features of consecutive frames. This process is executed over slave devices, having limited execution resources, thus, we used low-level features intelligently. Further, instead of using only Euclidean or any other distance measuring method, we compared mutual information between two consecutive feature vectors to avoid redundancy and obtain better results. The feature descriptor is acquired from each frame by computing the ORB points (Rublee et al., 2011). The frames with persons, having exactly the same information, are discarded and only a single frame is selected from them. The selected frame is considered as a keyframe on the concerned resource constrained device. The formula used for mutual information is given in Eq. 1.

$$MI(f_p, f_n) = \sum_{i=1}^{f_p} \sum_{j=1}^{f_n} \frac{|f_p(i) \bigcap f_n(j)))|}{N} log \frac{N|f_{p(}i) \bigcap f_{n(}j)|}{|f_p||f_n|} \tag{1}$$

Here f_p is the previous frame while f_n is the current frame and N is the size of feature vector. The mutual information is computed using scikit-learn library in Python.

3.3 ACTIVITY RECOGNITION

Human activity recognition is an important task in surveillance video analysis. An activity is a sequence of movements in continuous video frames, therefore, it requires extraction of spatiotemporal features. In our system, the activity needs to be recognized on master device after getting summary from the slave device. For activity recognition, we used a pre-trained VGG-19 CNN model for spatial feature representations followed by auto-encoder which captures the temporal features in the sequence. In contrast to VGG-19, the well-known AlexNet (Krizhevsky et al., 2012) and GoogleNet (Szegedy et al., 2015) are not effective to capture tiny patterns in visual data because these models have 1111 and 77 filter size with 4 and 2-pixel stride, respectively. We also made experiments on MobileNetV2 (Sandler et al., 2018), where we extracted features from final convolution layer, but the results were not convincing compared to state-of-the-art (see Table 2). VGG-16 model is recently studied for activity recognition problem in (Ullah et al., 2019) and we outperformed their method over various datasets by investigating VGG-19. It can extract discriminative visual features because it uses 33 filter in all convolutional layers with fixed 1-pixel stride. This helps in processing the smallest receptive field to grab the notions of tiny patterns. It has sixteen convolutional and three fully connected layers where we extract deep features from the final fully connected layer (FC8) having 11000 dimension. FC7 layer of VGG-19 outputs 4096 dimension representations which result a large sized feature vector for a single sequence and yield huge processing for encoding of temporal representations. Thus, it cannot be considered for activity recognition over resource constrained devices in real-time scenarios. We are processing 15 fps, therefore, we extract 15000 spatial frame-level features from one second summarized frames. The features extracted in a single interval of time are compared to compute inter-view correlations. The features in the same interval from different views, are considered as overlapped and hence a single feature vector is selected from them for further processing.

Our feature extraction is performed on Movidius VPU stick, therefore, the original VGG-19 model is compressed using neural computing stick software development kit (NCSDK). This optimized the original model of size 574.7 MB to 287.3 MB to Movidiuss compatible graph format. The temporal changes and sequential patterns in 15000 features are learned using our proposed auto-encoder. Motivated from the wide usage of auto-encoder network for many applications (Yang et al., 2019), we utilized it in our framework. Auto-encoder reduces the dimensionality by analyzing patterns in high dimensional data, therefore, we claim that it can learn the spatio-temporal patterns in deep features of 15 consecutive frames. We performed several experiments for squeezing 15000 features to low dimensions to effectively learn the temporal changes. We first encoded 15000 features to 8000, followed by its encoding to 1000 features (first setting). This kind of setting is very effective because its mean square error (MSE) for encoded features is very low, providing precise accuracy. However, its time complexity is not meeting the needs of embedded devices because the total squeezing has one extra step of encoding to 8000 features. In another setting, we encoded the 15000 high dimensional features directly to 1000 (second setting). In this setting, the accuracy is slightly compromised when using linear SVM, however, we met the computational requirements of embedded devices. We trained our auto-encoders for 40 epochs in which we utilized sparsity regularization with proportion value of 0.1 and L2 weights regularization to reduce the chance of over-fitting and avoid being stuck in local minima. To this end, MSE with support of sparsity regularization and L2 regularization is trained as a cost function of our auto-encoders. In first setting, the MSE decreased to 0.012 after 40 epochs, while in the second setting, it reduced to 0.044. The encoded features are spatiotemporal representation of an activity, which are passed to SVM classifier for activity recognition. We trained one versus all multi-class SVM because it trains N-1 (N is the number of total classes) classifiers, which is efficient for resource restricted devices. All SVMs are examined for activity recognition problem including linear, quadratic, and cubic SVM.

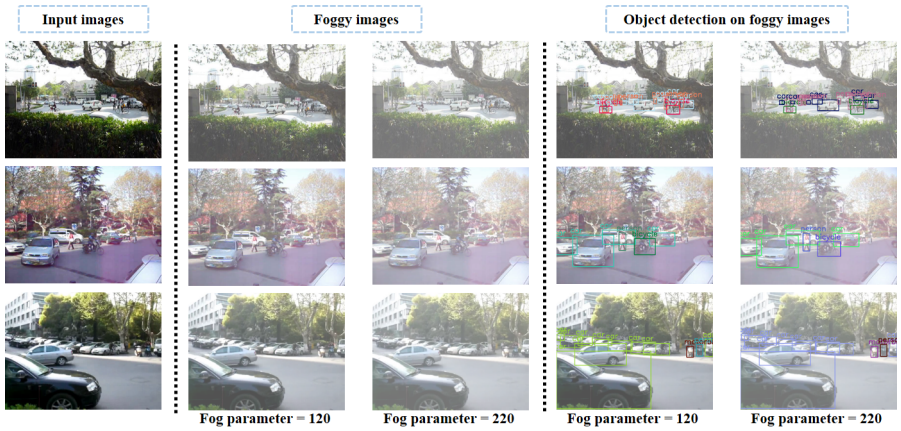

Figure 3: Sample results from road dataset. (a) Normal input images, (b) fog applied with different parameters, (c) object detection in foggy images. The detection results are encouraging for object detection in foggy environment.

3.4 MALLEABILITY OF PROPOSED FRAMEWORK

We designed this framework considering the major limitations of WSNs and IoT devices such as computational complexity and storage capacities. We also aimed to provide an independent activity analysis system, which is malleable in both certain and uncertain environments. For this purpose, we first investigated different MVS and activity recognition methods, and studied the capabilities of WSNs. Concluding our literature study, the need of an independent system is realized, which could capture multi-view video data, analyze it, and provide a compact summary with recognized activities in lengthy surveillance videos. To this end, we made extensive experiments with different configurations and parameters, which can be interchangeably used according to users requirements and available resources. For instance, a slave node with extensively limited resources in WSN can encode its summary with PNG compression before its transmission to master node for activity recognition. With this mechanism, a real-time summary generation and transmission to master node is

ensured despite the slower connectivity and limited communication bandwidth. Similarly, at master node, different options can be considered to balance execution time and accuracy of activity recognition. Various classifiers were used for experiments with different computational complexities. In addition, different experiments were performed for direct feature extraction from a sample summary as well as encoded features with different configurations for activity recognition. The complete evaluation details of these configurations are discussed in Section 4.

4 EXPERIMENTAL RESULTS

We divide the experimental evaluation of our proposed framework into three subsections: MVS, activity recognition, and statistical analysis of data transmission. First, we explain the performance evaluation of MVS employed on resource constrained devices and compared with state-of-the-art techniques. Next, we evaluated the activity recognition module that is carried out on master device with Movidius setup. In the MVS module, we used Raspberry Pi (RPi) for our experiments, but our work is not limited to only RPi and can be applied to any other device with embedded vision. For activity recognition, we used MATLAB as a simulation tool and then transformed it to Keras deep learning framework in Python. The model trained using Keras with Tensorflow in backend can be converted into Movidius compatible graph format using NCSDK. Currently, we tested our model on RPi that can be adjustably used over other Movidius-supported devices. Sample results of integrated framework for an input video from UCF-50 dataset with different views are visualized in Figure 4.

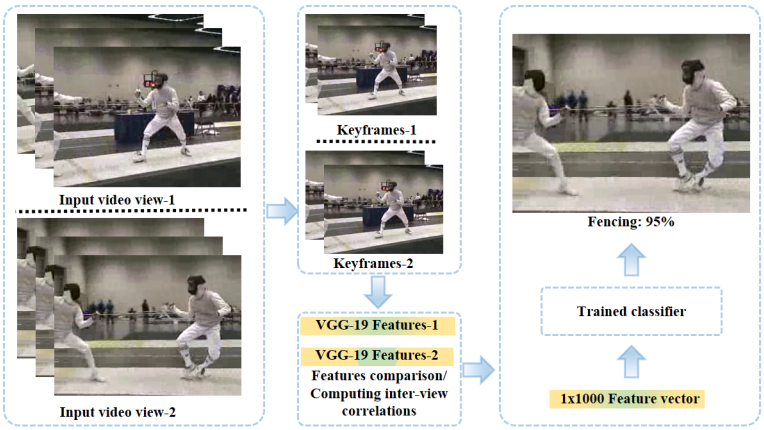

Figure 4: Sample results generated on UCF-50 video captured from two different views. Keyframes from both input videos are generated and then VGG-19 features are extracted. These features are compared and the selected single feature vector is encoded into 11000. Final output of activity is generated by passing this feature vector to trained model that gives probability score along with the predicted class.

4.1 DATASETS

We used five different datasets for experiments where three of them are utilized for object detection in foggy scenarios and MVS and rest of the them are used to evaluate our activity recognition method. Office (Fu et al., 2010; Li & Merialdo, 2010) is widely used for performance evaluation of MVS methods because of its public availability along with the ground truth. It is created by utilizing four stably-held cameras fitted in an office environment. The cameras have motion with no synchronization between the captured videos with light variations. Bl-7f (Ou et al., 2014a) is the most challenging dataset in MVS literature that has 19 different cameras installed at 7th floor of Taiwan University. The recorded videos contain different events closely related to each other with maximum overlapping. The installed cameras are synchronized, still, and fixed but with different light conditions in different views. The ground truth as well as the dataset is publicly available for research purposes. Road (Fu et al., 2010; Li & Merialdo, 2010) multi-view dataset is captured using handheld cameras with extreme level of shuddering effects and variable light conditions. Different

from the above two datasets, it contains persons and vehicles, recorded in an outdoor day light. This dataset is used for foggy outdoor videos to test our object detection model in uncertain environment. Results of object detection over Road dataset are given in Figure 3 in uncertain environment. The activity recognition part of our framework is evaluated using UCF50 (Reddy & Shah, 2013) and YouTube 11 (Liu et al., 2009) action datasets. UCF50 contains assorted human action videos with high-level shuddering due to camera motion, difference in viewpoint, and scalability of objects. The total number of classes are 50 with some actions closely related to each other, effecting the overall accuracy of classifiers. YouTube dataset, in contrast to UCF50 is less challenging and includes motion of camera, cluttered background, light variation, and changes in viewpoint. A total of 11 different action videos are available in this dataset.

Table 1: Performance comparison of the proposed framework with state-of-the-art. Office and Bl-7f datasets are used for comparison. The highest F1 score among all the methods is made bold. P refers to Precision and R represents Recall score.

Methods	Office			Bl-7f			Execution time per frame (secs)
	P	R	F1 Score	P	R	F1 score	
(Fu et al., 2010)	1.00	0.61	0.75	-	-	-	-
(Ou et al., 2014b)	0.41	0.63	0.50	-	-	-	-
(Kuanar et al., 2015)	1.00	0.69	0.81	0.75	0.98	0.85	-
(Ou et al., 2014a)	0.25	0.75	0.40	0.58	0.61	0.6	-
(Panda et al., 2016b)	1.00	0.73	0.81	-	-	-	-
(Panda et al., 2016a)	1.00	0.70	0.81	-	-	-	-
(Panda & Roy-Chowdhury, 2017)	1.00	0.81	0.89	-	-	-	-
(Hussain et al., 2019b)	0.93	0.86	0.90	-	-	-	-
(Hussain et al., 2019a)	0.94	0.89	0.91	-	-	-	1.23
Proposed	0.94	0.92	**0.93**	0.85	0.88	**0.85**	0.45

4.2 MVS EVALUATION

Table 1 shows the MVS results of our implemented framework compared to other state-of-the-art techniques. We provide objective evaluation for two benchmark datasets in MVS literature i.e., Office and Bl-7f. The time complexity of our framework is given in the last column, indicating the execution time for a RPi device. The better performance of our method compared to existing methods can be seen from Table 1. On Office dataset, our method achieved the highest F-measure score compared to all methods under consideration. The most recent methods [38] for MVS on Office dataset scores 0.90 and 0.91 with higher computational complexity. The method presented in [38] uses heavy-weight CNN model for spatio-temporal point processing to select video skims. A gap in computational complexity from a recent MVS method over resource constrained devices is indicated in Table I where our method generates MVS 0.78 secs faster than (Hussain et al., 2019a). In case of Bl-7f dataset, we outperformed (Ou et al., 2014a) and have similar F1-score with (Kuanar et al., 2015), but our framework is functional over resource constrained devices.

4.3 ACTIVITY RECOGNITION EVALUATION

The activity recognition module of our system is evaluated using two benchmark action datasets including UCF-50 and YouTube 11. The comparison using an overall accuracy metric with state-of-the-art techniques is given in Table 2. We conducted different experiments for activity recognition using original VGG-19 features, encoded features with S1, and encoded features with S2 along with variants of SVM including linear, quadratic, and cubic SVM. S1 in Table 2 represents the first setting, where the sequential features are encoded into 8000, followed by 1000 from 15000 features of a single sequence. S2 presents the second setting of directly encoding 15000 features to a 1000 feature vector. These different settings can be alternatively used for different scenarios, considering both accuracy and complexity. On UCF50 dataset, trajectories analysis (Peng et al., 2016), hierarchical clustering (Liu et al., 2016), ML-LSTM (Ullah et al., 2018), and deep auto-encoder with CNN (Ullah et al., 2019) achieved 92.3%, 93.2%, 94.9%, and 96.4% accuracy, repetitively. On the other hand, our S1 and S2 with cubic SVM achieved 96.2% and 93.8% overall accuracy, respectively. On YouTube 11 action dataset, the single stream CNN (Ramasinghe & Rodrigo, 2015), hierarchical clustering (Liu et al., 2016), DB-LSTM (Ullah et al., 2017), ML-LSTM (Ullah et al., 2018), and

Table 2: Detailed comparative analysis of various settings of the proposed activity recognition against recent activity recognition methods.

Method	**Features/Learning**	**Overall accuracy (%)**	
		UCF-50	YouTube actions
Single stream CNN **(Ramasinghe & Rodrigo, 2015)**	A CNN network for motion and static information in a stream	-	93.1
Trajectories analysis **(Peng et al., 2016)**	Bag of visual words and their fusion	92.3	-
Hierarchical clustering **(Liu et al., 2016)**	Hierarchical clustering multi-task learning	93.2	89.7
DB-LSTM **(Ullah et al., 2017)**	AlexNet, bi-directional LSTM	-	92.84
ML-LSTM **(Ullah et al., 2018)**	Optical flow convolutional features, LSTM	94.9	95.8
Deep auto-encoder and CNN **(Ullah et al., 2019)**	VGG-16, deep auto-encoder	96.4	96.21
MobileNetV2 (Linear SVM)	Final layer features (1x1000)	33.3	35.6
Proposed (Linear SVM)	VGG-19 (1x15000)	92.6	93.9
	S1 (1x1000)	86.4	84.6
	S2 (1x1000)	80.8	90.6
Proposed (Quadratic SVM)	VGG-19 (1x15000)	94.9	98.5
	S1 (1x1000)	92.2	89.1
	S2 (1x1000)	94.4	94.7
Proposed (Cubic SVM)	VGG-19 (1x15000)	**96.6**	**98.7**
	S1 (1x1000)	96.2	91.3
	S2 (1x1000)	96.1	95.6

auto-encoder with CNN (Ullah et al., 2019) achieved 93.1%, 89.7%, 92.84%, 95.8%, and 96.21% accuracy, respectively. On the other hand, the proposed method with full VGG-19 features and cubic SVM reached 98.7% accuracy, S1 reached 89.1% on quadratic and 91.3% on cubic SVM, and S_2 achieved 94.7% and 95.6% on quadratic and cubic SVM, respectively. From the above comparisons, it can be observed that the accuracy of our different settings are either higher or similar to the state-of-the-art. However, in terms of computational complexity, the existing methods are designed for high processing GPUs but our system is functional over resource constrained devices. The performance of our settings with linear SVM is under 80% but with quadric SVM, it touches the milestone of 90%. Therefore, it can be considered as state-of-the-art results for processing over embedded devices with low cost VPU stick. Finally, the experiments performed over MobileNetV2 give lower accuracy on both the datasets. Thus, light-weight MobileNetV2 is not a suitable option for consideration in real-time surveillance for activity recognition.

4.4 Statistical evaluation of WSN

Wireless networks have constrained resources of bandwidth and communication cost. It is therefore not a feasible solution to transmit huge-sized surveillance videos to cloud or local servers due to limited transmission and storage of data. Also, searching for an event in long videos manually is a tiresome task. This section provides statistical details about saving storage capacity, bandwidth, and transmission time as illustrated in Figure 5.

4.4.1 Storage capacity

To the best of our knowledge, there is no specific dataset available to confirm the efficiency and effectiveness of our framework in terms of bandwidth, transmission time, and storage capacity. Thus, we considered an example video to generate keyframes, encode and transmit them to the master device for recognizing activities. We then compared all the mentioned parameters as given in Figure 5. The frames in the example video are considered as non-compressed with ideal situation of transmission. First, we calculate the total number of pixels by multiplying height (h) of the frame with width (w), no_of_pixels = hw. In office video, h = 480 and w = 640, resulting in 307,200, which is multiplied by the depth of the frame i.e., 8. Finally, we divided the obtained number by 8 to get the size of the frame in bytes and is further converted to MB. The size of a single frame from this video is 0.29 MB. Similarly, to calculate the size of the office video-0, multiply the size of each frame by the number of frames inside it. This video has 26,940 frames, and multiplying it with size of

each frame, we get 7892.57. It can be observed that there is a huge difference of storage, indicating the effectiveness of our system in terms of saving storage capacity. Saving storage directly refers to preservation of communication bandwidth because the bandwidth consumed by data is the same as their size.

4.4.2 TRANSMISSION TIME

The transmission time in our framework is saved in two ways: 1) in low-bandwidth networks: encode the keyframes with PNG compression, and 2) transmit only important frames over IoT network saving huge time. Suppose an ideal situation in the network to transmit a frame in the WSN from slave to master device. Assume the distance as 0.3 km and speed as 200,000 km/s with data rate of 32 Mbps. To receive the data from slave (source), the data will take 3 types of times: 1) getting frame on WSN, 2) transmitting the data through the network, 3) loading data from router/hub to the master (destination) device. The time denoted as t1 in Eq. 2 is transmission time from source to network router. Transmission time t_2 in Eq. 3 is considered as the time taken by network to transmit data from source router to destination. The transmission time from destination router to the master device is the same as t1. The overall transmission time is the sum of t_1, t_2, and t_3, where t_1 and t_3 are same. The calculated transmission time for overall frames, keyframes, and encoded keyframes is 4137.98, 4.45, 1.13 seconds, respectively, as given in Figure 5.

$$t_1 = \frac{data\_size}{data\_rate} \tag{2}$$

$$t_2 = \frac{distance\_from\_slave\_to\_master\_device}{transmission\_speed} \tag{3}$$

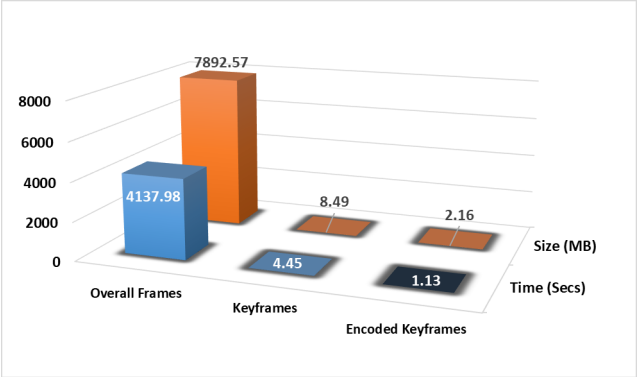

Figure 5: Transmission time and storage size comparison of transmitting and storing overall video (Office-0 video) versus keyframes and encoded keyframes. A huge difference of saving transmission time and storage capacity can be observed among all these possible settings.

5 CONCLUSION

In this paper, we integrated MVS and activity recognition under an umbrella of a unified framework. A complete setup including slaves and a master resource constrained device working independently in an IoT is presented. The hardware requirements include slave devices equipped with camera and wireless sensors, a master device with Intel Movidius NCS for running optimized deep learning models on the edge. The slave devices capture multi-view video data, detect objects, extract features, compute mutual information, and finally generate summary. The generated summary is received at master device with optimized trained model for activity recognition. The MVS algorithm as well activity recognition models presented in this paper outperform state-of-the-art.

In future, we have intention to extend this work by deeply investigating multi-view action recognition algorithms with different parameters and configurations in resource constrained environments. Further, we want to explore spiking neural networks used for various tasks [40, 41] in our framework for spatio-temporal features extraction advanced to activity recognition.

REFERENCES

Yanwei Fu, Yanwen Guo, Yanshu Zhu, Feng Liu, Chuanming Song, and Zhi-Hua Zhou. Multi-view video summarization. *IEEE Transactions on Multimedia*, 12(7):717–729, 2010.

Tanveer Hussain, Khan Muhammad, Javier Del Ser, Sung Wook Baik, and Victor Hugo C de Albuquerque. Intelligent embedded vision for summarization of multi-view videos in iiot. *IEEE Transactions on Industrial Informatics*, 2019a.

Tanveer Hussain, Khan Muhammad, Amin Ullah, Zehong Cao, Sung Wook Baik, and Victor Hugo C de Albuquerque. Cloud-assisted multi-view video summarization using cnn and bi-directional lstm. *IEEE Transactions on Industrial Informatics*, 2019b.

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 675–678. ACM, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Sanjay K Kuanar, Kunal B Ranga, and Ananda S Chowdhury. Multi-view video summarization using bipartite matching constrained optimum-path forest clustering. *IEEE Transactions on Multimedia*, 17(8):1166–1173, 2015.

Yingbo Li and Bernard Merialdo. Multi-video summarization based on video-mmr. In *11th International Workshop on Image Analysis for Multimedia Interactive Services WIAMIS 10*, pp. 1–4. IEEE, 2010.

An-An Liu, Yu-Ting Su, Wei-Zhi Nie, and Mohan Kankanhalli. Hierarchical clustering multi-task learning for joint human action grouping and recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(1):102–114, 2016.

Jingen Liu, Jiebo Luo, and Mubarak Shah. Recognizing realistic actions from videos in the wild. Citeseer, 2009.

Behrooz Mahasseni, Michael Lam, and Sinisa Todorovic. Unsupervised video summarization with adversarial lstm networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 202–211, 2017.

Xiongkuo Min, Guangtao Zhai, Ke Gu, Yucheng Zhu, Jiantao Zhou, Guodong Guo, Xiaokang Yang, Xinping Guan, and Wenjun Zhang. Quality evaluation of image dehazing methods using synthetic hazy images. *IEEE Transactions on Multimedia*, 2019.

Yuki Muramatsu, Takatsugu Hirayama, and Kenji Mase. Video generation method based on user's tendency of viewpoint selection for multi-view video contents. In *Proceedings of the 5th Augmented Human International Conference*, pp. 1. ACM, 2014.

Shun-Hsing Ou, Chia-Han Lee, V Srinivasa Somayazulu, Yen-Kuang Chen, and Shao-Yi Chien. On-line multi-view video summarization for wireless video sensor network. *IEEE Journal of Selected Topics in Signal Processing*, 9(1):165–179, 2014a.

Shun-Hsing Ou, Yu-Chen Lu, Jui-Pin Wang, Shao-Yi Chien, Shou-De Lin, Mi-Yen Yeti, Chia-Han Lee, Phillip B Gibbons, V Srinivasa Somayazulu, and Yen-Kuang Chen. Communication-efficient multi-view keyframe extraction in distributed video sensors. In *2014 IEEE Visual Communications and Image Processing Conference*, pp. 13–16. IEEE, 2014b.

Rameswar Panda and Amit K Roy-Chowdhury. Multi-view surveillance video summarization via joint embedding and sparse optimization. *IEEE Transactions on Multimedia*, 19(9):2010–2021, 2017.

Rameswar Panda, Abir Das, and Amit K Roy-Chowdhury. Embedded sparse coding for summarizing multi-view videos. In *2016 IEEE international conference on image processing (ICIP)*, pp. 191–195. IEEE, 2016a.

Rameswar Panda, Abir Dasy, and Amit K Roy-Chowdhury. Video summarization in a multi-view camera network. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 2971–2976. IEEE, 2016b.

Xiaojiang Peng, Limin Wang, Xingxing Wang, and Yu Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *Computer Vision and Image Understanding*, 150:109–125, 2016.

Sameera Ramasinghe and Ranga Rodrigo. Action recognition by single stream convolutional neural networks: An approach using combined motion and static information. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pp. 101–105. IEEE, 2015.

Kishore K Reddy and Mubarak Shah. Recognizing 50 human action categories of web videos. *Machine Vision and Applications*, 24(5):971–981, 2013.

Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R Bradski. Orb: An efficient alternative to sift or surf. In *ICCV*, volume 11, pp. 2. Citeseer, 2011.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Antonio Tejero-de Pablos, Yuta Nakashima, Tomokazu Sato, Naokazu Yokoya, Marko Linna, and Esa Rahtu. Summarization of user-generated sports video by using deep action recognition features. *IEEE Transactions on Multimedia*, 20(8):2000–2011, 2018.

Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 4489–4497, 2015.

Amin Ullah, Jamil Ahmad, Khan Muhammad, Muhammad Sajjad, and Sung Wook Baik. Action recognition in video sequences using deep bi-directional lstm with cnn features. *IEEE Access*, 6: 1155–1166, 2017.

Amin Ullah, Khan Muhammad, Javier Del Ser, Sung Wook Baik, and Victor Albuquerque. Activity recognition using temporal optical flow convolutional features and multi-layer lstm. *IEEE Transactions on Industrial Electronics*, 2018.

Amin Ullah, Khan Muhammad, Ijaz Ul Haq, and Sung Wook Baik. Action recognition using optimized deep autoencoder and cnn for surveillance data streams of non-stationary environments. *Future Generation Computer Systems*, 96:386–397, 2019.

Han Wang, Huangyue Yu, Pei Chen, Rui Hua, Chuyi Yan, and Ling Zou. Unsupervised video highlight extraction via query-related deep transfer. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pp. 2971–2976. IEEE, 2018.

Wenzhong Wang, Qiaoqiao Zhang, Bin Luo, Jin Tang, Rui Ruan, and Chenglong Li. Selecting attentive frames from visually coherent video chunks for surveillance video summarization. In *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 2408–2412. IEEE, 2017.

Shuai Xiao, Junchi Yan, Mehrdad Farajtabar, Le Song, Xiaokang Yang, and Hongyuan Zha. Learning time series associated event sequences with recurrent point process networks. *IEEE transactions on neural networks and learning systems*, 2019.

Zengmin Xu, Ruimin Hu, Jun Chen, Chen Chen, Junjun Jiang, Jiaofen Li, and Hongyang Li. Semisupervised discriminant multimanifold analysis for action recognition. *IEEE transactions on neural networks and learning systems*, 2019.

Xu Yang, Cheng Deng, Feng Zheng, Junchi Yan, and Wei Liu. Deep spectral clustering using dual autoencoder network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4066–4075, 2019.

A APPENDIX

Algorithm 1 Proposed algorithm for MVS and activity recognition.

Parameters

C_n = Camera-n, M_o = Object detection model, $M_{vgg\text{-}19}$ = VGG-19 model, AE = auto-encoder, F = input frame, F_a = annotated frame, S_c = current shot, KF_e = encoded keyframes, KF_s = Selected keyframes, F_d = decoded sequence, F_{15000} = 15000 dimensional features, and F_{000} = 1000 dimensional features.

1: **Initialization**: C_1, C_2, … C_n, M_o, $M_{vgg\text{-}19}$, AE, SVM.

2: **Procedure** Slave device
3: F ? read frame
4: F_a ? M_o (F)
5: **Do while** F_a has target objects
6: S_c ? Store F_a
7: **End while**
8: Extract ORB features from each frame of S_c
9: Compute mutual information among features of S_c
10: Select KF_s from S_c
11: KF_e ? encode the KF_s
12: Transmit KF_e to Master device
13: **Repeat** Step 3 for next shot

14: **Procedure** Master device
15: F_d ? decode KF_e
16: $F_{15000} = M_{vgg\text{-}19}$ (F_d)
17: Compute F_{15000} from all the views for current interval
18: Select non-redundant features among all views
19: F_{1000} ? AE (F_{15000})
20: Final prediction ? SVM (F_{1000})
21: **Output:** Final prediction

Algorithm 2 Foggy data generation algorithm

1: **input**: Directory of input images, fog parameter = fp
2: N ? length (directory)
3: **for** K = 1 **to** N **do** > loop through folder
4: I_c ? read_image(K) > read single image
5: M_1 ? max(max(I_c)) > find maximum value1
6: I_r ? I_c + fp > add fog parameter
7: M_2 ? max(max(I_r)) > find maximum value2
8: $I_o = \frac{(I_r \times M_1)}{M_2}$ > normalize foggy image
9: save I_o > save foggy image
10: **end for**