Multi-View Summarization and Activity Recognition Meet Edge Computing in IoT Environments

Tanveer Hussain, *Student Member*, *IEEE*, Khan Muhammad, *Member, IEEE*, Amin Ullah, *Student Member*, *IEEE*, Javier Del Ser, *Senior Member, IEEE*, Amir H. Gandomi, *Senior Member*, *IEEE*, Muhammad Sajjad, Sung Wook Baik, *Member, IEEE, and Victor Hugo C. de Albuquerque, Senior Member, IEEE*

***Abstract*—Multi-view video summarization (MVS) has not received much attention from the research community due to inter-view correlations and views' overlapping, etc. The majority of previous MVS works offline, relying on only summary, and require additional communication bandwidth and transmission time, with no focus on foggy environments. We propose an edge intelligence-based MVS and activity recognition framework that combines artificial intelligence with Internet of Things (IoT) devices. In our framework, resource-constrained devices with cameras use a lightweight CNN-based object detection model to segment multi-view videos into shots, followed by mutual information computation that helps in summary generation. Our system does not rely solely on a summary, but encodes and transmits it to a master device using a neural computing stick for inter-view correlations computation and efficient activity recognition, an approach which saves computation resources, communication bandwidth, and transmission time. Experiments show an increase of 0.4 unit in F-measure on an MVS Office dataset and 0.2% and 2% improved accuracy for UCF-50 and YouTube 11 datasets, respectively, with lower storage and transmission times. The processing time is reduced from 1.23 to 0.45 seconds for a single frame and optimally 0.75 seconds faster MVS. A new dataset is constructed by synthetically adding fog to an MVS dataset to show the adaptability of our system for both certain and uncertain IoT surveillance environments.**

***Index Terms*—Activity Recognition, Deep Autoencoder, Deep Learning, IoT, Multi-view Video Summarization, Sequential Learning, Video Data Analytics, Video Summarization.**

I. Introduction

Surveillance cameras installed both indoors and outdoors, for example in offices, public places, and on roads, generate a large amount of video data on a daily basis. There are two significant issues associated with these gigantic volumes of data: the need for storage and the demand to process that data for future use, which involves huge time complexities. Video summarization aims to solve these problems and reduces the data size by extracting key information from lengthy videos and suppressing redundant frames. A video summary generated by a single camera is called a single-view video summarization (SVS) [1], whereas a summary generated from a camera network is called multi-view video summarisation (MVS) [2]. SVS has been intensely researched for applications in various domains, including surveillance [3], sports [4], and news videos [5]. In contrast, MVS has not been extensively studied due to the presence of challenges such as the computation of inter- and intra-view correlations, overlapping regions among connected cameras, and the variation in light conditions among different views. The basic flow of the MVS process includes input acquisition, pre-processing, feature extraction, post-processing, and summary generation. Mainstream MVS methods use traditional machine learning approaches such as clustering, combined with low-level feature extraction from an entire frame without focusing on specific targets in surveillance videos.

The most important aspect of MVS is to consider different surveillance objects that may be useful for summary generation. However, existing techniques do not focus on objects such as persons and vehicles when generating a summary, and thus the final summary may miss important frames containing persons or vehicles. Furthermore, all existing techniques rely only on MVS, and do not apply

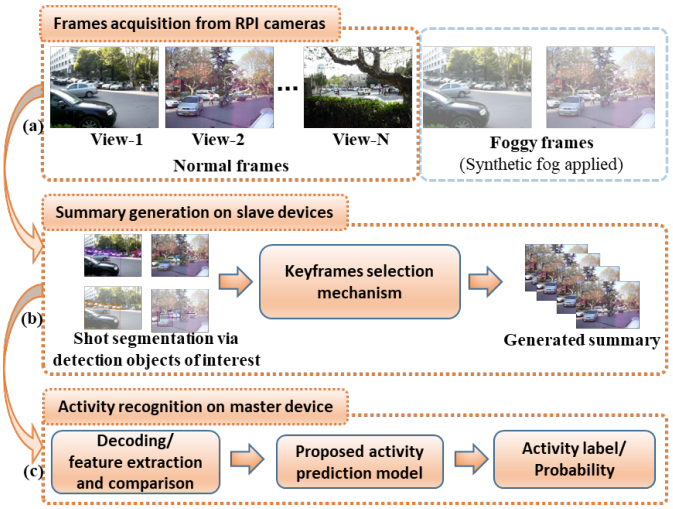

Figure 1: Input and output flow of the proposed framework: (a) input multi-view video frames (both foggy and non-foggy); (b) annotate frames on slave devices by detecting objects of interest, apply keyframes selection mechanism, generate summary, encode and transmit it to master device; (c) decode the generated summary, perform features extraction and comparison to computer inter-view correlations, and forward to the activity prediction model on the master device to obtain the output class with a probability score.

Manuscript received March 30, 2020; Accepted: XXX, Published: XXXX. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2019R1A2B5B01070067). This paper was recommended by Associate Editor XYZ. (*Corresponding author: Sung Wook Baik*)

Tanveer Hussain, Aminullah, and Sung Wook Baik are with Intelligent Media Laboratory, Digital Contents Research Institute, Sejong University, Seoul 143-747, South Korea (Email: tanveer445@ieee.org, aminullah@ieee.org, and sbaik@sejong.ac.kr)

Khan Muhammad is with the Department of Software, Sejong University, Seoul 143-747, South Korea. (Email: khan.muhammad@ieee.org)

Javier Del Ser is with TECNALIA, 48160 Derio, Spain, University of the Basque Country (UPV/EHU), 48013 Bilbao, Spain, and Basque Center for Applied Mathematics (BCAM), 48009 Bilbao, Spain (Email: javier.delser@tecnalia.com)

Amir H. Gandomi is with the Faculty of Engineering & Information Technology, University of Technology Sydney, Australia (e-mail: gandomi@uts.edu.au)

Muhammad Sajjad is with Department of Computer Science, Islamia College Peshawar, Pakistan (Muhammad.sjjad@icp.edu.pk)

Victor Hugo C. de Albuquerque is with University of Fortaleza, Brazil (e-mail: victor.albuquerque@unifor.br).

additional analysis to the generated summary. For instance, the generated summary can be used for indexing, browsing, and activity recognition. On top of this, existing methods are functional only in certain environments, with no focus on foggy scenarios, which make them inadequate for use in real-world environments. Finally, all existing methods process data in local/online cloud analysis centres or on personal computers with high computational power. This requires extra processing time, high transmission power, and does not guarantee quick responsive action in abnormal situations if not handled at the edge. To ensure appropriate and quick responsive countermeasures, activity recognition at the edge is a requirement of current smart surveillance systems. Although the activity recognition literature is mature, researchers have not generally focused on processing at the edge. Almost all of the existing techniques classify activities using cloud analysis centres with high computational power [6]. Classifying activity at the edge is an important task for real-time monitoring of smart cities. Therefore, to effectively address these challenges, we present a novel framework for MVS and activity recognition at the edge that is applicable to both normal and foggy environments.

The problems addressed in this paper are different from the schemes considered in the existing literature. In this work, we integrate two different domains (MVS and activity recognition) under the umbrella of a unified framework in an IoT environment. An IoT network contains a number of interrelated computing devices, sensors, objects, animals, etc. with unique identification, and functions as part of a shared network that has the ability to transfer data and information without human involvement. In this paper, we present interconnected resource-constrained IoT devices that work together to accomplish several tasks, such as 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 vision sensors to capture multi-view video data, segment it into shots, generate a summary, encode a sequence of keyframes, and transmit it to the master device. The master device is equipped with an INTEL Movidius neural computing stick (NCS) [7], which classifies the ongoing activity in the acquired sequence. The INTEL Movidius is a modular deep learning accelerator on a standard USB 3.0 stick. It has a vision processing unit (VPU) that functions with ultra-low power and better performance as compared to normal central processing unit. It enables activity recognition with significantly lower power, storage, and computational costs. This paper addresses the MVS and activity recognition related problems over resource-constrained devices and makes the following major contributions.

- We employ an algorithm for MVS on resource-constrained devices, which reduces the time complexity and gives higher accuracy than existing approaches.
- To demonstrate the efficiency and effectiveness of our proposed framework, we use the generated summary to recognise the underlying activity in all the views through an autoencoder and learned spatiotemporal features followed by different variants of SVM classifiers.
- We add uncertainties such as fog to an outdoor MVS benchmark dataset to demonstrate the functionality of the proposed framework in any type of scenario and develop a new research direction for the MVS literature.
- The framework presented here has a high level of adaptability and a special focus on the capacity and traffic of a WSN. It has many flavours based on trade-offs among transmission time, the quality of keyframes, and the accuracy of the activity recognition model with computationally variant classifiers.

Subsequent sections are organised as follows. **Section 2** provides a literature review and **Section 3** explains the proposed framework in detail. **Section 4** presents experimental results for MVS and activity recognition and **Section 5** concludes the overall paper with a discussion of the limitations of this work and future directions for research.

II. Related Work

This section is divided into two sub-sections, which describe representative studies of MVS and activity recognition that are related to the proposed framework. The first MVS study was published a decade ago in 2010 by Fu et al., who introduced the first MVS indoor dataset [8]. The employed MVS methods are computationally complex and hence are not suitable for resource-constrained devices. Unlike MVS, activity recognition is a dense research field in which a variety of techniques have been developed, focusing on a wide range of applications.

A. Multi-view Video Summarization

The majority of MVS methods are based on handcrafted features that are integrated with traditional machine learning approaches to achieve the final summary generation task. The initial MVS schemes [8, 9] utilised features such as SIFT descriptors, Gaussian and Laplacian differences, and motion features, followed by statistical learning approaches such as K-means or other clustering techniques for summary generation. The next group of MVS schemes [10, 11] used the same features but added pre- or post-processing such as background subtraction, together with supervised (e.g. SVM) or unsupervised learning. The final summary in this approach has either a uniform duration or a fixed-length set by the user. The next development in the area of MVS [12, 13] utilised mid-level features, motion-based shot boundary detection, spatiotemporal graphs, and low-level features such as colour and edge histograms. The final summary is generated using the same statistical learning-based techniques.

A breakthrough in the field of MVS is noticed after the use of learned features in the prerequisite steps for generating summaries. In 2016 [14], BVLC CaffeNet and spatiotemporal C3D features were first used for sparse coding and video representation, respectively. In addition to clustering and template matching, sparse representative selection over learned embedding have been used to generate final summaries. Hussain et al. incorporated long short-term memory (LSTM) with CNN features to generate video skims from multi-view videos, using cloud servers for the generation

of the final output [15]. Another recent study utilised resource-constrained devices for MVS, without a post-processing mechanism for recognition of the underlying activity [16]. For a detailed investigation of these methods, readers are referred to our recent survey that covers the MVS literature in a detailed manner [2].

B. Activity Recognition

Recent deep learning-based methods for activity recognition have achieved high levels of accuracy. These methods extract features from the final layers of various deep learning models and apply sequential learning for activity recognition [17-20]. For example, Ullah et al. used features extracted from a pre-trained AlexNet, followed by bi-directional LSTM and achieved improved results in terms of human action recognition [20]. A similar strategy explored the convolutional features of optical flow with multi-layer LSTM to achieve activity recognition [21]. This approach outperformed [20] and other state-of-the-art methods; however, its long-running times restricted its applicability in real-world surveillance networks. In a similar way to previous methods, researchers have used different types of CNN architectures and configurations to represent human actions that considered both stationary and non-stationary environments [22]. To date, activity recognition methods have not performed computation over resource-constrained devices with significant accuracy. Thus, to effectively address this problem, we optimised the VGG-19 model [23] to allow it to function over resource-constrained devices [24], and the details of our scheme are presented below.

III. Proposed Framework

The overall framework is divided into two major modules based on the differences in functionalities and processing complexity. The first module is related to the acquisition of multi-view video data, shot segmentation, and summary generation. The second module performs computationally expensive processes involving deep features extraction, its comparison and encoding, and final activity prediction. The entire framework uses a number of resource-constrained devices (master and slave) connected to a WSN in an IoT environment. This section provides details of the proposed system, and the major steps are given in **Algorithm 1** and visualised in **Figure 2**. **Algorithm 1** explains the overall processing in our proposed framework: the "slave device" procedure involves the keyframes generation and their propagation in an encoded form to the master device, whereas the "master device" procedure is responsible for keyframes decoding, features extraction, inter-view correlations computation, features encoding, and finally, activity prediction. **Algorithm 2** takes a directory of images as input and initialises a fog parameter, loops throughout the directory to read the images and applies image processing to obtain foggy output images based on the value of the fog parameter. The image processing step involves finding the maximum values for the input image before and after adding the fog parameter, followed by image normalisation for final output generation. These processes are explained in subsequent sections.

A. Objects of Interest Detection for Shot Segmentation

Surveillance analysts are typically interested in human and vehicle activities and we, therefore, considered these to be objects of interest. For this purpose, we employed a miniature version of the YOLO CNN model [25], which is much faster and has higher accuracy as compared to other object detectors. This object detector works well in normal scenarios but performs poorly when videos contain fog. To overcome this limitation, the pre-trained weights of the YOLO model are used and retrained using foggy data. Fog is applied globally to the entire image, without targeting only the objects of interest. The global addition of fog allows better learning since in real-life surveillance scenarios, fog is always observed throughout the image. Foggy data are generated from the COCO dataset [26] by adding synthetic fog to the images using a MATLAB script with different levels of fog ranging from 0 to 255. We conducted experiments with different values of the fog parameter (fp), as described in the experimental section, and value of fp = 220 is selected as optimal. The process of adding fog to images is illustrated in **Algorithm 2**.

Most of the available multi-view datasets are related to indoor environments, such as Office and Bl-7f. We, therefore, conducted experiments only with the available multi-view outdoor datasets, such as Road [8]. The trained model is used in each slave device for object detection. The camera attached to the device transmits frames to our customised object detection model, and frames containing people or vehicles with a confidence score greater than 0.9 are stored and used in further steps, while the remaining frames are discarded.

B. Keyframe Selection Mechanism

A summary is generated for the segmented frames containing objects since events occur due to the interactions between these objects. We, therefore, considered shots from the MVS datasets that contained possible events such as sitting on a chair or entering a room. These events are possible only due to the presence of a human, and we therefore segmented shots in which humans were detected. The shots containing the desired objects are numerous and contained redundant frames, and therefore could not be considered in the final summary. To avoid redundancy and provide users with a very compact representation, mutual information of features of consecutive frames are computed. This process is executed over slave devices with limited resources for execution, therefore, we used low-level features in an intelligent way. Furthermore, to avoid redundancy and obtain better results, we compared mutual information between two consecutive oriented FAST and rotated BRIEF (ORB) feature descriptors rather than using only the Euclidean or another distance measuring method. The feature descriptor is acquired from each frame by computing the ORB points [27]. ORB points perform well on real-world images involving viewpoint changes, and this approach is an efficient alternative to the use of SIFT and SURF features. The ORB descriptor is rotationally invariant, resistant to noise, and performs better even in low-light scenarios, as shown by the results for object tracking [28] and image retrieval [29] problems. The ORB features descriptor is, therefore, the best fit for our redundancy removal task, functional over

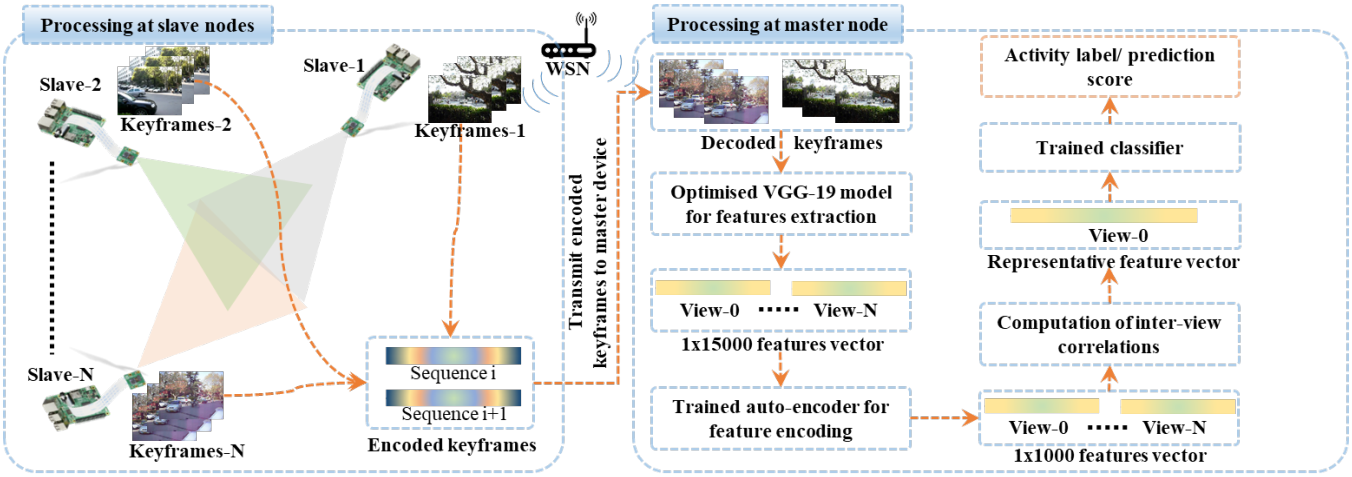

Figure 2: Conceptual diagram of our proposed framework, in which slave devices with a vision sensor capture multi-view video data, apply shot segmentation, extract keyframes, and encode and transmit them to the master device for computation of inter-view correlations and activity recognition

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^N = encoded keyframes for view N, KF_s^N = selected keyframes of view N, KF_d^N = decoded keyframe sequence for view N, F_{15000} = 15,000 dimensional features, and F_{1000} = 1,000 dimensional features.

1: **Initialisation**: 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^N from S_c
11: KF_e^N ← encode the KF_s^N
12: Transmit KF_e^N to master device
13: **Repeat** Step 3 for next shot until video ends
14: **Procedure** Master device
15: KF_d^N ← decode KF_e^N
16: F_{15000} = $M_{vgg\text{-}19}$ (KF_d^N)
17: Compute F_{15000} for acquired keyframes for all the views of the current interval
18: F_{1000} ← AE (F_{15000})
19: **IF** views > 1
a. Loop through all features (f_m, f_n) of different views
b. **IF** Euclidean distance (f_m, f_n) < 0.3
i. Select f_m and compare it with f_{m+1}
ii. Continue comparison until view N
c. End **IF**
20: End **IF**
21: Final prediction ← SVM (F_{1000})
22: **Output:** Final prediction

resource-constrained devices. In our case, frames containing people with the same information (i.e., similar ORB descriptors) are discarded and only a single frame (the first one) is selected. The selected frame is considered to be a keyframe on the resource-constrained device. The formula used to calculate the mutual information is given in (1).

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

$$I_o = \frac{(I_r \times M_1)}{M_2} \quad (2)$$

Here, f_p is the previous frame, f_n is the current frame, and N is the size of the feature vector. Mutual information is computed using the scikit-learn [30] library in Python.

Algorithm 2. Foggy data generation algorithm

input: Directory of input images, fog parameter = fp
N ← length (directory)
for K = 1 **to** N **do** > loop through folder
I_c ← read_image(K) > read single image
M_1 ← max(max(I_c)) > find maximum value1
I_r ← I_c + fp > add fog parameter
M_2 ← max(max(I_r)) > find maximum value2
I_o ← **Eq. 2** (I_r) > normalise foggy image
save I_o > save foggy image
end for

C. *Inter-view Correlation Computation and Activity Recognition*

Human activity recognition is an important task in surveillance video analysis. An activity is a sequence of movements in continuous video frames that requires spatiotemporal features extraction. In our system, the activity needs to be recognised on the master device after acquiring the summary from the slave device. For activity recognition, we use a pre-trained VGG-19 CNN model for spatial feature representation, followed by an auto-encoder that captures the temporal features of the ongoing sequence. Unlike VGG-19, the well-known AlexNet [31] and GoogleNet [32] show poor performance in terms of capturing tiny patterns in visual data, due to the use of 11×11 and 7×7 filter sizes with strides of four and two pixels, respectively. We also performed experiments on MobileNetV2 [33], in which features are extracted from the final convolution layer; however, the results were not convincing compared to state-of-the-art

methods (see **Table 2**). The VGG-16 model has recently been studied in the activity recognition problem by Ullah et al., and we outperformed their method on various datasets using VGG-19 [34]. VGG-19 can extract discriminative visual features since it uses a 3×3 filter in all convolutional layers with a fixed one-pixel stride. This helps in processing the smallest receptive field to grab the notions of tiny patterns. VGG-19 has 16 convolutional and three fully connected layers, and deep features are extracted from the final fully connected layer (FC8) with dimensions 1×1000. The FC7 layer of VGG-19 outputs representations with 4096 dimensions, resulting in a large feature vector for a single sequence and yields huge processing for the encoding of temporal representations. Thus, FC7 layer features are not practical to be considered for activity recognition over resource-constrained devices in real-time scenarios. We extract spatial frame-level features with dimensions 1×15000 from a sequence of 15 summarized keyframes (1×1000 from each frame).

Feature extraction is performed on a Movidius VPU stick and the original VGG-19 model is compressed using the Neural Computing Stick software development kit (NCSDK). This optimised the size of the original model from 574.7 MB to 287.3 MB using Movidius' compatible graph format. The temporal changes and sequential patterns in the 15,000 features are learned using our proposed auto-encoder. Motivated by the widespread use of an auto-encoder network in many applications [35], we utilised this in our framework. An auto-encoder analyses patterns in high-dimensional data to reduce the dimensionality and we believe that it could learn spatiotemporal patterns in the deep features of 15 consecutive frames. To effectively learn the temporal changes, we performed several experiments that squeezed the 15,000 features to a lower number of dimensions. We first encoded 15,000 features to 8,000, followed by encoding to 1,000 features (called as *first setting*). This type of setting is very effective, because its mean square error (MSE) for encoded features is very low, which provides high accuracy. However, the time complexity of the first setting did not meet the needs of the embedded devices, since the squeezing involves an additional step of encoding features to 8,000 dimensions. In our second attempt, we encoded the 15,000 high-dimensional features directly to 1,000 (called as *second setting*). In this setting, the accuracy was slightly compromised when using linear SVM, although the computational requirements of the embedded devices were met. Our auto-encoders are trained for 40 epochs and we utilised sparsity regularisation with a proportion value of 0.1 and L2 weight regularisation to reduce the chances of over-fitting and being stuck in local minima. To this end, MSE with support from sparsity regularisation and L2 regularisation is trained as a cost function for our auto-encoders. In the first setting, the MSE was reduced to 0.012 after 40 epochs and in the second setting, the value was reduced to 0.044.

The trained auto-encoder is utilised to squeeze 1×15000-dimension VGG-19 features to 1×1000, which are further employed to compute the inter-view correlations and finally to carry out activity recognition. The features of different views after encoding are compared with each other using the Euclidean distance and if the output difference is below a certain threshold, a single feature vector is considered for further processing. For instance, if the features of views 1 and 2 have high-level similarity, the features of view 1 are used and are then compared with those of views 3, 4, and so on, until the total number of views is reached. In case of lower similarity (i.e., a higher value of the Euclidean distance between two views), both of the feature vectors are used in the advanced activity recognition step. The encoded features of different views are spatiotemporal representations of an activity, which are passed to the SVM classifier for activity recognition. We trained a one-versus-all multi-class SVM because it trains $N-1$ classifiers (where N is the number of total classes), which is an efficient approach for resource-restricted devices. All types of SVMs were examined for the activity recognition problem, including linear, quadratic, and cubic SVMs, as reported in **Section 4.3**.

D. Malleability of the Proposed Framework

We designed this framework by considering the major limitations of WSNs and IoT devices, such as the computational complexity and storage capacities. We also wanted to provide an independent activity analysis system that is malleable in both certain and foggy environments. For this purpose, we first investigated different MVS and activity recognition methods and carried out in-depth research on the capabilities of WSNs. Our literature study showed that an independent system is needed that could capture multi-view video data, conduct analysis, and provide a compact summary of recognised activities from lengthy surveillance videos. To this end, we performed extensive experiments with different configurations and parameters, which could be interchanged according to the users' requirements and the available resources. For example, a slave node with very limited resources in a WSN could encode its summary using PNG compression before transmission to the master node for activity recognition. This would ensure real-time summary generation and transmission to the master node, despite the relatively slow connectivity and limited communication bandwidth. Similarly, at the master node, different options can be applied to find a balance between the execution time and accuracy of the activity recognition method. Various classifiers were used in our experiments, with different computational complexities. In addition, different experiments are performed to directly extract features from a sample summary as well as encoded features with different configurations, in order to achieve optimal recognition of ongoing activities from multi-view camera settings. The details of the evaluation of these configurations are discussed in **Section 4.4.**

IV. Experimental Results

The experimental evaluation of our proposed framework is divided into three subsections: MVS, activity recognition, and data transmission. First, we explain the performance of summarization using resource-constrained devices when compared to state-of-the-art techniques. Next, we evaluate the activity recognition module on the master device with the Movidius setup. Although we used Raspberry Pi (RPi) for our

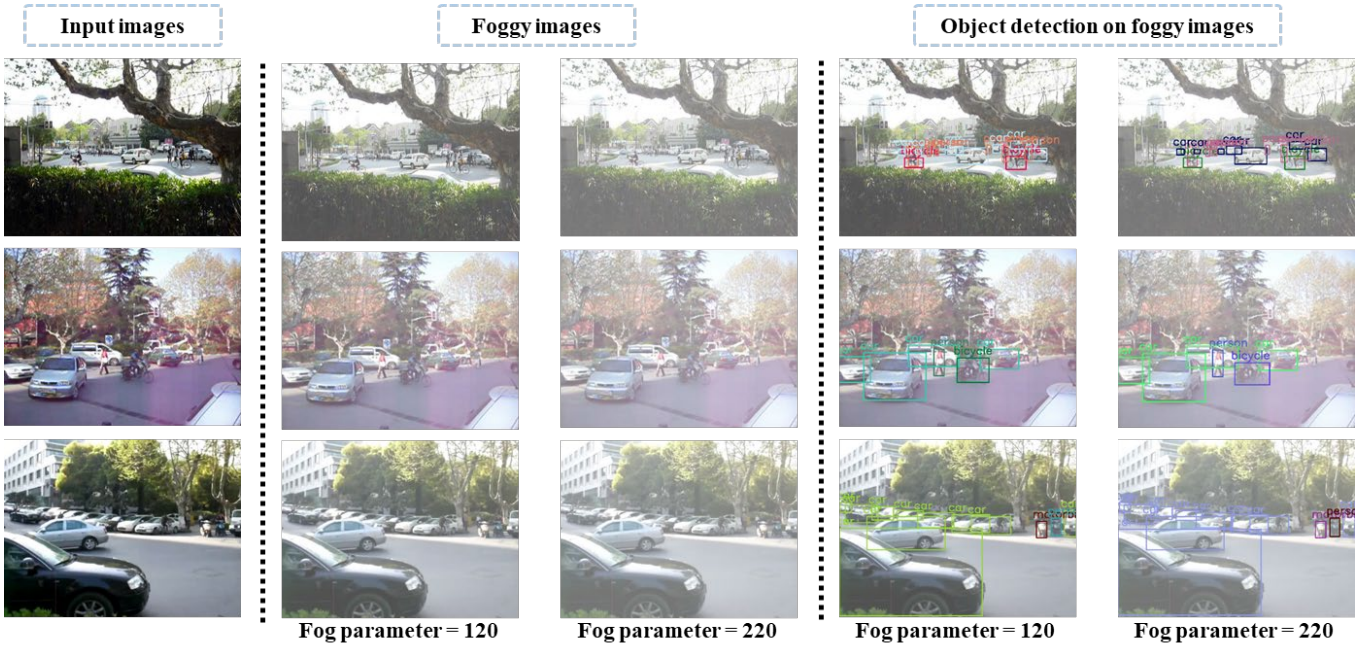

Figure 3: Sample results from road dataset: (a) normal input images; (b) fog applied with different parameters; (c) object detection in foggy images. The detection results are encouraging for object detection in a foggy environment.

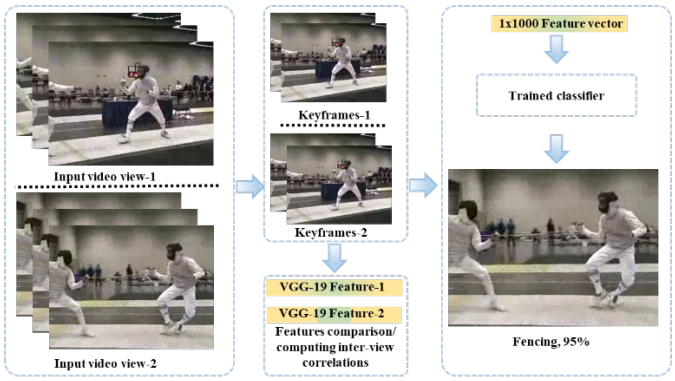

Figure 4: Sample results generated from UCF-50 video captured from two different views. Keyframes from both input videos are generated and VGG-19 features are extracted. These features are then encoded and compared to select a representative feature vector. The final output for the activity is generated by passing this feature vector to a trained model that yielded a probability score along with the predicted class.

experiments with the MVS module, our work is not limited to RPi and can be applied to similar devices. At the activity recognition stage, we used MATLAB as a simulation tool and then transformed the concepts and programming implementation to the Keras deep learning framework in Python. The model trained using Keras with Tensorflow backend could be converted into a Movidius-compatible graph format using NCSDK. We tested our model on RPi, that can be adjusted for use with other Movidius-supported devices. Sample results from the integrated framework for an input video from the UCF-50 dataset with different views are shown in **Figure 3**, and the implementation is available from Github[1].

[1]https://github.com/tanveer-hussain/MVSandARinIoT

A. Datasets

We used five different datasets for our experiments, three of which were utilised for object detection in foggy scenarios and MVS and the remainder to evaluate our activity recognition method. Office [8] is widely used to evaluate the performance of MVS methods due to its public availability and the presence of ground truths. It was created by utilising four stable cameras in an office environment. There is no synchronisation among the captured videos in terms of light variation. Bl-7f [13] is the most challenging dataset in the MVS literature and was created using 19 different cameras installed on the seventh floor of Taiwan University. The recorded videos contain different events that are closely related to each other, with maximum overlap. The installed cameras are synchronised, still, and fixed, but show different light conditions in different views. The ground truth and the dataset are publicly available for research purposes. The Road [8] multi-view dataset was captured using handheld cameras and contains an extreme level of shuddering effects and variable light conditions. In contrast to the two datasets described above, the Road dataset contains people and vehicles that were recorded in outdoor daylight. This dataset was used to test our object detection model in foggy outdoor environments. The results of object detection tested on the Road dataset are shown in **Figure 4**, for different levels of fog. The activity recognition part of our framework was evaluated using the UCF50 [36] and YouTube 11 [37] action datasets. UCF50 contains assorted human action videos with high-level shuddering due to camera motion, differences in viewpoints, and the scalability of objects. The total number of classes was 50, where some actions were closely related to each other,

which affected the overall accuracy of the classifiers. The YouTube dataset was less challenging than UCF50 due to absence of closely overlapping classes, but the videos still include camera motion, cluttered backgrounds, light variations, and changes in viewpoint. A total of 11 different action videos are available in this dataset.

TABLE I
Performance comparison of the proposed framework with state-of-the-art methods. The Office and Bl-7f datasets are used for comparison. The highest F1 score of all the methods is shown in bold. P refers to precision and R represents the recall score. The research by Kuanar et al. [12] achieved overlapping scores for the Bl-7f dataset, but their method cannot be used for IoT devices, thus indicating the superiority of our proposed framework.

Methods	Office			Bl-7f			Execution time per frame (seconds)
	P	R	F1 score	P	R	F1 score	
[8]	1.00	0.61	0.75	-	-	-	-
[38]	0.41	0.63	0.50	-	-	-	-
[12]	1.00	0.69	0.81	0.75	0.98	**0.85**	-
[13]	0.25	0.75	0.40	0.58	0.61	0.6	-
[39]	1.00	0.73	0.81	-	-	-	-
[40]	1.00	0.70	0.81	-	-	-	-
[41]	1.00	0.81	0.89	-	-	-	-
[15]	0.93	0.86	0.90	-	-	-	3.98
[42]	0.94	0.89	0.91	-	-	-	1.23
Proposed	0.94	0.92	**0.93**	0.85	0.88	**0.85**	0.45

B. MVS Evaluation

Table I shows the MVS results for our framework in comparison with other state-of-the-art techniques. We provide an objective evaluation of two benchmark datasets from the MVS literature (Office and Bl-7f). The time complexity of our framework is given in the last column of **Table I**, which indicates the execution time for an RPi device. The better performance of our framework compared to existing methods can be seen from **Table I**. For the Office dataset, our method achieved the highest F1 score of all the MVS methods considered here. The most recent methods for MVS with the Office dataset scored 0.90 and 0.91 and involved higher computational complexity. The method presented in [15] used a heavy-weight CNN model for spatiotemporal point processing to select video skims. The drop in computational complexity from a recent MVS method operating on resource-constrained devices is shown in **Table I**; our method generated 0.78 secs faster MVS in comparison scheme in [42]. For the Bl-7f dataset, our framework outperformed [13] and achieved an F1-score similar to that reported in [12], but our framework can also operate on resource-constrained devices.

C. Evaluation of Activity Recognition

The activity recognition module of our system was evaluated using two benchmark action datasets, UCF-50 and YouTube 11. The comparison with state-of-the-art techniques based on overall accuracy is shown in **Table II**. We conducted several activity recognition experiments using the original VGG-19 features, encoded features with S_1 and encoded features with S_2 and applied several variants of SVM including linear, quadratic, and cubic SVM. In **Table II**, S_1 represents the first setting, in which the 15,000 features of a single sequence were encoded into 8,000 and then 1,000 features. S_2 represents the second setting, in which 15,000 features were directly encoded into a 1,000-feature vector. These different settings can be used as alternatives for different scenarios, in terms of both accuracy and complexity. For the UCF50 dataset, approaches involving trajectory analysis [18], hierarchical clustering [19], ML-LSTM [43], and a deep auto-encoder with CNN [34] achieved values of accuracy of 92.3%, 93.2%, 94.9%, and 96.4%, respectively. Our S_1 and S_2 with cubic SVM achieved overall accuracies of 96.2% and 93.8%, respectively. For the YouTube 11 action dataset, the single-stream CNN [17], hierarchical clustering [19], DB-LSTM [20], ML-LSTM [43], and auto-encoder with CNN [34] achieved accuracies of 93.1%, 89.7%, 92.84%, 95.8%, and 96.21%, respectively.

The proposed method with full VGG-19 features and cubic SVM achieved a value of 98.7% for accuracy; S_1 achieved 89.1% for quadratic SVM and 91.3% for cubic SVM, whereas S_2 achieved 94.7% and 95.6% for quadratic and cubic SVM, respectively. Based on the above comparisons, the accuracies of our different settings are higher than or similar to those of state-of-the-art methods; however, in terms of the computational complexity, existing methods are designed for high processing GPUs, while our system operates on resource-constrained devices. Although the performance of our settings with linear SVM is less than 80%, our framework achieves 90% accuracy with quadratic SVM and this can be considered a state-of-the-art result for processing using embedded devices with a low-cost VPU stick. Finally, the experiments performed with MobileNetV2 gave lower accuracy for both datasets. Hence, lightweight MobileNetV2 is not a suitable option for real-time surveillance for activity recognition.

TABLE II
Detailed comparative analysis of various settings of the proposed activity recognition method against the recent alternative techniques

Method	Features/learning	Overall accuracy (%)	
		UCF-50	YouTube actions
Single-stream CNN [17]	A CNN network for motion and static information in a stream	-	93.1
Trajectory analysis [18]	Bag of visual words and their fusion	92.3	-
Hierarchical clustering [19]	Hierarchical clustering multi-task learning	93.2	89.7
DB- LSTM [20]	AlexNet, bi-directional LSTM	-	92.84
ML-LSTM [43]	Optical flow convolutional features, LSTM	94.9	95.8
Deep auto-encoder and CNN [34]	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
	S_1 (1x1000)	86.4	84.6
	S_2 (1x1000)	80.8	90.6
Proposed (quadratic SVM)	VGG-19 (1x15000)	94.9	98.5
	S_1 (1x1000)	92.2	89.1
	S_2 (1x1000)	94.4	94.7
Proposed	VGG-19 (1x15000)	**96.6**	**98.7**

(cubic SVM)	S_1 (1x1000)	96.2	91.3
	S_2 (1x1000)	96.1	95.6

A.Statistical Evaluation of WSNs

Wireless networks have limited bandwidth and communication cost resources, and it is, therefore, unfeasible to transmit huge-sized surveillance videos to the cloud or local analysis centres due to the constraints on transmission and data storage. In addition, manual searching for an event in long videos is a tiresome task. This section provides statistical details of the savings in storage capacity, bandwidth, and transmission time of our method, as illustrated in **Figure 5**.

(i) Storage Capacity

To the best of our knowledge, no specific video analytics dataset is available that could be used to confirm the efficiency and effectiveness of our framework in terms of bandwidth, transmission time, and storage capacity. We, therefore, used an example video to generate keyframes and to encode and transmit them to the master device for activity recognition. We compared all the parameters mentioned above and reported in **Figure 5**. The frames in the example video are non-compressed, with ideal conditions for transmission. We first calculated the total number of pixels by multiplying the height (h) of the frame by the width (w) in pixels = h × w. In the Office video, h = 480 and w = 640, giving a total of 307,200 pixels; this is then multiplied by the depth of the frame (eight), which output the size of the whole video in bits. Finally, we divided the obtained number by eight to acquire the size of the frame in bytes, subsequently converted to MB. The size of a single frame from this video is 0.29 MB. Similarly, to calculate the size of the Office view-1 video (video-0), the size of each frame was multiplied by the number of frames inside it. This video contained 26,940 frames and multiplying this by the size of each frame yielded 7892.57 MB. The keyframes, if stored instead of all the frames occupy 8.49 MB space and similarly, 2.16 MB space is reserved if the keyframes are encoded using the proposed compression technique. A large difference in storage was observed, thus indicating the effectiveness of our system in terms of saving storage capacity; this refers to the preservation of communication bandwidth.

(ii) Transmission Time

The transmission time in our framework is reduced in two ways: (i) in low-bandwidth networks, we encode the keyframes with PNG compression; and (ii) we transmit only important frames over the IoT network, which saves a significant amount of time. We assume an ideal situation in the network for transmission of a frame in the WSN from a slave to the master device and assume that the distance is 0.3 km and the speed is 200,000 km/s, with a data rate of 32 Mbps. In receiving the data from the slave (source), there are three stages that determine the time required: (i) getting the frame to the WSN; (ii) transmission of the data through the network; and (iii) loading the data from the router/hub to the master (destination) device [44]. The time t_1 in (2) represents the transmission time from the source to network router, whereas t_2 in (3) is the time taken by the network to transmit data from the source router to the destination. The transmission time from the destination router to the master device is the same as t_1. The overall transmission time is the sum of t_1, t_2, and t_3, where t_1 and t_3 are the same. The transmission times calculated for the overall frames, keyframes, and encoded keyframes were 4137.98, 4.45, 1.13 seconds, respectively, as shown in **Figure 5**.

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

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

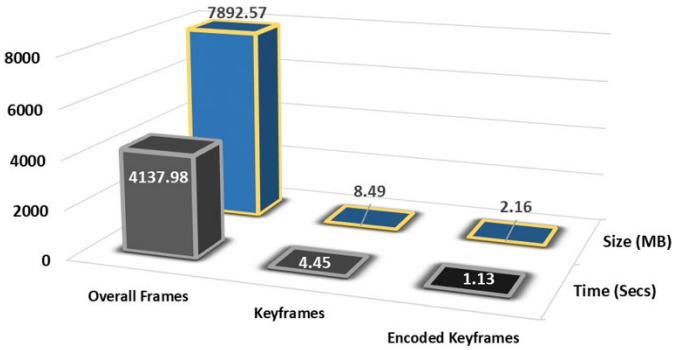

Figure 5: Transmission times and storage sizes for transmitting and storing the overall video (Office-0 video) versus keyframes and encoded keyframes. Very considerable savings in transmission time and storage capacity can be observed from these possible settings.

V. Conclusions and Future Research Directions

In this paper, we integrated MVS with activity recognition under the umbrella of a unified framework. A complete setup including slaves and a master resource-constrained device working independently in an IoT environment is presented in this research. The hardware requirements included slave devices equipped with a vision sensor and wireless sensors, and a master device with Intel Movidius NCS to run optimised deep learning models at the edge. The slave devices captured multi-view video data, detected objects, extracted features, computed mutual information, and generated a summary. This summary was received at the master device, which used an optimised trained model to compute the inter-view correlation and optimal activity recognition. The proposed MVS algorithm and activity recognition models outperformed state-of-the-art methods.

The framework implemented here suggests some avenues for future research. The current framework is only able to detect objects in foggy environments and does not apply a defogging strategy. The activity recognition algorithm in our framework only considers single-view video sequences as input, and hence has limited applicability to multi-view cameras. In the future, we plan to extend this work by deeply investigating multi-view activity recognition algorithms with different parameters and configurations in resource-constrained environments. Furthermore, we intend to explore fuzzy set theories [45], control charts [46], and reinforcement learning [47] for use in our framework, as these are used for various challenging tasks [48]. Finally, we aim to apply defogging techniques prior to MVS, to enhance the activity recognition module and to allow the overall framework to be adopted in any type of scenario.

References

[1] K. Muhammad, T. Hussain, and S. W. J. P. R. L. Baik, "Efficient CNN based summarization of surveillance videos for resource-constrained devices," vol. 130, pp. 370-375, 2020.

[2] T. Hussain, K. Muhammad, W. Ding, J. Lloret, S. W. Baik, and V. H. C. de Albuquerque, "A comprehensive survey of multi-view video summarization," *Pattern Recognition,* vol. 109, p. 107567, 2021/01/01/ 2021.

[3] K. Muhammad, T. Hussain, M. Tanveer, G. Sannino, and V. H. C. J. I. I. o. T. J. de Albuquerque, "Cost-effective video summarization using deep CNN with hierarchical weighted fusion for IoT surveillance networks," vol. 7, no. 5, pp. 4455-4463, 2019.

[4] A. Tejero-de-Pablos, Y. Nakashima, T. Sato, N. Yokoya, M. Linna, and E. Rahtu, "Summarization of user-generated sports video by using deep action recognition features," *IEEE Transactions on Multimedia,* vol. 20, no. 8, pp. 2000-2011, 2018.

[5] H. Wang, H. Yu, P. Chen, R. Hua, C. Yan, and L. Zou, "Unsupervised Video Highlight Extraction via Query-related Deep Transfer," in *2018 24th International Conference on Pattern Recognition (ICPR)*, 2018, pp. 2971-2976: IEEE.

[6] M. Tiwary, D. Puthal, K. S. Sahoo, B. Sahoo, L. T. J. J. o. P. Yang, and D. Computing, "Response time optimization for cloudlets in mobile edge computing," vol. 119, pp. 81-91, 2018.

[7] Intel. *Intel® Neural Compute Stick 2*. Available: https://software.intel.com/en-us/neural-compute-stick (Accessed on 5 May 2020)

[8] Y. Fu, Y. Guo, Y. Zhu, F. Liu, C. Song, and Z.-H. Zhou, "Multi-view video summarization," *IEEE Transactions on Multimedia,* vol. 12, no. 7, pp. 717-729, 2010.

[9] Y. Li and B. Merialdo, "Multi-video summarization based on video-mmr," in *11th International Workshop on Image Analysis for Multimedia Interactive Services WIAMIS 10*, 2010, pp. 1-4: IEEE.

[10] Y. Muramatsu, T. Hirayama, and K. 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*, 2014, p. 1: ACM.

[11] S. H. Ou *et al.*, "Communication-efficient multi-view keyframe extraction in distributed video sensors," in *2014 IEEE Visual Communications and Image Processing Conference*, 2014, pp. 13-16.

[12] S. K. Kuanar, K. B. Ranga, and A. S. Chowdhury, "Multi-view video summarization using bipartite matching constrained optimum-path forest clustering," *IEEE Transactions on Multimedia,* vol. 17, no. 8, pp. 1166-1173, 2015.

[13] S.-H. Ou, C.-H. Lee, V. S. Somayazulu, Y.-K. Chen, and S.-Y. Chien, "On-line multi-view video summarization for wireless video sensor network," *IEEE Journal of Selected Topics in Signal Processing,* vol. 9, no. 1, pp. 165-179, 2015.

[14] R. Panda, A. Das, and A. K. Roy-Chowdhury, "Embedded sparse coding for summarizing multi-view videos," in *2016 IEEE international conference on image processing (ICIP)*, 2016, pp. 191-195: IEEE.

[15] T. Hussain, K. Muhammad, A. Ullah, Z. Cao, S. W. Baik, and V. H. C. de Albuquerque, "Cloud-Assisted Multiview Video Summarization Using CNN and Bidirectional LSTM," *IEEE Transactions on Industrial Informatics,* vol. 16, no. 1, pp. 77-86, 2019.

[16] T. Hussain, K. Muhammad, J. Del Ser, S. W. Baik, and V. H. C. de Albuquerque, "Intelligent Embedded Vision for Summarization of Multiview Videos in IIoT," *IEEE Transactions on Industrial Informatics,* vol. 16, no. 4, pp. 2592-2602, 2019.

[17] S. Ramasinghe and R. Rodrigo, "Action recognition by single stream convolutional neural networks: An approach using combined motion and static information," in *Pattern Recognition (ACPR), 2015 3rd IAPR Asian Conference on*, 2015, pp. 101-105: IEEE.

[18] X. Peng, L. Wang, X. Wang, and Y. Qiao, "Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice," *Computer Vision and Image Understanding,* vol. 150, pp. 109-125, 2016.

[19] A.-A. Liu, Y.-T. Su, W.-Z. Nie, and M. Kankanhalli, "Hierarchical clustering multi-task learning for joint human action grouping and recognition," *IEEE transactions on pattern analysis and machine intelligence,* vol. 39, no. 1, pp. 102-114, 2017.

[20] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, and S. W. Baik, "Action recognition in video sequences using deep bi-directional LSTM with CNN features," *IEEE Access,* vol. 6, pp. 1155-1166, 2018.

[21] A. Ullah, K. Muhammad, J. Del Ser, S. W. Baik, and V. H. C. de Albuquerque, "Activity recognition using temporal optical flow convolutional features and multilayer LSTM," *IEEE Transactions on Industrial Electronics,* vol. 66, no. 12, pp. 9692-9702, 2018.

[22] A. Ullah, K. Muhammad, I. U. Haq, and S. W. Baik, "Action recognition using optimized deep autoencoder and CNN for surveillance data streams of non-stationary environments," *Future Generation Computer Systems,* vol. 96, pp. 386-397, 2019.

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556,* 2014.

[24] K. S. Sahoo *et al.*, "ESMLB: Efficient Switch Migration-based Load Balancing for Multi-Controller SDN in IoT," 2019.

[25] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767,* 2018.

[26] T.-Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context," in *Computer Vision – ECCV 2014*,

Cham, 2014, pp. 740-755: Springer International Publishing.

[27] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *2011 International Conference on Computer Vision*, 2011, pp. 2564-2571.

[28] H. Zhou, Y. Yuan, and C. Shi, "Object tracking using SIFT features and mean shift," *Computer vision and image understanding,* vol. 113, no. 3, pp. 345-352, 2009.

[29] C. Zhang, X. Wang, J. Feng, Y. Cheng, and C. Guo, "A car-face region-based image retrieval method with attention of SIFT features," *Multimedia Tools and Applications,* vol. 76, no. 8, pp. 10939-10958, 2017.

[30] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of machine learning research,* vol. 12, no. Oct, pp. 2825-2830, 2011.

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097-1105.

[32] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1-9.

[33] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510-4520.

[34] A. Ullah, K. Muhammad, I. U. Haq, and S. W. Baik, "Action recognition using optimized deep autoencoder and CNN for surveillance data streams of non-stationary environments," *Future Generation Computer Systems,* 2019/01/28/ 2019.

[35] X. Yang, C. Deng, F. Zheng, J. Yan, and W. Liu, "Deep Spectral Clustering using Dual Autoencoder Network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4066-4075.

[36] K. K. Reddy and M. Shah, "Recognizing 50 human action categories of web videos," *Machine Vision and Applications,* vol. 24, no. 5, pp. 971-981, 2013.

[37] J. Liu, J. Luo, and M. Shah, "Recognizing realistic actions from videos “in the wild”," in *Computer vision and pattern recognition, 2009. CVPR 2009. IEEE conference on*, 2009, pp. 1996-2003: IEEE.

[38] S.-H. Ou *et al.*, "Communication-efficient multi-view keyframe extraction in distributed video sensors," in *Visual Communications and Image Processing Conference, 2014 IEEE*, 2014, pp. 13-16: IEEE.

[39] R. Panda, A. Dasy, and A. K. Roy-Chowdhury, "Video summarization in a multi-view camera network," in *Pattern Recognition (ICPR), 2016 23rd International Conference on*, 2016, pp. 2971-2976: IEEE.

[40] R. Panda, A. Das, and A. K. Roy-Chowdhury, "Embedded sparse coding for summarizing multi-view videos," in *Image Processing (ICIP), 2016 IEEE International Conference on*, 2016, pp. 191-195: IEEE.

[41] R. Panda and A. K. Roy-Chowdhury, "Multi-view surveillance video summarization via joint embedding and sparse optimization," *arXiv preprint arXiv:1706.03121,* 2017.

[42] T. Hussain, K. Muhammad, J. Del Ser, S. W. Baik, and V. H. C. de Albuquerque, "Intelligent Embedded Vision for Summarization of Multi-View Videos in IIoT," *IEEE Transactions on Industrial Informatics,* 2019.

[43] A. Ullah, K. Muhammad, J. Del Ser, S. W. Baik, and V. Albuquerque, "Activity Recognition using Temporal Optical Flow Convolutional Features and Multi-Layer LSTM," *IEEE Transactions on Industrial Electronics,* 2018.

[44] L. Bruce. (2019). *Computer Networks: A Systems Approach*. Available: https://book.systemsapproach.org/foundation/performance.html#delay-bandwidth-product (Accessed on 11 May 2020)

[45] L. Kong *et al.*, "Distributed Feature Selection for Big Data using Fuzzy Rough Sets," *IEEE Transactions on Fuzzy Systems,* 2019.

[46] T. Hussain *et al.*, "Intelligent baby behavior monitoring using embedded vision in IoT for smart healthcare centers," vol. 1, no. 15, p. 2019, 2019.

[47] K. Gai and M. Qiu, "Reinforcement learning-based content-centric services in mobile sensing," *IEEE Network,* vol. 32, no. 4, pp. 34-39, 2018.

[48] K. Gai and M. Qiu, "Optimal resource allocation using reinforcement learning for IoT content-centric services," *Applied Soft Computing,* vol. 70, pp. 12-21, 2018.