Received: July 4, 2022. Revised: August 25, 2022.

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Query-Based Video Summarization System Based on Light Weight Deep Learning Model

Saif K. Jarallah[1] **Sawsen Abdulhadi Mahmood**[1]*

Computer Science Department, College of Education, Mustansiriyah University, Iraq
* Corresponding author's Email: sawsenhadi@uomustansiriyah.edu.iq

Abstract: Video summarization technologies strive to provide a succinct and thorough description by selecting the most informative frames from the original video. The essential concept of query-based video summarization methods is represented by constructing a video summary related to user query in term of user interest. This paper aims to address the problem of query-based video summarization construction task by adopting a lightweight deep learning model for object detection task in order to evident how concerning a frame or shot is to a given query. Both YOLOv3 and Tiny YOLOv3 deep learning models are used separately in the proposed system to train the networks using images and videos dataset including diverse types of objects such as; (motorcycles, bicycles, cars, buses, and trucks). Subsequence, the most relative frames to a given query are selected and assembled as keyframes using a modified K-mean clustering scheme to provide different interesting summaries from the original video. Based on the experimental results of object detection phase, we have obtained an average object detection accuracy around 93%, 83% based on YOLOv3 and Tiny YOLOv3 deep learning models respectively. Comprehensive experiments were performed to evaluate the proposed video summarization system, which exhibited an efficient summarization rate close to 33% of the original video. Further experiments were conducted using standard UTE dataset to exhibit the competitive performance of the proposed method compared to state of art query-based video summarization methods.

Keywords: Video summarization, Deep learning, Real time object detection, K-mean clustering, Pipeline parallelism.

1. Introduction

The amount of video surveillance cameras deployed on private premises and public places are in a continual progression. The estimated number of cameras is more than 350 million surveillance cameras worldwide in 2016 [1], which produced millions of gigabytes of videos data per week. Although considerable progress has been achieved in automated content-based video interpretation, there is still a need for humans (security guards, police officers, etc...) to look at the content of recorded videos. Visual analysis of video data can be time-consuming and prone to errors after hours of observations. Therefore, effective solutions that can facilitate this laborious task are of great interest. A video summary is an abbreviated video that preserves the important shots of the original video while removing the spatiotemporal segments which are not of interest. Object detection is a technique used to detect objects in videos and images from surveillance cameras[2]. Video summarization methods are tend to detect the important visual data from surveillance stream and can help in efficient indexing and retrieving of required data from huge surveillance datasets [3]. Deep learning methods are widely used for object detection task such as YOLO, RNN, F-CNN, etc. [4]. YOLOv3 network is a fast and accurate deep learning model used for object detection task and consists of convolution neural network (CNN) supported by classification techniques to analyse the outputs of the stacked layers [5]. Commonly, YOLOv3 model used a customize dataset for detecting and recognizing variant types of objects in the environment.

The salient challenge of query-based video summarization methods is to determine the relevance between a given video and user defined query in order to construct a video summary includes the most interested frames of user subjectivity. In this paper, a query-based video summarization method is presented and implemented using YOLOv3 and Tiny YOLOv3 deep learning models for object (query) detection task. A modified K-mean clustering algorithm is presented in our framework for key frames selection task. The main contributions of the proposed system can be summarized as follows; collecting and annotating video frames taken from public dataset, adopting light weight deep learning model for query object detection task, performing key frames selection approach using modified K-mean clustering algorithm, adopting pipeline parallelism technique to implement the proposed system functions and finally constructing the video summery related to a given query.

The rest of this paper is organized as follows; section 2 illustrates the most interested and related works; section 3 introduces the background theory. Section 4 presents the proposed methodology. The dataset description used in our experiments is illustrated in section 5. An illustration of pipeline parallelism technique is clarified in section 6. The proposed key frames selection scheme is presented in section 7. The video construction phase is presented in section 8. Section 9 involves the whole experimental results and evaluation of the proposed system. This paper is concluded in section 10.

2. Related works

The objective of query-focused video summarization methods is to get a diversified and concise selection of video frames or segments using supervised learning approaches including various types of query. Recently, deep learning models-based object detection task are widely used. Query conditioned video summarizing task considers user-provided queries in the form of texts to learn and provide user-oriented summaries [6–9].

A. Sharghi, J. S. Laurel, and B. Gong [9] proposed a memory network [10] parameterized sequential determinantal point process [11] for tackling the query-focused video summarization. They used memory network to implicitly attend the user query about the video onto different frames within each shot. They compared their method with the sequential determinantal point process (SeqDPP)[11] and Sequential and Hierarchical DPP (SH-DPP)[7]. To address this challenge, they trained a sequential and hierarchical DPP (SH-DPP) against their approach, and achieved 44.19% average F1-Score using UT Egocentric (UTE) dataset [12]. Y. Zhang et al. were suggested a query-conditioned three-player generative adversarial network for query-conditioned video summarization [13]. Video representations conditioned by user queries are generated in the generator through concurrently encoding visual information and user queries, where generator is used to learn the joint representation of query and video. The discriminator takes three different summaries as input and discriminate the real summary from the other two summaries which are randomly generated. Their approach achieved 46.05% average F1-Score using UTE dataset.

However, user preferences are not considered in the approaches indicated above, therefore summaries may not be resilient or generalizable for various users. As a result, video summaries based on user queries came into focus and started to offer more individualized summaries to users. N. Baghel et al. were proposed an image conditioned keyframe-based video summarization using object detection[14]. Their approach has taken both global and local features to generate video summary. Object detection based on (YOLOv3) method has been utilized for local features extraction task. For global features, they used the salient region to find important regions in the video frames using colour combination and depth of that region. They employed HSV colour model to find salient region in images by using subtraction of left and right images with saturation and value components of HSV colour space. Their approach achieved 57.06% average F1-score using UTE dataset with processing time close to 7.81 times faster than actual time of original video. Practically, YOLOv3 network makes predictions at three scales through down-sampling the size of the input image into blocks of size 32×32, 16×16 and 8×8 blocks[14]. However, the proposed work in [14] approach used two scales only; 16×16 and 8×8 blocks to decrease the processing time. As a result, YOLOv3 workflow will miss the advantage of detecting small objects and counting objects in each frame, which limits the user's flexibility in querying and these small objects may contain critical information and events and should be included in the video summary.

2.1 Real-time object detection

In recent years, with the rapid development of deep learning and the excellent results of the Alex-Net model in the ILAVRC challenge, an increasing number of deep convolutional neural networks are being used in the field of computer vision-based object detection task. The current algorithms are mainly based on candidate-window- and regression-for performing object detection algorithms. With the development of convolutional neural networks, two-stage algorithms for object detection based on candidate windows have been adopted. The regions with convolutional features method (RCNN) proposed by Girshick [15] applied the convolutional neural network for object detection task at the first time, using a selective search to extract candidate frames. However, this method is time-consuming and computationally complicated. In response to these problems, K. He, X. Zhang, S. Ren, and J. Sun proposed the pyramid pooling networks algorithm (SPPNet)[16], using a convolutional neural network to extract features of the candidate frame, but the training time was hang increased[17]. The fast region-based convolutional network Fast-RCNN[18] was an improvement on these methods that used the visual geometry group network VGG-16[19] as the backbone network to integrate feature extraction, object classification, and position regression into a model, thereby improving the detection speed and accuracy [20]. However, the selective search method adopted in Fast-RCNN cannot guarantee real time detection. In 2015, S. Ren, K. He, R. Girshick, and J. Sun proposed Faster-RCNN towards real-time object detection with region proposal networks [21] , which used a region proposal network (RPN) to generate candidate regions to truly realize end-to-end training of the object detection network. However, their research has the problem of inaccurate positioning. To resolve this problem, J. Dai, Y. Li, K. He, and J. Sun proposed a region-based fully convolutional network (R-FCN)[22], which used a residual network ResNet [23] as a feature extraction network to improve the effectiveness of feature extraction and classification processes. Aiming to solve the problem of inaccurate positioning of the prediction box in Faster-RCNN, Cai and Vasconcelos proposed a cascade structure detector named a cascade region-based convolutional network Cascade RCNN [24] , which set different intersection over union (IOU) thresholds for training to improve the accuracy of the network prediction box, but the detection speed cannot be guaranteed. Due to low speed of object detection methods based on candidate windows, regression-based object detection algorithms have been suggested to use. YOLOv1 network “you only look once” [25] treats object detection task as regression problem. To improve the detection accuracy, Redmon and Farhadi proposed YOLOv2 [26] based on YOLOv1 and Darknet-19[27] as the backbone network of YOLOv2, with anchor boxes used to predict bounding boxes to improve detection accuracy. In 2016, W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. C. Berg proposed a multi scale feature fusion method called a single shot multi box detector (SSD) [28] which added an anchor mechanism to the RPN network, and proposed a similar a priori box method to generate a bounding box for objects. It uses feature maps of different layers for detection, which is more accurate than YOLO but has lower detection accuracy for small objects. Attempts have been made to solve the problems of SSD [29] by improving its feature extraction and detection accuracy. In 2018, Redmon and Farhadi proposed YOLOv3 [30] with Darknet-53[27] as the backbone network, using the idea of feature pyramid networks [31] and feature maps of different scales for detection, thus improving the detection of small objects.

3. Theoretical background

The pre-trained YOLOv3 deep learning model is suggested to use in our framework to perform the object detection task due to its efficiency, speed and accuracy. YOLOv3 network is three times faster than SSD [30]. Tiny YOLOv3 deep learning model adopts lightweight training proposed by V. Mazzia et al. and Zhang Z. Yi et al. [32] [33]. Tiny YOLOv3 network has a smaller module size than YOLOv3 network and more suitable for real-time object detection. However, it loses some detection accuracy. In this research, the object detection task based on YOLOv3 and Tiny YOLOv3 networks were employed separately, and their results were compared.

3.1 YOLOv3 network architecture

YOLO is a Convolutional Neural Network (CNN) used for real-time object detection. CNNs network is classifier-based frameworks that interacts with input images as structured arrays of data and aims to recognize patterns. YOLO network has the benefit of being a lot faster than other object detection models with maintaining higher detection accuracy. It permits the model to view the entire image at testing mode, so its predictions are informed by the whole global context of the image. YOLO network gives the scores of regions based on similarities present in the input images for predefined classes [25]. Darknet-53 adopts the idea of ResNet[23] network and adds residual modules to the network, where 1, 2, 8, 8, and 4 are the numbers of repeated residual modules, and each residual module consists of two convolution layers and a residual layer. The entire network structure has no pooling layer, and the downsampling operation of the network is completed by setting the convolution step size to 2. After the convolution layer, the size of the image is reduced by half [2]. The specific network structure is shown in Table 1. YOLOv3 network makes detection in three different scales to accommodate variant size of objects through

Received: July 4, 2022. Revised: August 25, 2022.

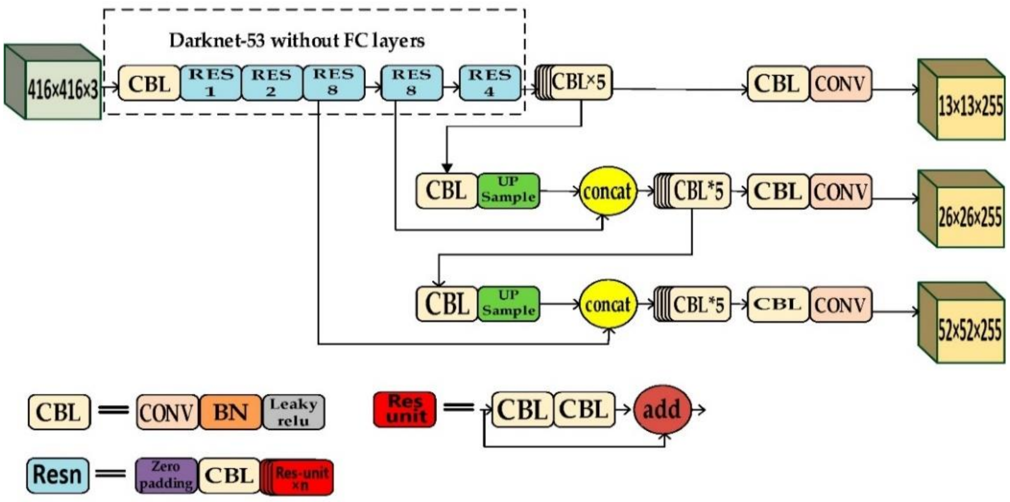

Figure. 1 YOLOv3 network architecture [34]

Table 1. Darknet-53 network structure [2].

	Type	Filter	Size/Stride	Output
	Conv. Conv.	32 64	3 × 3 3 × 3/2	416 × 416 208 × 208
1×	Conv. Conv. Residual	32 64	1 × 1 3 × 3	208 × 208
	Conv.	128	3 × 3/2	104 × 104
2×	Conv. Conv. Residual	64 128	1 × 1 3 × 3	104 × 104
	Conv.	256	3 × 3/2	52 × 52
8×	Conv. Conv. Residual	128 256	1 × 1 3 × 3	52 × 52
	Conv.	512	3 × 3/2	26 × 26
8×	Conv. Conv. Residual	256 512	1 × 1 3 × 3	26× 26
	Conv.	1024	3 × 3/2	13× 13
4×	Conv. Conv. Residual	512 1024	1 × 1 3 × 3	13× 13

using strides 32, 16, and 8. This means, if we feed an input image of size 416×416, YOLOv3 will make detection on the scale of 13×13, 26×26, and 52×52. YOLOv3 downsamples the input image into 13×13 and predicts the 82nd layer for the first scale. The 1st detection scale yields a 3-D tensor of size 13×13×255. After that, YOLOv3 takes the feature map from layer 79 and applies one convolutional layer before upsampling it by a factor of 2 to have a size of 26×26. Then, the upsampled feature map is concatenated with the feature map from layer 61. The concatenated feature map is subjected to a few more convolutional layers until the 2nd detection scale is performed at layer 94. The second prediction scale produces a 3-D tensor of size 26×26×255. The same design is performed one more time to predict the 3rd scale. The feature map from layer 91 is added to one convolutional layer and concatenated with a feature map from layer 36. The final prediction layer is done at layer 106, which yield a 3-D tensor of size 52×52×255. In summary, YOLO network predicts over three different scales detection, so if we feed an image of size 416×416, it produces three different output shape tensors, 13×13×255, 26×26×255, and 52×52×255. Fig. 1. illustrates the pipeline workflow of YOLOv3 deep learning network.

3.2 Tiny YOLOv3

Tiny YOLOv3 is a simplified version of YOLOv3. The main difference is that the backbone network uses 7 convolution layers and max networks to extract features (similar to darknet19), while the grafted network uses 13×13 and 26×26 resolutions for detection task. The main advantages of Tiny YOLOv3 are that the network is simple, the calculation is small, and it can runs on the mobile terminal or device side. The disadvantage is that the accuracy is relatively low (both candidate frames and classification accuracy are relatively low)[35].

4. Proposed methodology

Specifically, a light weight deep learning model is adopted in this paper to predict and determine the interested frames which included the query object for constructing the related video summarization .The concept of relatedness that apparent how concerning a video frame is with a given query is achieved by using YOLOv3 network that converts the input video frames into query frames space.

Received: July 4, 2022. Revised: August 25, 2022.

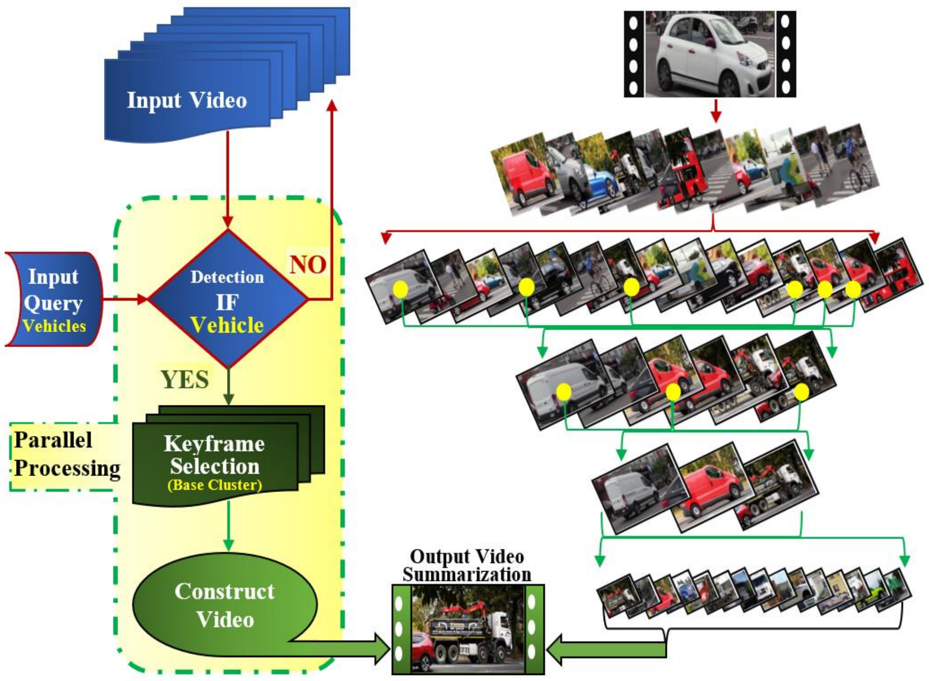

Figure. 2 The structure of the proposed query-based video summarization system based on light weight deep learning model.

The framework of the proposed query-based video summarization system involves four main phases; video preprocessing using annotation process, object detection phase to determine the interested frames (including the query object) along input video sequence using YOLOv3 network, keyframes extraction phase based on modified K-mean clustering method and video summary construction phase. First, the video frames are collected and preprocessed through the annotation process to provide the ground truth of the target object's location at each frame. The annotation process was performed based on the bounding box strategy. Afterward, video frames are divided into non-overlapped segments with uniform length in term of shots. Then, each shot is fed to YOLOv3 network to detect the interested frames relative to query object. As a result, the output will be set of interested frames (key frames) need to ensample them in term of video summary that involves the relative frames to the query. Fig. 2 illustrates the main structure of the proposed query-based video summarization system based on light weight deep learning model.

5. Datasets description

In this paper, five types of objects "classes" are specified for the purpose of training YOLOv3 network and Tiny YOLOv3 deep learning models to achieve object detection task. The classes are; motorcycle, bicycle, car, bus and truck. The process of collecting the samples is mainly involves three main steps; images collection, images resizing and data splitting step. We have used the large-scale object detection dataset (benchmarks) named Microsoft's COCO dataset to perform the performance evaluation of the proposed query-based video summarization system. COCO dataset is widely used by machine learning and computer vision methods[36]. The five types of objects have been taken from (COCO dataset 2017 for training and validation) and (open image-v6[37] for testing). The collected dataset composes of 38297 images which are used for training both models. Furthermore, we have used 13 videos taken from Pexels (online videos dataset)[38], for testing models, which included short videos of natural vehicle movement on streets. Furthermore, we have used UT Egocentric (UTE) standard dataset [12] for comparison study purpose. UTE dataset involves four videos named P01, P02, P03 and P04 captured under various scenarios. Each video long is 3–5 hours and contains a diverse set of events.

5.1 Dataset splitting

Throughout the experiments, we have divided the dataset into three main sets; training, validation and testing sets for the purpose of conducting evaluation process. The dataset was divided by a ratio (training ≅ 80%, testing ≅ 20%) along with appropriate validation ratio as clarified in Fig. 3.

Received: July 4, 2022. Revised: August 25, 2022.

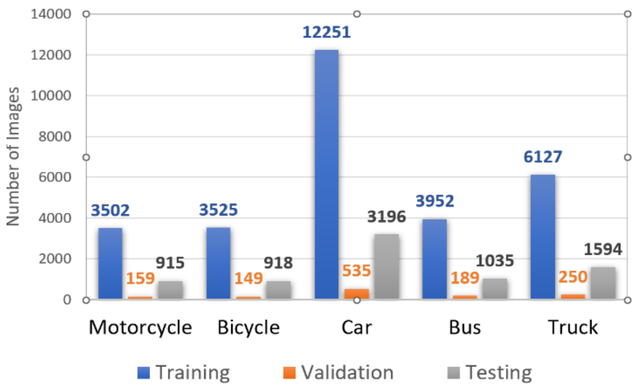

Figure. 3 An illustration of dataset splitting strategy according to COCO-dataset 2017 [36]

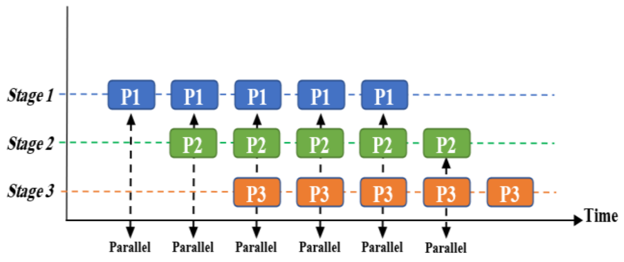

Figure. 4 Illustrated pipeline parallelism, where P1, P2 and P3 are different process [43]

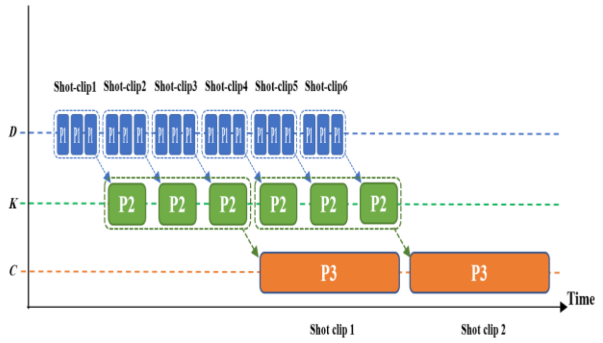

Figure. 5 Pipeline parallelism workflow; where *D* denotes detection phase, *K* represents keyframes selection phase and *C* refers to video summary construction

It's worth noting that the collected image samples may contained more than desired object or query. In this context, we have relied on the number of images for each class to determine the ratio of training, validation, and testing images. In addition, all objects belonging to the previously defined query or class present in the single image were exploited for training, so the number of images will be different from the number of bounding box labels for each category. In this way, the dataset splitting process will provide the flexibility to exploit memory during network training with less number of samples. Subsequence, we have got 94095 bounding boxes of all the trained images.

5.2 Pre-processing

Pre-processing techniques are applied to assist machine learning methods for obtaining higher detection rate of the input samples. The following pre-processing techniques have been used in our framework and applied upon the input images: (i) scaling the input image into fixed size: 416×416 px. (ii) Annotate the query object presents in the acquired images or video frames by a boundary box to generate the ground-truth. The frame's labelling process was conducted according to the method used in [39] to create the bounding box with (YOLO format box [40]) for all video frames that containing the interested object (the query). (iii) Data augmentation technique was performed to enrich dataset samples such as; flipping on horizontal axis (left - right) flipping [41].

6. Pipeline parallelism

Pipeline parallelism technique organizes a parallel program as a linear sequence of *s* stages. Each stage processes elements of a data stream, passing each processed data element to the next stage, and then taking on a new element before the subsequent stages have necessarily completed their processing [42]. Pipeline parallelism is used especially in streaming applications that perform video, audio, and digital signal processing [43] as shown in Fig. 4.

We have used the pipeline parallelization technique in our framework to ensure that the processing time of the proposed video summarization system is suitable for real-time application. The proposed system has three main phases; query object detection, keyframes selection and video summary construction. Practically, the input for each phase is different, since the input of object detection phase is video sequence, the input of keyframes selection phase is set of frames (containing the query object) and the input of video summary construction is keyframes. Therefor the shot size between phases is different, as shown in Fig. 5.

The size of video shots in the proposed video summarization system is determined by the number of frames such as; the shot size between object detection phase and keyframes selection phase is 900 frames, while the shot size between keyframes selection phase and video summary construction phase is 5400 frames. The possibility of increasing the number of frames according to user defined has been taken in our consideration.

Table 2. Quantitative results of *Speedup* factor experiment using sequential and parallelism workflows

Model Used	T_s	T_p	*S(N)*
YOLOv3	510.51	423.39	1.20
Tiny YOLOv3	322.36	277.08	1.16

```
Algorithm 1: The modified K-mean clustering
Input: x, N // x is the data points and N is the size of x
Output: k // output cluster
Initialization: i = 0
Begin
Step1. Calculate the minimum mean between $x_i$ and $x_j$, by using Eq. (5).
Step2. Assume the centroid is the index ($x_i$) of the minimum mean.
Step3. Calculate the threshold (Th), by using Eq. (4).
Step4. Calculate the initial (maxd and mind) by Eq. (6)
Step5. While i < (N-1) do
        Calculate the distance (dist) between centroid and $x_i$, by using Eq. (6)
        If (dist > maxd) then maxd = dist
        else (dist < mind) then mind = dist
        endif
        i= i+1
        end while
Step6. Calculate the bias, by using Eq. (7)
        i=0, j=0
Step7. While i < (N-1) do
        Calculate the distance (dist) between centroid and $x_i$, by using Eq. (6).
        Add bias to distance: dist = dist + bias.
        If (dist < Th) then assign data point $x_i$ to the cluster:
            $k_j = x_i$
            j = j+1
        endif
      i =i+1
    end while
End
```

6.1 Speedup factor

The potential benefit of parallel computing is typically measured by the consuming time of implementing specific task on a single processor versus the consuming time of implementing the same task on *N* parallel processors. The *Speedup* factor S(N) using *N* parallel processors is defined according to Eq. (1) [44]:

$$Speedup = S(N) = \frac{T_p(1)}{T_p(N)} = \frac{T_s}{T_p} \quad (1)$$

Where *Tp(1)* is the processing time with single processor and *Tp(N)* is the processing time with parallel processors. Note that speedup factor should be greater than 1. An experiment was conducted to evaluate the performance of the proposed system in the two cases; sequential and parallel mods, as shown in Table 2. The experiment was conducted on video sequence with duration time 851 seconds and frames number are 17432 frames.

As shown in Table 2, the *S(N)* factor is greater than 1, indicating the superiority of consuming time when using pipeline paralleling technology compared to sequential technology in the proposed framework.

7. Keyframes selection scheme

This section clarified the keyframes selection scheme of the output frames generated from the object detection phase. The most informative and similar frames are grouped and resampled as shots based on modified K-mean clustering algorithm. A modified version of K-mean clustering algorithm is adopted in our framework which inspired from the research presented by M. Z. Hossain et al. [45], in which dynamically clusters all data from big data set without defining the cluster size *K*. Specifically, the selection of *K* value is considered as critical issue to obtain an efficient and meaningful cluster using K-mean algorithm. The authors in [45], have suggested a strategy for grouping huge data sets based on threshold value *Th* and the index of data point *Idx* to improve the clustering quality as illustrated in Eq. (2) and Eq. (3):

$$Th = \sum_{i=0}^{N-1} \frac{\sum_{j=0}^{N-1} \frac{dist(xi,xj)}{N}}{N} \quad (2)$$

$$Idx = \min\left(\sum_{i=0}^{N-1}{}_{j=0}\, dist(x_i, x_j)\right) \quad (3)$$

7.1 The proposed modification

A modified version of K-mean clustering method has been suggested in our framework to obtain an effective grouping of data points. Based on the following equations, the threshold value is dynamically determined to enhance the threshold movement range, the distance between cluster center and threshold was decreased by improving Eq. (2) and Eq. (3) according to Eq. (4) and Eq. (5):

$$Th = \sum_{i=0}^{N-1} \frac{\sum_{j=0}^{N-1} \frac{\sqrt{(x_i - x_j)}}{N}}{N} \quad (4)$$

$$Idx = min\left(\sum_{i=0}^{N-1}{}_{j=0} \sqrt{(x_i - x_j)}\right) \quad (5)$$

Where *N* is cluster size and *x* denotes the data points set. Further, a *bias* value has been added to equalize the distance between data points, which leads to increasing the number of points located out of *Th* range as clarified in Eq. (6) and Eq. (7).

$$dist = \sqrt{centroid - x_i} \qquad i=1,2,\ldots N \qquad (6)$$

$$bias = \sqrt{maxd - mind} \qquad (7)$$

Where *dist* is the distance between centroid and x_i, *(maxd)* is the max distance between centroid and x_i, and *(mind)* is the min distance between centroid and x_i. The main steps of the modified K-mean clustering algorithm are stated in algorithm 1.

Increasing the distance between the points and adding the bias leads to increasing in the flexibility in displacing the point out the threshold range in an equal way and suitable for the farthest point and the nearest point in the same cluster. It's worth noting, we have implemented the process of key frames selection for each 900 frames (30 seconds along). In this way, the cluster size is set to be 900 data points or frames, with the possibility of increasing the number of frames (cluster size) according to user defined.

8. Video summary construction

The last phase in the framework of the proposed system represents by constructing the video summary. After keyframes selection process, set of the interested frames that included the query object and nearest to cluster centre, the video summary is generated from these key frames. As mentioned earlier, video creation mechanics are among the stages of pipeline paralleling, so the time of videos is predetermined, and the process of calculating the time of the video by the number of frames (with the possibility of increasing the number of frames according to user defined), where one second consists of 30 frames. The main steps of video summary construction are stated in algorithm 2.

9. Experimental results

In this section, the experimental results and performance evaluation of the proposed video summarization system are denoted and recorded. We have used three videos to evaluate the object detection performance of YOLOv3 and Tiny Yolv3 models in the testing mode. Video-1 sequence consists of close and clear objects with free overlapping in their movement. Video-2 sequence has a middle overlapping of the movement objects. Video-3 sequence involves overlapping and non-overlapping of movement objects. The tested videos have (640×480) resolution and mp4. format with threshold value intersection over union IoU = 0.5.

```
Algorithm 2: The proposed video summary construction
Input: k // set of Keyframes
       N //  the number of keyframes
       L // number of video summary frames (user defined)
Output: v // video summary
Initialization: i = 0, m = 0, j = L
Begin
  While i < (N-1) do
        if (i ≤ j) then construct video:
            add frame (k_i) to video (v_m)
        else
            m = m +1
            j = j + L
        endif
        i = i + 1
  end while
End
```

9.1 Object detection evaluation

In order to evaluate the performance of query object detection task based on YOLOv3 and Tiny YOLOv3 models, we have used four metrics for characterizing the performance such as; *IoU*, precision, recall, average precision *AP@α* and mean average precision *mAP* [46] as described in the following subsections:

9.1.1 Intersection over union (IoU)

Intersection over union (*IoU*) is calculated as the overlapping area (intersection) divided by the union area of the ground truth bounding box (*gt*) and the predicted bounding box (*pd*) respectively with certain threshold as shown in Eq. (8):

$$IoU = \frac{(gt \cap pd)}{(gt \cup pd)} = \frac{Area\ of\ overlap}{Area\ of\ Union} \qquad (8)$$

9.1.2. Precision and recall

Precision measures how accurate your predictions are. It is calculated as the number of true positive (*TP*) divided by the sum of true positive (*TP*) and false positive (*FP*), as shown in Eq. (9).

$$Precision = \frac{Tp}{TP+FP} \qquad (9)$$

The recall metric is used to determine the true predictions from all correctly predicted data. It is calculated as the number of true positive (*TP*) divided by the sum of the true positives (*TP*) and false negative (*FN*), as illustrated in Eq. (10).

Table 3. Hyperparameters setting of YOLOv3 and Tiny YOLOv3 networks.

hyperparameter	setting
Images resolution	416×416
Batch size	8
Optimizer	SGD
Initial learning rate	0.01
No. of epochs	600 epochs
No. of Classes	5
GPU memory	RTX 6 GByte
Train images (COCO 2017)	29357
Valid. images (COCO2017)	1282
Test images (Openimage.v6)	7658

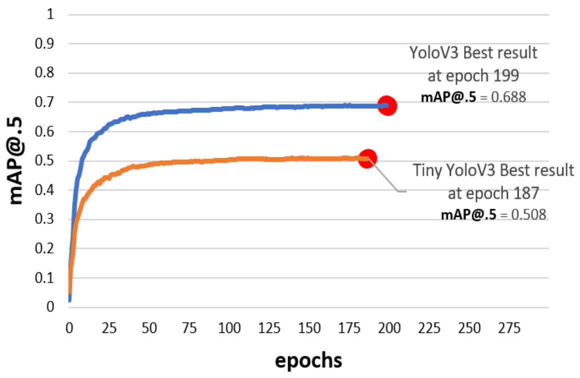

Figure. 6 Simulation results of training mode for YOLOv3 and Tiny YOLOv3 networks

$$Recall = \frac{Tp}{TP+FN} \tag{10}$$

9.1.3. Average precision

The average precision metric *AP@α* represents the area under the Precision-Recall Curve (*AUC-PR*) and evaluated based on the threshold *α* of *IoU* metric. Formally, it is defined in Eq. (11).

$$Ap@\alpha = \int_0^1 p(R)dR \tag{11}$$

Where $p(r)$ is precision measure at recall *R*.

9.1.4. Mean average precision (mAP)

The mean average precision *mAP* refers to the average of (*AP*) metric and calculated for all classes with threshold α, as indicated in Eq. (12).

$$mAP@\alpha = \frac{1}{n}\sum_i^n Api\ , \quad for\ n\ classes \tag{12}$$

9.2 Video summarization evaluation

In order to evaluate the performance of video summarization, we have utilized the summarization rate *R(S)* of the output summarized video and *F1-Score* to compare the proposed method against state of art query-conditioned video summarization methods.

9.2.1. Summarization rate

The summarization rate *R(S)* of the output summarized video denotes the number of frames of output video summary *m* divided by total number of frames of the original video *n*, as clarified in Eq. (13) [47].

$$R(S) = \frac{m}{n} \tag{13}$$

Since *R(S)* rate values are located within the interval (0, 1], the ideal video summary rate is preferred to be between 0.1 and 0.5. The proposed video summarization system is based on selecting the interested frames which included the query object. Based on our experiments, approaching *R(S)* rate to zero value does not mean distorting the video sequence. For example, if the input video composed of 1800 frames (one minuet- duration) and the query object appears along one second (30 frames), then *R(S)* is 0.16.

9.2.2. F1-Score

As mentioned in section 2., the most related works have been utilized the common evaluation metric *F1-Score* to evaluate the output video summary based on the formula illustrated in Eq. (14)[48].

$$F1 - Score = \frac{2\times(precision\times Recall)}{(precision+Recall)} \tag{14}$$

9.3 Networks setting

In the proposed framework, we have used YOLOv3 and Tiny YOLOv3 deep learning models separately to train the collected dataset for object detection task and choose the best model that achieve higher detection rate. In the training mode, extensive experiments have been implemented on the dataset samples with hyperparameters configuration process as illustrated in Table 3. Throughout training experiments, the batch size is set to 8, since it’s based on image resolution and GPU memory.

9.4 Training experiments (images-dataset)

In these experiments, we have implemented YOLOv3 and Tiny YOLOv3 networks using COCO-2017 images dataset to detect five types of objects

Received: July 4, 2022. Revised: August 25, 2022.

Table 4. Hyperparameters setting –based training, validation and testing mods

Hyperparameter	YOLOv3	Tiny YOLOv3
Layers	261	48
Parameters	61,524,355	8,675,932
GPU Memory Usage	3.17 (GByte)	1.75 (GByte)
Size File of Weights	128m	17m
Training Time	36h: 45m	8h: 49m
Epochs No.	299	288
Best Result Epoch	199	187
Precision -valid	0.78	0.72
Recall -valid	0.613	0.443
mAP@.5 -valid	0.688	0.508
Precision-test	0.855	0.81
Recall-test	0.829	0.744
mAP@.5-test	0.873	0.794

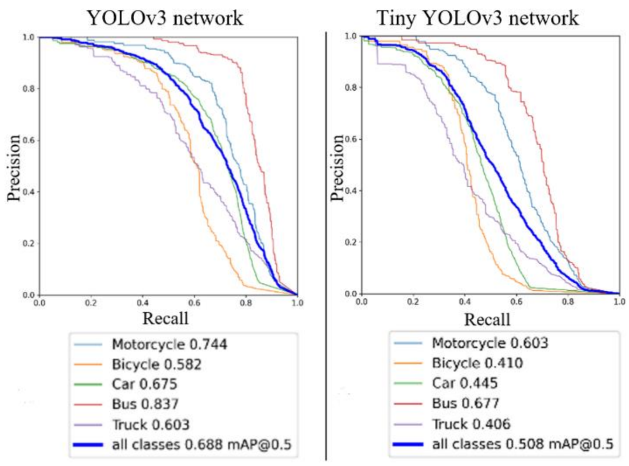

Figure. 7 Simulation results of validation mode for YOLOv3 network and Tiny YOLOv3 network

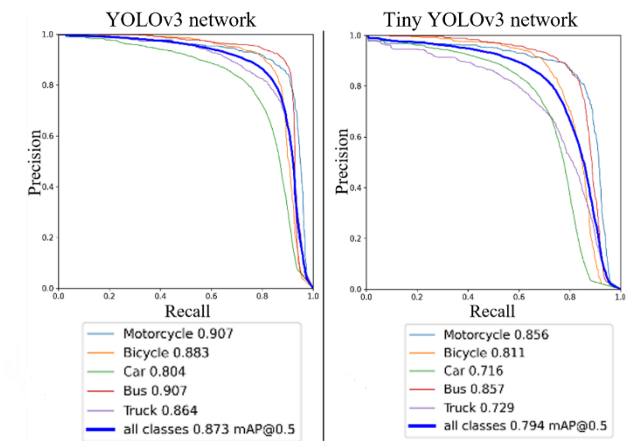

Figure. 8 Simulation results of testing mode for YOLOv3 network and Tiny YOLOv3 network

(classes) and recorded the obtained results of *mAp*@0.5 metric as shown in Fig. 6. The quantitative results of training, validation and testing mods of YOLOv3 and Tiny YOLOv3 networks are illustrated in Table 4. The visualization results of

Table 5. The quantitative results of object detection accuracy performed on first video

	Category	Precision	Recall	Ap@.5
YOLOv3	Motorcycle	0.981	0.94	0.967
	Car	0.985	0.971	0.988
	All	0.983	0.956	**mAP@.5 = 0.978**
Tiny YOLOv3	Motorcycle	1	0.9	0.961
	Car	0.975	0.937	0.968
	All	0.987	0.918	**mAP@.5 = 0.964**

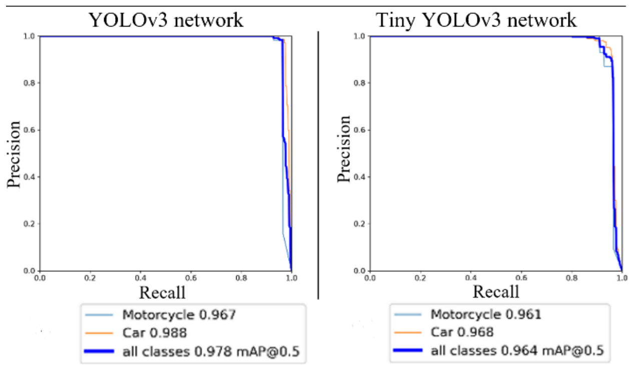

Figure. 9 Visualization results of object detection accuracy performed on first video by (YOLOv3 network and Tiny YOLOv3 network)

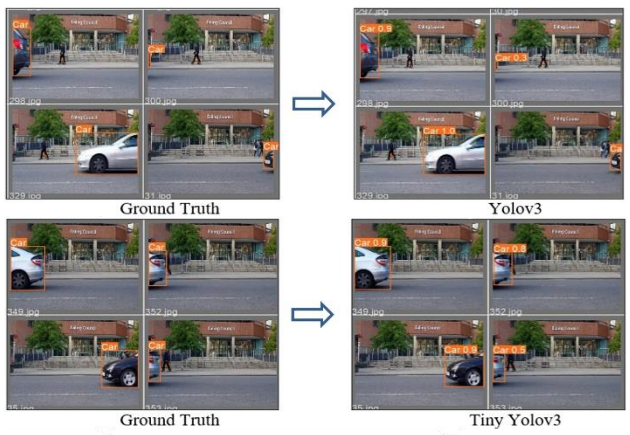

Figure. 10 Query object detection results of first video by YOLOv3-based detection and Tiny YOLOv3-based detection

validation mode are shown in Fig. 7. The visualization results of testing mode are shown in Fig. 8 which exhibited *mAP@*.5 =0.873 for YOLOv3 model, while Tiny YOLOv3 model specified *mAP@*.5=0.794.

9.5 Training experiments (videos dataset)

The performance evaluation of the proposed video summarization system using YOLOv3 and Tiny YOLOv3 networks was conducted based on

Received: July 4, 2022. Revised: August 25, 2022.

Table 6. The quantitative results of object detection accuracy performed on second video

	class	Precision	Recall	Ap@.5
YOLOv3	Bicycle	0.954	0.876	0.939
	Car	0.858	0.89	0.931
	Truck	0.961	0.892	0.953
	All	0.925	0.886	**mAP@.5 = 0.941**
Tiny YOLOv3	Bicycle	0.824	0.62	0.63
	Car	0.85	0.865	0.89
	Truck	0.952	0.616	0.856
	All	0.875	0.7	**mAP@.5 = 0.792**

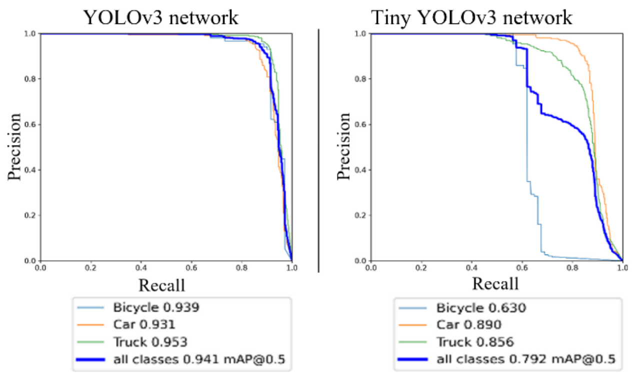

Figure. 11 Visualization results of object detection accuracy performed on second video by (YOLOv3 network and Tiny YOLOv3 network)

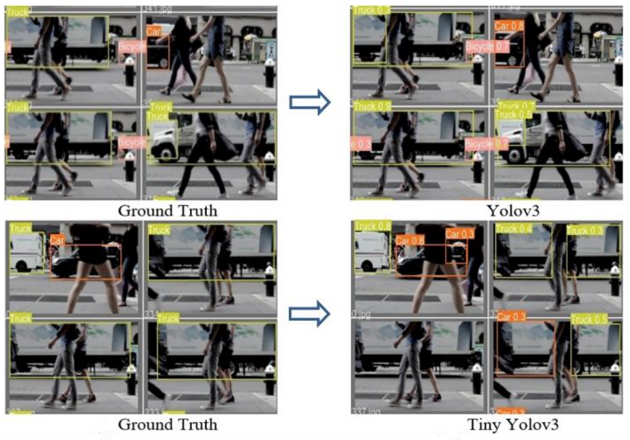

Figure. 12 Query object detection results of second video by YOLOv3-based detection and Tiny YOLOv3-based detection

videos dataset. Three experiments were achieved to show the performance evaluation for each model. The quantitative results of object detection accuracy were performed on five types of query objects. The evaluation metrics *Precious*, *Recall* and *mAP@.5* were calculated and recorder as shown below;

Experiment 1:

In this exterminate, the testing video composes of 372 frames including 239 ground truth frames. The quantitative results of object detection accuracy were

Table 7. The quantitative results of object detection accuracy performed on third video

	Category	Precision	Recall	Ap@.5
YOLOv3	Motorcycle	1	0.904	0.93
	Bicycle	0.818	0.534	0.662
	Car	0.925	0.63	0.786
	Bus	0.967	0.986	0.99
	Truck	0.98	0.906	0.958
	All	0.938	0.792	**mAP@.5 = 0.865**
Tiny YOLOv3	Motorcycle	0.951	0.919	0.927
	Bicycle	0.426	0.31	0.309
	Car	0.662	0.432	0.502
	Bus	0.942	0.982	0.979
	Truck	0.934	0.892	0.928
	All	0.783	0.707	**mAP@.5 = 0.729**

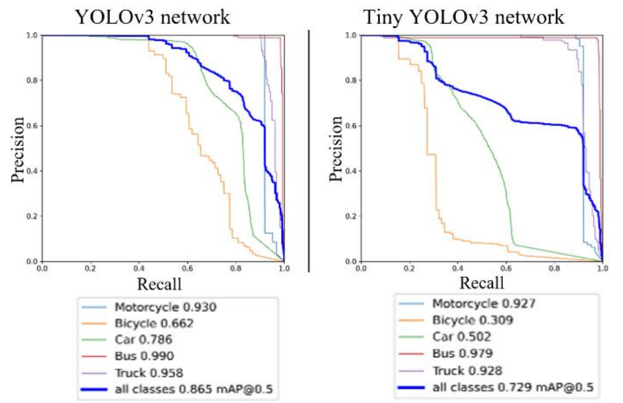

Figure. 13 Visualization results of object detection accuracy performed on third video by (YOLOv3 network and Tiny YOLOv3 network)

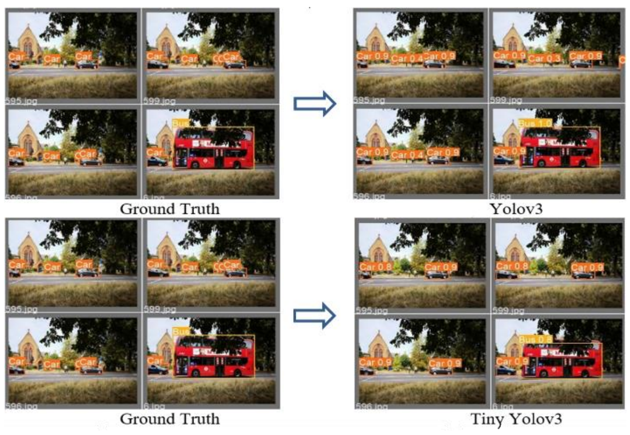

Figure. 14 Query object detection results of third video by YOLOv3-based detection and Tiny YOLOv3-based detection

Table 8. The *RS* results based on the original K-mean clustering, the modified K-mean clustering and the modified K-mean with bias algorithms

Videos	Frames	RS based on original K-mean [37]		RS based on modified K-mean		RS based on modified K-mean and bias	
		Key frames	RS	Key frames	RS	Key frames	RS
1	2027	1351	0.67	1153	0.57	675	0.33
2	1387	887	0.64	752	0.54	402	0.29
3	1108	736	0.66	629	0.57	333	0.30
4	1230	818	0.67	699	0.57	377	0.31
5	1262	840	0.67	717	0.57	389	0.31

stated in Table 5. Based on YOLOv3 network, we have obtained *mAP@*.5 = 0.978, while the detection accuracy based on Tiny YOLOv3 network was *mAP@*.5 = 0.964. The visualization results of object detection accuracy are shown in Fig. 9 and Fig. 10.

Experiment 2:
In this experiment, the testing video composes of 393 frames including 390 ground truth frames. The detection accuracy based on YOLOv3 network was *mAP@*.5=0.94, while Tiny YOLOv3network has achieved *mAP@*.5=0.79 as shown in Table 6 and Fig. 11 and Fig. 12.

Experiment 3:
In this experiment, the testing video composes of 882 frames including 822 ground truth frames. The detection accuracy based on YOLOv3 was mAP@.5 = 0.86, while Tiny YOLOv3 has achieved mAP@.5 = 0.72 as shown in Table 7. and Fig. 13 and Fig. 14.

Based on the previous experiments we have founded that; The test results (experiments-1, experiments-2, and experiments-3) show what to expect when employing deep learning in video summarization. As a result of experiments (1, 2, 3), we have founded that the greater distance between objects and camera, the more visible degradation in video sequence. In addition, the overlapping between the query object and the rest of video objects may induce distortion. These flaws result in lower detection accuracy and prediction of keyframes.

9.6 Clustering evaluation

In order to evaluate the performance of the modified K-mean clustering scheme proposed in our framework, we have fulfilled several experiments on five videos with various scenarios and time duration. These video sequences acquired by fixed camera and included normal movement of vehicles. The summarization rats *RS*s based on the original K-mean clustering algorithm and the modified K-mean clustering as well as the modified K-mean with bias algorithms are illustrated in Tables 8.

Table 9. Illustrates the experiment results of video-1

Video-1 length	53.76s	No. of Frames	1613		
Video Summarization using YOLOv3 network and modified K-means algorithm							

Object	GT frames	Original K-mean clustering		Modified K-mean clustering with bias		Time (s)	
		KF	RS	KF	RS	CPU	GPU
M	1613	916	0.57	454	0.28	401	41
B	340	202	0.13	49	0.03	378	30
C	1375	774	0.48	406	0.25	398	40
Bu	356	223	0.14	81	0.05	374	30
T	331	209	0.13	175	0.11	373	30
Video Summarization using Tiny YOLOv3 network and modified K-means algorithm							
Object	GT frames	Original K-mean clustering		Modified K-mean clustering with bias		Time(s)	
		KF	RS	KF	RS	CPU	GPU
M	1592	906	0.56	446	0.28	74	29
B	216	148	0.09	48	0.03	56	13
C.	1051	558	0.35	354	0.22	65	21
Bu	283	161	0.10	61	0.04	57	14
T	82	55	0.03	46	0.03	54	12

Table 10. Illustrates the experiment results of video-2

Length Video	36.7s	Frames	1101
Video Summarization using YOLOv3 network and modified K-means algorithm			

Object	GT frames	Original K-mean clustering		Modified K-mean clustering with bias		Time (s)	
		KF	RS	KF	RS	CPU	GPU
M	163	92	0.08	77	0.07	251	21
C	936	539	0.49	246	0.22	256	28
B.	1101	626	0.57	298	0.27	626	29
T	976	567	0.51	278	0.25	261	28
Video Summarization using Tiny YOLOv3 network and modified K-means algorithm							
Object	GT frames	Original K-mean clustering		Modified K-mean clustering with bias		Time (s)	
		KF	RS	KF	RS	CPU	GPU
M	80	52	0.05	37	0.03	36	11
C	568	302	0.27	205	0.19	41	15
Bu	1093	620	0.56	296	0.27	50	20
T	617	351	0.32	163	0.15	42	16

Received: July 4, 2022. Revised: August 25, 2022.

Table 11. Illustrates the experiment results of video-3

Length Video		87s		Frames		2610	
Video Summarization using YOLOv3 network and modified K-means algorithm							
Object	**GT frames**	**Original K-mean clustering**		**Modified K-mean clustering with bias**		**Time(s)**	
		KF	**RS**	**KF**	**RS**	**CPU**	**GPU**
M	46	43	0.02	7	0.00	589	40
B.	1764	1045	0.40	546	0.21	612	56
C	2573	1474	0.56	745	0.29	631	62
Bu	197	113	0.04	37	0.01	583	47
T	126	71	0.03	21	0.01	580	45
Video Summarization using Tiny YOLOv3 network and modified K-means algorithm							
Object	**GT frames**	**Original K-mean clustering**		**Modified K-mean clustering with bias**		**Time (s)**	
		KF	**RS**	**KF**	**RS**	**CPU**	**GPU**
M	36	33	0.01	3	0.00	82	15
B	459	359	0.14	271	0.10	90	20
C	2500	1393	0.53	710	0.27	111	46
Bu	182	105	0.04	33	0.01	84	18
T	131	114	0.04	63	0.02	84	18

Table. 12 Comparison results of the proposed video summarization method against related works in term of F1-score metric

Model		**UTE dataset**				**Avg.**
		P01	**P02**	**P03**	**P04**	
		F1-Score				
Seg-DPP[9], [11]		35.6	43.67	25.26	18.15	30.92
SH-DPP[7], [9]		48.68	42.72	36.51	15.62	33.38
QC-DPP[9]		49.66	41.66	56.47	29.96	44.19
QC-3PA[13]		58.60	45.30	56.51	33.64	46.05
YOLOv3 and mathematical model[14]		64.65	64.70	50.30	48.60	57.06
Tiny-YOLOv3 and K-mean **(Ours)**	without bias	**66.06**	**55.93**	**56.46**	**48.06**	**56.62**
	with bias	**51.73**	**43.41**	**41.93**	**46.45**	**45.88**
YOLOv3 and K-mean **(Ours)**	without bias	**67.45**	**58.68**	**63.76**	**39.45**	**57.33**
	with bias	**53.49**	**43.87**	**45.58**	**37.45**	**45.09**

As shown in Table 8, we have obtained an efficient improvement of summarization rates RS based on the modified K-mean clustering algorithm compared to the original algorithm, and the effect of adding bias.

9.7 Video summarization evaluation

Comprehensive experiments were conducted in this research to evaluate the proposed query video summarization system based on YOLOv3 and Tiny YOLOv3 deep learning models. Three videos were used with various types of query objects (Motorcycle:M, Bicycle:B, Car:C, Bus:Bu and Truck:T) supported by ground truth frames GT and time duration. Tables 9, 10, and 11 illustrate the obtained results using the modified k-mean clustering for keyframes (KF) extraction process from video-1, video-2 and video-3 respectively. The main measurements used in our experiments are; No. of Keyframes KF, summarization rate RS and consuming time of both CPU and GPU.

Table 10 shows that there is a kind of balance for Tiny YOLOv3 in terms of not losing too many frames and also when there are fixed vehicles with no movement all the time (such as the query about bicycle and truck), making object detection take up the entire video time, making unsupervised video summarization the only way to summarize. Table 11 demonstrates the false alarms results in detection accuracy of motorcycle object, which increased the number of frames and resulted in the inclusion of un-queried data. The quantitative results illustrated in Tables 9, 10, and 11, have been showed that Tiny YOLOv3 outperforms YOLOv3 in terms of consuming time, regardless using GPU or not, allowing its work to be classified for real-time application. However, the detection accuracy based on YOLOv3 network is more accurate when supported by GPU.

9.8 Comparison study

The proposed method was compared with other frameworks used UTE dataset and F1-score in their evaluation. Table 12 illustrates the respective comparison F1-score metric for each of the four videos. We have obtained an average F1-Score close to 57.33% and 56.62% based on YOLOv3 and Tiny YOLOv3 networks respectively.

Theoretically, the models used in our comparison study (Seg-DPP, SH-DPP, QC-DPP and QC-3PA) have an architecture which takes a long time to learn temporal relations between video shots and queries. However, the performance evaluation of these works was not performed in real-world environment of

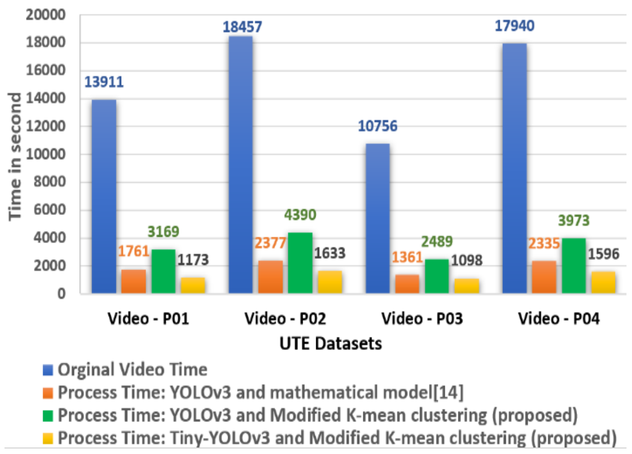

Figure. 15 The processing time results of the proposed framework using UTE dataset

video samples as our proposed framework was. As shown in Table 12, the proposed method achieved an average *F1-Score* not far from the results of research work [14] on UTE dataset, in which the authors were used the same resolution of input images. The resolution of all video frames of UTE dataset is (480×320 with frame rate =15 fps), which caused high distortion for small objects as well as interfere with other objects with varying brightness leads to difficultness for small objects detection. Obviously, YOLOv3 network detects well small objects, and the results of the three approaches were very close, despite the fact that the Tiny YOLOv3 network and the (YOLOv3 network used by [14]) make predictions with two scales only, while YOLOv3 makes predictions with three scales. From other hand, the processing time is 4 times faster than original video time when using YOLOv3 and 11 times faster than original video when using Tiny YOLOv3. Fig. 15 shows the superiority of the proposed method when using Tiny YOLOv3 for object detection task over other methods.

10. Conclusions

In this paper, we have presented a query-based video summarization system using deep learning models (YOLOv3 and Tiny YOLOv3). Through experiments and analysis, we have exhibited the accuracy detection of query objects based on each model. Set of video sequences were used to train, validate and test the proposed system. As a result, YOLOv3 network achieved superior detection accuracy close to 93%, while Tiny YOLOv3 network achieved superior in processing time denoted by 11 times faster than original video time. Furthermore, we have proposed a modification on K-mean clustering algorithm to extract the keyframes from the original input video. The modified algorithm used a dynamic selection of *K* value with an equivalent bias value for all the clusters. Comprehensive experiments were conducted on various standard video dataset and real world video sequences, which exhibited the efficacy and superiority of the proposed method. Efficient results have been obtained related to object detection accuracy and key frames extraction task. We have obtained video summarization rate close to 33% of the original video. The experiments on UTE dataset exhibited the outperform of the proposed method against state of art query based video summarization methods with average F1-Score close to 57.33% and 56.62% based on YOLOv3 and Tiny YOLOv3 networks respectively. In addition, the implementation of the proposed framework was achieved and evaluated in parallel technology with real world video data respectively, which minimize the consuming time of video summary construction. Relative to future work, we'll focus on improving key frames selection approach, and attempting to explore a new method to automatically detect shots boundary with different challenges.

Conflicts of interest

The authors declare no conflict of interest.

Author contributions

This work is a contribution of the authors: “Conceptualization, Saif K. Jarallah and Sawsen Abdulhadi Mahmood; methodology, Saif K. Jarallah and Sawsen Abdulhadi Mahmood; software, Saif K. Jarallah; validation, Saif K. Jarallah; formal analysis, Saif K. Jarallah and Sawsen Abdulhadi Mahmood; writing-original draft preparation, Sawsen Abdulhadi Mahmood; writing-review and editing, Saif K. Jarallah and Sawsen Abdulhadi Mahmood; visualization, Saif K. Jarallah; supervision, Sawsen Abdulhadi Mahmood.

Acknowledgments

This research was sponsored by Computer Science Department in College of Education-Mustansiriyah University located at Baghdad-Iraq. It was funded by Mustansiriyah University.

References

[1] H. Sheng, K. Yao, and S. Goel, “Surveilling Surveillance: Estimating the Prevalence of Surveillance Cameras with Street View Data”, In: *Proc. of the 2021 AAAI/ACM Conf. on AI, Ethics, and Society*, New York, United State, pp. 221–230, 2021.

[2] Z. Cheng, J. Lv, A. Wu, and N. Qu, “YOLOv3 Object Detection Algorithm with Feature Pyramid Attention for Remote Sensing Images”, *Sensors and Materials*, Vol. 32, No. 12, pp. 4537–4558, 2020.

[3] K. Muhammad, T. Hussain, and S. W. Baik, “Efficient CNN Based Summarization of Surveillance Videos for Resource-Constrained Devices”, *Pattern Recognition Letters*, Vol. 130, pp. 370–375, 2020.

[4] S. Gupta and T. U. Devi, “YOLOv2 Based Real Time Object Detection”, *International Journal of Computer Science Trends and Technology*, Vol. 8, 2013.

[5] S. V. Viraktamath, M. Yavagal, and R. Byahatti, “Object Detection and Classification Using YOLOv3”, *International Journal of Engineering Research & Technology (IJERT)*, Vol. 10, No. 2, 2021.

[6] H. Oosterhuis, S. Ravi, and M. Bendersky, “Semantic Video Trailers”, *arXiv*, arXiv:1609.01819, 2016.

[7] A. Sharghi, B. Gong, and M. Shah, “Query-Focused Extractive Video Summarization”, In: *Proc. of the European Conf. on Computer Vision*, Amsterdam, Netherlands, pp. 3–19, 2016.

[8] Z. Ji, Y. Ma, Y. Pang, and X. Li, “Query-Aware Sparse Coding for Multi-Video Summarization”, In: *Proc. of the ACM International Conf. on Multimedia*, California, United State, pp. 582–590, 2017.

[9] A. Sharghi, J. S. Laurel, and B. Gong, “Query-Focused Video Summarization: Dataset, Evaluation, and a Memory Network Based Approach”, In: *Proc. of 2017 IEEE Conf. on Computer Vision and Pattern Recognition,* Honolulu, United State, pp. 4788–4797, 2017.

[10] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus, “End-To-End Memory Networks”, *arXiv*, arXiv:1503.08895, 2015.

[11] B. Gong, W. L. Chao, K. Grauman, and S. Fei, “Diverse Sequential Subset Selection for Supervised Video Summarization”, In: *Proc. of the Advances in Neural Information Processing Systems*, Montreal, Canada, pp. 2069–2077, 2014.

[12] Y. J. Lee, J. Ghosh, and K. Grauman, “Discovering Important People and Objects for Egocentric Video Summarization”, In: *Proc. of the IEEE Computer Vision and Pattern Recognition*, Providence, Rhode Island, pp. 1346–1353, 2012.

[13] Y. Zhang, M. Kampffmeyer, X. Liang, M. Tan, and E. P. Xing, “Query-Conditioned Three-Player Adversarial Network for Video Summarization”, *arXiv,* arXiv:1807.06677, 2018.

[14] N. Baghel, S. C. Raikwar, and C. Bhatnagar, “Image Conditioned Keyframe-Based Video Summarization Using Object Detection”, *arXiv*, arXiv:2009.05269, 2020,

[15] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation”, *arXiv*, arXiv:1311.2524, 2013.

[16] K. He, X. Zhang, S. Ren, and J. Sun, “LNCS 8691 - Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition”, *arXiv*, arXiv:1406.4729, 2014.

[17] Z. Q. Zhao, P. Zheng, S. Xu, and X. Wu, “Object Detection with Deep Learning: A Review”, arXiv, arXiv:1807.05511, 2018.

[18] R. Girshick, “Fast R-CNN”, *arXiv*, arXiv:1504.08083, 2015.

[19] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition”, *arXiv*, arXiv:1409.1556, 2014.

[20] S. Li, S. Zhang, J. Xue, H. Sun, and R. Ren, “A Fast Neural Network Based on Attention Mechanisms for Detecting Field Flat Jujube”, *Agriculture*, Vol. 12, No. 5, p. 717, 2022.

[21] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, *arXiv*, arXiv:1506.01497, 2015.

[22] J. Dai, Y. Li, K. He, and J. Sun, “R-FCN: Object Detection Via Region-Based Fully Convolutional Networks”, *arXiv*, arXiv:1605.06409, 2016.

[23] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition”, In: *Proc. of the IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, Las Vegas, Unites State, pp. 770–778, 2016.

[24] Z. Cai and N. Vasconcelos, “Cascade R-CNN: High Quality Object Detection and Instance Segmentation”, *arXiv*, arXiv:1906.09756, 2019.

[25] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection”, *arXiv*, arXiv:1506.02640, 2015.

[26] J. Redmon and A. Farhadi, “YOLO9000: Better, Faster, Stronger”, In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR),* Honolulu, United State, pp. 6517–6525, 2017.

[27] Y. C. Huang, K. C. Hung, C. C. Liu, T. H. Chuang, and S. J. Chiou, “Customized Convolutional Neural Networks Technology for

Machined Product Inspection", *Applied Sciences (Switzerland)*, Vol. 12, No. 6, 2022.

[28] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. C. Berg, "SSD: Single Shot Multibox Detector", In: *Proc. of 14th European Conf. on Computer Vision*, Amsterdam, Netherlands, pp. 21–37, 2016.

[29] D. Bai, Y. Sun, B. Tao, X. Tong, M. Xu, G. Jiang, B. Chen, Y. Cao, N. Sun, and Z. Li, "Improved Single Shot Multibox Detector Target Detection Method Based on Deep Feature Fusion", *Concurrency and Computation: Practice and Experience*, Vol. 34, No. 4, 2022.

[30] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement", *arXiv*, arXiv:1804.02767, 2018.

[31] O. Masurekar, O. Jadhav, P. Kulkarni, and S. Patil, "Real Time Object Detection Using YOLOv3", *International Research Journal of Engineering and Technology*, Vol. 7, No. 3, 2020.

[32] V. Mazzia, A. Khaliq, F. Salvetti, and M. Chiaberge, "Real-Time Apple Detection System Using Embedded Systems with Hardware Accelerators: An Edge AI Application", *IEEE Access*, Vol. 8, pp. 9102–9114, 2020.

[33] Z. Yi, S. Yongliang and Z. Jun, "An Improved Tiny-Yolov3 Pedestrian Detection Algorithm", *International Journal for Light and Electron Optics (Optik)*, Vol. 183, pp. 17–23, 2019.

[34] D. Xu and Y. Wu, "Improved YOLO-V3 with Densenet for Multi-Scale Remote Sensing Target Detection", *Sensors*, Vol. 20, No. 15, pp. 1–24, 2020.

[35] S. Wang, "Research Towards YOLO-Series Algorithms: Comparison and Analysis of Object Detection Models for Real-Time UAV Applications", *Journal of Physics: Conf. Series*, Vol. 1948, No. 1, 2021.

[36] T. Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollar, "Microsoft COCO: Common Objects in Context", *arXiv*, arXiv:1405.0312, 2014.

[37] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. P. Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov, T. Duerig, and V. Ferrari, "The Open Images Dataset V4: Unified Image Classification, Object Detection, and Visual Relationship Detection at Scale", *IJCV*, 2020.

[38] B. Joseph, I. Joseph and D. Frese, "Pexels: Online Video", *Pexels*. www.pexels.com, 2021.

[39] T. Lin, "LabelImg", *github.com/tzutalin*, 2021.

[40] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin and A. A. Kalinin, "Albumentations: Fast and Flexible Image Augmentations", *Information (Switzerland)*, Vol. 11, No. 2, 2020.

[41] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning", *Journal of Big Data*, Vol. 6, No. 1, 2019.

[42] S. Rul, H. Vandierendonck, and K. D. Bosschere, "A Profile-Based Tool for Finding Pipeline Parallelism in Sequential Programs", *Parallel Computing*, Vol. 36, No. 9, pp. 531–551, 2010.

[43] I. T. A. Lee, C. E. Leiserson, T. B. Schardr, J. Sukha, and Z. Zhunping, "On-the-Fly Pipeline Parallelism", In: *Proc. of the twenty-fifth annual ACM symposium on Parallelism in algorithms and architectures*, Montreal, Canada, pp. 140–151, 2013.

[44] P. S. Pacheco and M. Malensek, "Parallel Hardware and Parallel Software (Second Edition)", *Morgan Kaufmann*, Cambridge, United States, pp. 17–88, 2022.

[45] M. Z. Hossain, M. N. Akhtar, R. B. Ahmad, and M. Rahman, "A dynamic K-means clustering for data mining", *Indonesian Journal of Electrical Engineering and Computer Science*, Vol. 13, No. 2, pp. 521–526, 2019.

[46] R. Padilla, W. L. Passos, T. L. B. Dias, S. L. Netto, and E. A. B. D. Silva, "A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit", *Electronics (Switzerland)*, Vol. 10, No. 3, pp. 1–28, 2021.

[47] Z. Li, G. M. Schuster, and A. K. Katsaggelos, "MINMAX Optimal Video Summarization", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 15, No. 10, pp. 1245–1256, 2005.

[48] M. Otani, Y. Nakashima, E. Rahtu, and J. Heikkilä, "Rethinking the Evaluation of Video Summaries", *arXiv*, arXiv:1903.11328, 2019.