Edge-based Video Analytic for Smart Cities

Dipak Pudasaini[1], Abdolreza Abhari[2]
Department of Computer Science
Ryerson University
Toronto, Canada

***Abstract*—Video analytic is the important tool for smart city development. The video analytic application requires more memories and high processing devices. The problems of cloud-based approach for video analytic are high latency and more network bandwidth to transfer data into the cloud. To overcome these problems, we propose a model based on dividing the jobs into smaller sub-tasks with less processing requirements in a typical video analytics application for the development of smart city. The object detection, tracking and pattern recognition method to reduce the size of videos based on edge network will be proposed. We will design a video analytic model, and simulation is performed using iFogSim simulator. We will also propose Convolutional Neural Network (CNN) based object tracking model. The experimental verification shows that our tracking model is more than 96% accurate, and the proposed edge and cloud-based model is more than 80% effective than only cloud-based approach for video analytic applications.**

Keywords*—Video analytic; cloud computing; smart city; object detection; object tracking; edge network

I. Introduction

Smart city is a city that uses technologies to provide the sophisticated lifestyle for humans. It provides improvement in transportation, accessibility, social services, sustainability, and other services. The smart cities have several types of technologies such as Information and Communication Technology (ICT), connected physical devices using the Internet of Things (IoT), Geographical Information System (GIS), Video Analytic System (VAS) and more.

IoT plays the important roles for the development of smart cities. IoT is used to input and transmit large volumes of data such as video, audio, text, etc. The suitable infrastructures are needed for the processing of large volumes of data from IoT devices to the processing devices. Therefore, edge computing and cloud computing technologies are the important concepts for the development of smart cities to process video data.

Edge network is a networking environment that focuses on bringing computing closer to the data source. It is the local processing technique near the Internet of Things (IoT) devices. It is the emerging technology used in many fields such as video analytics, machine learning, robotics and more. Edge computing is a helpful technique to solve the challenges of high latency and bandwidth consumption.

The combination of fog/edge computing architecture with IoT devices and the cloud computing is a very important research area for smart cities to minimize the resources and providing optimization for the users' benefits. The extension of cloud computing towards the IoT devices is called fog/edge computing. It is the middle layer between cloud layer and IoT layer. The fog computing consists of low processing servers or terminals with small storage capacity. It has limited physical resources in terms of storage, memory, and processing power [1]. Cloud computing architecture is the centralized architecture to store and process a huge amount of data. Edge computing is an open platform to store and process data at the edge of the network. Video analytics applications are examples of applications that uses edge computing.

Video analytic is a kind of analytic system that can be used to process and analysis the video files. Video analytic can be used for motion detection, facial recognition, license plate reading and more. The video data are excessively available in social media, traffics, film industry etc. The powerful technology is needed to process these data. Therefore, the combination of edge computing and cloud computing technology is the more powerful technology to process video data. In this research, we will propose video analytic system to process video data for smart city development. The object detection, tracking and pattern recognition methods are more important phases of video analytic system. We will propose the framework of object detection, tracking and pattern recognition of videos using Convolutional Neural Network (CNN). We will also propose the CNN based object tracking model.

The rest of the paper is organized as follows: Section II presents problem statements and contributions. Section III presents the literature review. Section IV describes the details of our proposed approach. The experimental results and simulation are explained in Section V and Section VI. Finally, Section VII presents the conclusion of a paper.

II. Problem Statements and Contributions

In traditional video analytic system, video data from the data source is directly transferred into the cloud where video frames are extracted, and objects are detected and analyzed [2]. The traditional cloud based centralized approach has suffered from high latency and more network bandwidth when transfer data into the cloud. The bandwidth usage problem's solution is to develop models that integrate the IoT devices with edge and cloud devices. Another problem is more uses of network resources in the existing approach. Since addressing these problems in a real system is very expensive or sometimes impossible, the known methodology to examine these problems' solutions is the simulation. The sample framework of dividing video analytic into subtasks was presented in [3] but was not simulated. In this proposed work, we define the details about video analytic pipeline,

prototyping model and parameter feed directly into the simulator. The video analytics jobs are huge applications referred to as edge computing killers [4]. We address this problem by assuming different tasks for a common video analytics application. The problem of video analytic application is that it requires more processing time and network bandwidth to transfer large files into the cloud. Therefore, the solution of this is to divide the video analytic system into more phases and reduce the size of video in the consecutive phases. We divide the video analytic into four phases which are motion detection, object detection, object tracking and pattern recognition. Then we propose the CNN method to reduce the video in consecutive phases based on edge computing architecture.

In a common video analytics application, there are many object tracking methods. Some of these are just tracking, and some are tracking by detection. Some of these methods are based on CNN, and some are not. The tracking methods without using CNN are faster but have low accuracy [5]. The CNN based tracking methods are more accurate, but the execution time is high [6]. In this research, we will modify the layers of the existing CNN model to decrease the execution time of the tracking model. We will also propose object detection, tracking and pattern recognition model using CNN based on Edge network.

The main contributions of this study are as follows:

- Dimensionality reduction: Proposing model for dividing video analytic application in different tasks by dimension reduction which means dividing them based on the processing requirement. The video analytic application consists of a number of phases such as motion detection, object detection, object tracking etc. We will purpose a model for dimensionality reduction in each consecutive phase of video analytic application.
- Object detection and tracking method: An object tracking module is a separate part of video analytics. There are the large number of object detection and tracking techniques for moving objects. We will use standard model for the detection of the objects, then modify the existing object tracking architecture using CNN to reduce the execution time of tracking.
- Verification of object Tracking method: We will experimentally verify our tracking method by using public video files and real time videos.
- Verification of model using iFogSim: The proposed model will be verified using iFogSim simulator. It will provide the effectiveness of using edge and cloud in our model in comparison with only the cloud-based architecture.

III. Related Work

The uses of fog computing in smart cities have been explained in [7]. The service oriented middle wire to reserve the issues of smart city development has proposed in [8]. It has presented the effective integration and utilization of Cloud of Things (CoT) and fog computing. Edge computing focuses on bringing the services and utilities of the cloud computing closer to the user for fast processing. The cost-effective technique for aerial surveillance in which large computation tasks are in the cloud and limited computation task in Unmanned Aerial Vehicle (UAV) devices using edge computing technique has been proposed [9]. The frames with normal behaviors are processed into edges and the frames with abnormal behaviors are passed into the cloud for abnormal behaviors detection. The simulation framework for the modelling of IoT and edge computing has been proposed [10]. It has extended the capacity of CloudSim to address the features of edge and IoT devices. The integration of edge and cloud computing with distributed deep learning for smart city IoT has been proposed [11]. It developed the hybrid model to optimize the system utility and bandwidth allocation.

The CNN-based framework for multi-object tracking has been proposed in [12]. It used RoI-pooling to obtain individual features for each target. In this method, spatial-temporal attention of the target is learned online to deal draft caused by occlusion. In [13], deep neural based appearance feature for multi-object tracking has been proposed. An algorithm for multi-object tracking was used for online and offline tracker. The real time object detection and tracking using deep learning OpenCV has been proposed [14]. It used Single Shot Detector (SSD) with mobile net framework for object detection and tracking. The fast vehicle detection based on evolving convolutional neural network has been proposed [15]. Tetris has proposed to provide maximum parallel processing of videos on a single GPU [16]. It has performed CPU-based tiling of active regions to combine the activities of video input. It ran the deep learning model and improved the GPU utilization.

In [17], the multiple objects tracking method with correlation filter has been proposed. In this method, the SSD was used for multi-object detector and CNN was used for tracking the objects. The real time object recognition model by using deep CNN to extract deep features has been proposed [18]. A multi-level three-dimensional convolutional neural network for the recognition of moving objects has been proposed [19].

IV. Methodology

A. Fog Computing Architecture for Smart Cities

Edge-cloud technology is a very important technology for wide geographical areas. The storage and processing of services in centralized based cloud approach provide more latency and bandwidth. We will use IoT-Edge-Cloud technology to support mobility with minimal overhead cost. The IoT-Edge-Cloud architecture is defined in Fig. 1. It consists of three tiers. The end devices such as sensors are considered as first tire. The fog/edge devices near the source are considered as second tire. The cloud devices joined with fog devices and far from the IoT devices are considered as third tire. The combination of three tires provides IoT-Fog-Cloud technology.

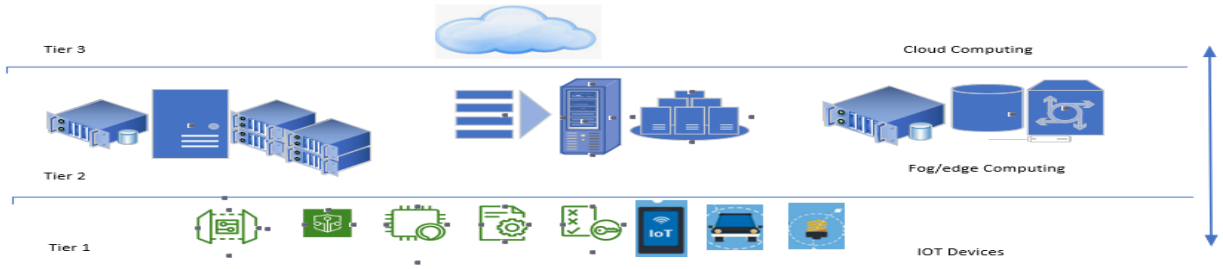

Fig. 1. Fog Computing Architecture for Smart Cities.

In the generic architecture, the IoT layer receives the input from the first tire. The fog layer consists of terminals, small servers, routers, access points, gateways and more [20]. This is an intermediate layer connected between IoT and the cloud. Cloud is the final layer in which data are transferred from fog layers. It has mass storage and processing capacity.

We propose a method for video analytic system to provide dimensionality reduction for object detection and tracking based on edge computing architecture. Edge devices are responsible to process the videos captured by cameras then object detection and tracking are taken place. Then the trajectories are sent to the cloud for pattern recognition. The testcase scenario of our proposed model is explained in section VI.

B. *Object Detection and Tracking Model for Video Analytic using CNN*

We will recommend the real video analytic application for object detection and tracking in this section. These real programs will be recommended in edge devices in our model. For the detection of the objects, we will use CNN-based object detection method YOLO. For the tracking of the objects, we will use deep sort and our own appearance model based on residual network. The trajectories created from tracking method are passed into the standard machine learning algorithms for pattern recognition.

1) Proposed object detection, tracking and pattern recognition model pipeline: The pipeline for this model consists of three stages: labelling stage, learning stage and prediction stage. In the labelling stage, the raw image data are annotated. Similarly, in the learning stage we fit different machine learning models on the data. And finally, we use the fitted Machine Learning (ML) model in the prediction stage. Since we will use two different models: Object Detection model and Appearance Model, in our prediction, we will apply labelling and learning stages separately for object detection and appearance model as illustrated in Fig. 2.

a) Labelling Stage: The labelling stage is the data annotation phase. Human annotators take in the raw data and annotate the data for the specific tasks. Since we have two models: object detection model and appearance model, we have two labelling stages where data gets annotated separately. For the detection model, human annotators take in the raw images and annotate the bounding boxes for the objects present and the corresponding categories of the objects. As a result, we get object detection annotation files. Similarly, for the appearance model, the human annotators take in frames from raw video data and annotate for object re-identification. They associate the objects with the same identities with a common id. This results in our annotated re-identification files.

b) Learning Stage: In the learning stage, a data pipeline gets created which takes in labelled images and the annotation files and creates datasets for the corresponding tasks. These datasets are then augmented randomly to increase robustness of the models and reduce overfitting. Then, different machine learning models with different architectures with varying numbers of parameters are learnt and validated by feeding in the data pipeline. The models which perform well on the validation sets are dumped to the disks. As a result, we have models with varying architectures and varying numbers of parameters which have different computational requirements. Based on the problem criterion, we choose the best model and mark it as the selected model for the prediction phase.

c) Prediction Stage: The prediction stage is the final stage. Here, we feed in the video frames and generate the final tracking results. First, we pass the video frames to a detector model, which we have learned from the learning stage. Then, we take in the predictions from the object detector to an association and tracking model. This appearance model performs deep association by using the appearance model we've learned earlier. Then the output from the tracking model is passed for pattern recognition.

In our proposed edge-based model, we will recommend using YOLO and updated deep Simple Online and Realtime Tracking (SORT) for object detection and tracking in edge levels. The architecture of this model is presented in Fig. 3. The frames of the video file are passed to YOLO method. We will use only vehicles class to reduce the timing of the detection method. Then YOLO provides classes and the bounding box. The bounding box is again passed to the deep association metric with residual network CNN architecture for object tracking. Finally, trajectory data are passed for pattern recognition. In this architecture, the dimensionality of the original data is reduced at each stage, which is another contribution of our model.

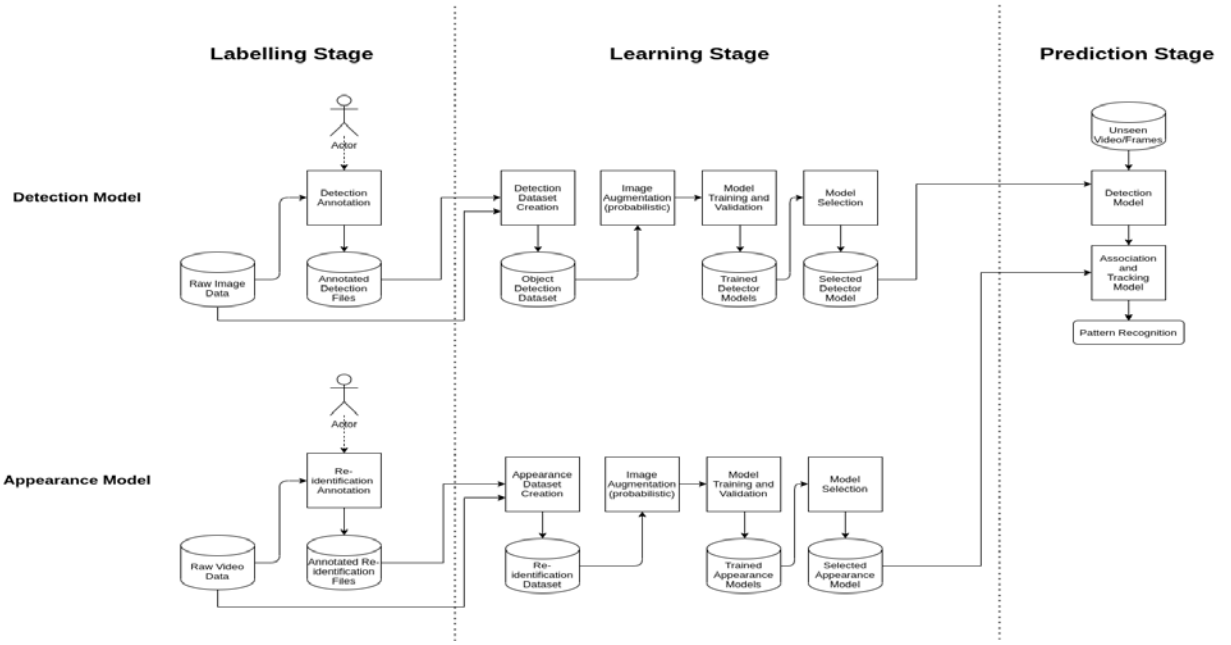

Fig. 2. Object Detection, Tracking and Pattern Recognition Model Pipeline.

2) Object detection model: YOLO is the object detection technique. The architecture of the YOLO is under the regression problem. In [21] [22], an image in the form of pixel values is the input of YOLO and the vector of bounding boxes with class predictions is the output. When the image inputs in the form of pixels, it passes through the neural network similar to CNN, then the vectors of bounding boxes and class predictions are in the form of output. The network uses the entire image to predict each bounding box. The image is divided into the SxS grid that grids are responsible for the detection of the objects. Each grid cell predicts *B* bounding boxes as well as *C* class probabilities. The bounding box prediction has 5 components: *(x, y, w, h, confidence)*, where *(x, y)* coordinates represent the centre of the bounding box, (w, h) represents the width and height of the bounding box. The confidence score is the score of predicting the object in a box. The YOLO is implemented in CNN using PASCAL VOC dataset. There are mainly two stages in YOLO. During the first stage, convolutional layers are used to extract the features from the image. During the second stage, the fully connected layers are responsible to provide the output probabilities and coordinates. It consists of 24 convolutional layers followed by 2 fully connected layers. The convolutional layers are pretrained in ImageNet dataset that used Darknet framework. The layers are presented in Table I.

The main strength of YOLO is speed. It is best object detection algorithm for fast detection. The weakness is more localization errors compared to faster R-CNN. The detection accuracy is less for very small objects. In this research, we are using object detection at edge level. It has light version and tiny version. Therefore, it is suitable for small processing edge devices. Another reason of using YOLO in this work is because of fast processing speed.

TABLE I. DARKNET-53 CONVOLUTIONAL NETWORK USED BY YOLOV3

	Type	Filters	Size	Output
	Convolutional	32	3x3	256x256
	Convolutional	64	3x3/2	128x128
1x	Convolutional	32	1x1	
	Convolutional	64	3x3	
	Residual			128x128
	Convolutional	128	3x3/2	64x64
2x	Convolutional	64	1x1	
	Convolutional	128	3x3	
	Residual			64x64
	Convolutional	256	3x3/2	32x32
8x	Convolutional	128	1x1	
	Convolutional	256	3x3	
	Residual			32x32
	Convolutional	512	3x3/2	16x16
8x	Convolutional	256	1x1	
	Convolutional	512	3x3	
	Residual			16x16
	Convolutional	1024	3x3/2	8x8
4x	Convolutional	512	1x1	
	Convolutional	1024	3x3	
	Residual			8x8
	Avgpool			Global
	Connected			1000
	Softmax			

We will use only vehicles classes to reduce the timing of the object detection model. We will use light version and standard version of the YOLO for the object detection. We will recommend light and standard YOLO depends upon the capacity of edge devices.

3) Object tracking using modified deep SORT: Deep SORT is a real time object tracking method [6]. It is an updated version of SORT. It integrates an appearance model which provides the deep appearance features for the detected objects in each frame. The method called the association of detected objects as a deep association metric. The inclusion of deep association metric allowed objects to be tracked in case of longer occlusions too and also the number of misclassifications were highly reduced. We use our modified version of deep SORT in this pipeline.

The pipeline for the deep SORT association is shown in Fig. 3. Video frames are fed into a robust object detector model. The object detector outputs the class categories and bounding boxes for the objects of interest in the video frames. Then, using the bounding boxes, the regions of interest in the frames are cropped and passed to the appearance model. The appearance model outputs the feature description for each detected object. The Deep SORT method leverages these feature descriptions for association of the detected objects with tracks in addition to the previous SORT based association, which uses the Kalman filter for predicting the location of the objects in the next timestamp.

The CNN architecture of our proposed architecture is shown in Table II. We use the residual network architecture as base line architecture [23]. In our proposed architecture, first two convolutional layers are used then pass into six residual blocks with same patch size and different stride. Then the outputs are passed into another convolutional layer. We employ a wide residual network with three convolutional layers and six residual blocks. In dense layer 11, the global feature map of dimensionality 128 is computed. This network is suited for online tracking.

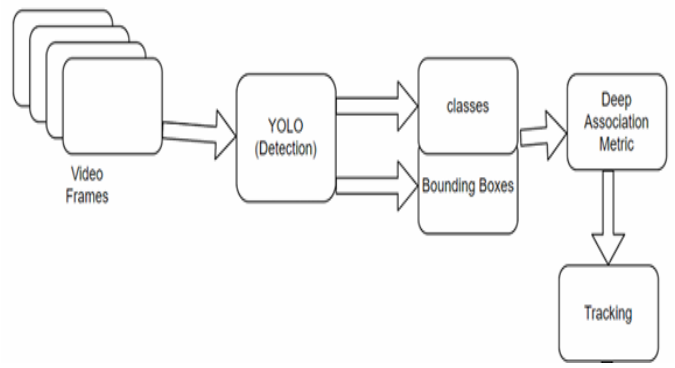

Fig. 3. Object Tracking Pipeline.

TABLE II. CNN ARCHITECTURE OF TRACKING

Name	Patch size/Stride	Output size
Conv 1	3 × 3/1	32 × 128 × 64
Conv 2	3 × 3/1	32 × 128 × 64
Max Pool 3	3 × 3/2	32 × 64 × 32
Residual 4	3 × 3/1	32 × 64 × 32
Residual 5	3 × 3/2	64 × 32 × 16
Residual 6	3 × 3/1	64 × 32 × 16
Residual 7	3 × 3/2	128 × 16 × 8
Residual 8	3 × 3/1	128 × 16 × 8
Conv 9	3 × 3/2	256 × 8 × 4
Flatten 10		8192
Dense 11		128
Batch and l2 normalization		128

4) Pattern recognition: The tracking model generates the trajectories of the moving objects. The large numbers of trajectories collected from tracking model are passed to machine learning algorithms for pattern recognition. The scalable pattern recognition of moving objects has been proposed in [5]. The supervised and unsupervised machine learning algorithms have been used for the pattern recognition of the trajectories.

V. OBJECT DETECTION AND TRACKING RESULTS

The object detection and tracking of our proposed model was implemented in a machine with an ubuntu operating system. It was implemented and tested on Intel Core I5-7300U CPU @2.6GHz with 16GB RAM. The programming platform for detection and tracking the objects was Python 3.7.6 and the OpenCV library. The Python with anaconda environment was used for its implementation. Other tools used in this research were Pytorch and TensorFlow.

1) Dataset: The model was trained using marketplace dataset for bounding box [24]. The Market-1501 dataset was collected with six cameras. The dataset contains 12937 images for training datasets and 19732 images for testing dataset. For the varification, it used any types of videos on of social media or stored in database or can create new videos. Mainly, the YouTube videos were collected, and tracking was done in real time to test this system.

The sample pictures of online tracking are shown in Fig. 4 and Fig. 5. Each figure shows the moving objects at a time and rectangle around objects shows travelling of motion of objects.

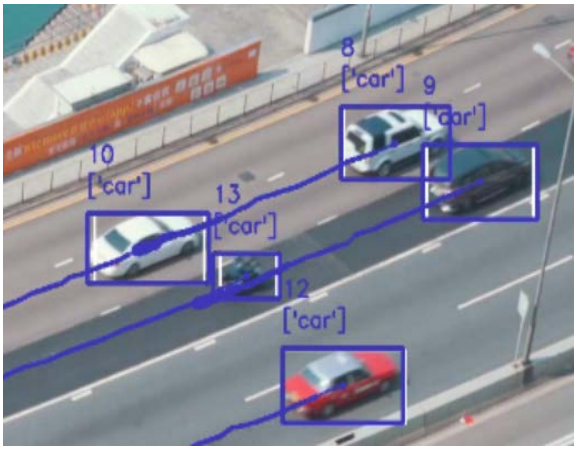

Fig. 4. Object Tracking Result Sample 1.

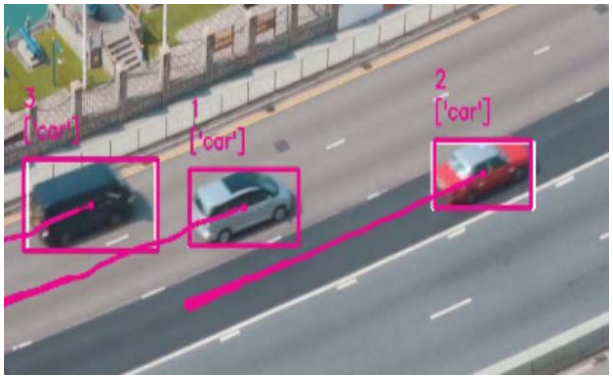

Fig. 5. Object Tracking Result Sample 2.

One objective of this research is the dimensionality reduction in each successive phase. The dimensionality of data in each phase is extremely reduced in our proposed model. The proof of dimensionality reduction has presented in Table I and Table II in methodology chapter. When the frames of the video are passed into proposed model then size of output data is reduced in each layer.

We also present the dimensionally reduction of input video in Table III. First input video is passed into YOLO method for object detection that produces the dimension reduced bounding box. The bounding box is passed into tracking method that produces the trajectories in the form of text data. In this experiment, we passed similar quality of video data.

TABLE III. DATA SIZE REDUCTION

Video number	Input size (kb)	No of frames	After detection	After tracking (kb)
Video1	526	122	Bounding boxes	12.3
Video2	720	140	Bounding boxes	16.8
Video3	655	131	Bounding boxes	15.0
Video4	912	185	Bounding boxes	25.1

TABLE IV. RECALL, PRECISION AND TRACKING ACCURACY OF DIFFERENT VIDEOS

Based on objects in videos	TP	FP	FN	Recall	Precision	Accuracy
Number of Vehicles in Highway	500	8	9	0.98	0.98	98
Number of Vehicles in city road	450	12	9	0.98	0.97	98

TABLE V. COMPARISON OF EXECUTION TIME OF PROPOSED TRACKING WITH YOLOV3 TINY AND YOLOV3

Type of YOLO	Average video size (KB)	Execution time (seconds)
YOLOv3	512	320
YOLOv3 tiny	512	80

TABLE VI. COMPARISONS OF EXECUTION TIME OF THE TRACKER

Tracking	Average video size (MB))	Execution time fps (frame per second)
POI [25]	1.8	0.18
Baseline Deep sort [6]	1.8	0.72
Improved Deep sort (Proposed Model)	1.8	0.75

The proposed model is trained by using marketplace data set and the performance is calculated on test data. We use two versions of YOLO that depends upon the size of edge devices which was defined on previous unit. The light YOLO goes to the small size edge device and the normal YOLO goes to the large size fog device. The execution time of the light YOLO is less than normal YOLO that is presented in Table V. The tracking accuracy of this model is around 98% which is explained in Table IV. The execution time of our proposed tracking model is faster than current deep sort model and Person of Interest (POI) model which is explained in Table VI.

VI. TEST SCENARIOS, SIMULATION AND COMPARISON OF RESULTS

A. Test Scenarios

In this research, we use video surveillance system of moving vehicles as test scenario for smart city. If we pass video files directly into the cloud, more memories are needed. In addition, centralized process takes more latency and uses more network bandwidth. Therefore, processing of videos in decentralized methods are more advisable methods nowadays. In our method, we will extremely reduce the size of video files, then pass to the cloud for further processing. We will recommend decentralized method for the processing of video files using edge computing architecture.

The overall performance of our proposed model will be measured by iFogSim simulator. We have four stages in the video analytic system: motion detection (s1), object detection (s2), object tracking (s3), and pattern recognition (s4) shown in Fig. 6.

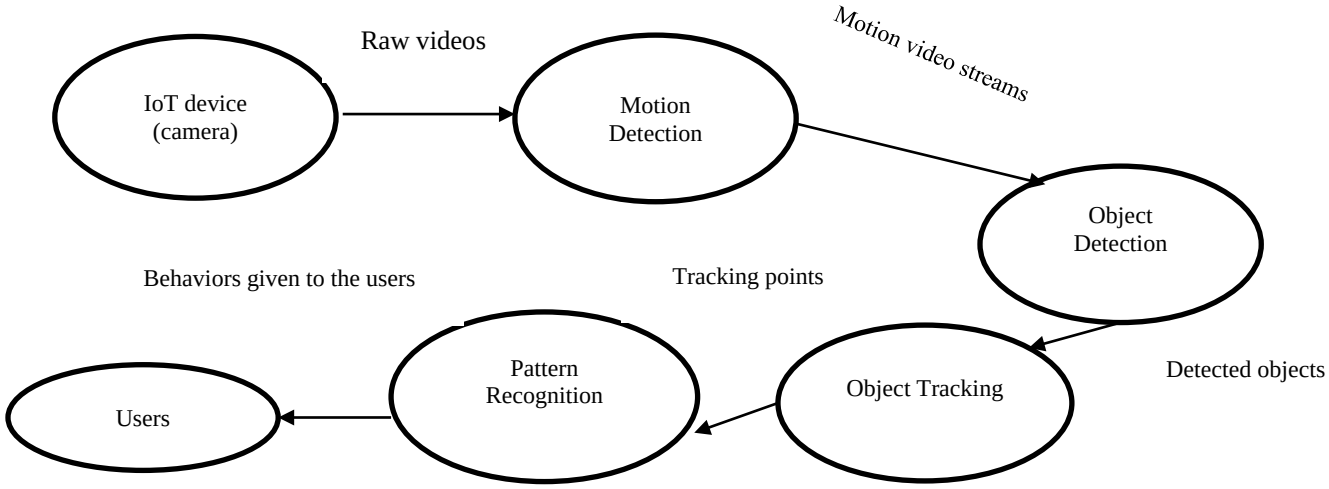

Fig. 6. Video Analytic Pipeline Scenario.

Motion detection (s1): The video camera continuously captures the raw video stream to detect the motion then forwarded to an object detection module.

Object detection (s2): The object detection module receives the video streams of detected motion from the motion detection module. This module is responsible to detect the objects then pass into the object tracking module.

Object tracking (s3): The object tracking module receives the results from object detection module. Then, the object tracking module tracks the path of moving objects and pass into pattern recognition module.

Pattern recognition (s4). It receives the tracking path from object tracking module. It is responsible to find the pattern of moving objects then recommend the patterns to the users.

The unique part of this work is to divide the video analytic into tasks which works as a pipeline. It is very close to real system because the output of one module is the input of another module such as the output of object detection is the input of object tracking.

CASE 1: In this case, edge and cloud are used for video processing. The motion detection (s1) from video camera goes to the edges for object detection (s2) and object tracking(s3). Finally goes to the cloud for pattern recognition (s4).

CASE 2: In this case, only edges are used for video processing. All the stages of video analytic application which are motion detection (s1), object detection (s2), object tracking (s3), and trajectory pattern recognition (s4) are performed on edge devices.

CASE 3: In this case, only cloud is used for video processing. The detected motion (s1) from video camera directly goes to cloud for object detection (s2), object tracking (s3), and pattern recognition (s4).

B. Simulation Tool and Physical Topology

The iFogSim simulator is used in this research. iFogSim [26] is a discrete event simulator for simulation and modelling of edge/fog computing environment. It is based on the CloudSim simulator. In this paper, new model is simulated in which each module is used for monitoring and its output results is the input of another module i.e. pipeline. The simulation has been achieved using a personal computer with Windows 10 operating system. It has simulated on Intel Core I5 CPU @2.3GHz with 8GB RAM. The programming language java with eclipse has been used for the implementation.

The physical topology consists of the cloud data center at the top of the network called first tire. The second tire is the proxy server which is connected between cloud and fog devices. The fog devices are called third tire containing fog nodes. The number of fog devices can be added in different places depending upon the demand of applications. The number of video cameras is connected to the fog devices for our proposed video analytic model for intelligent surveillance application. The physical topology is presented in Fig. 7.

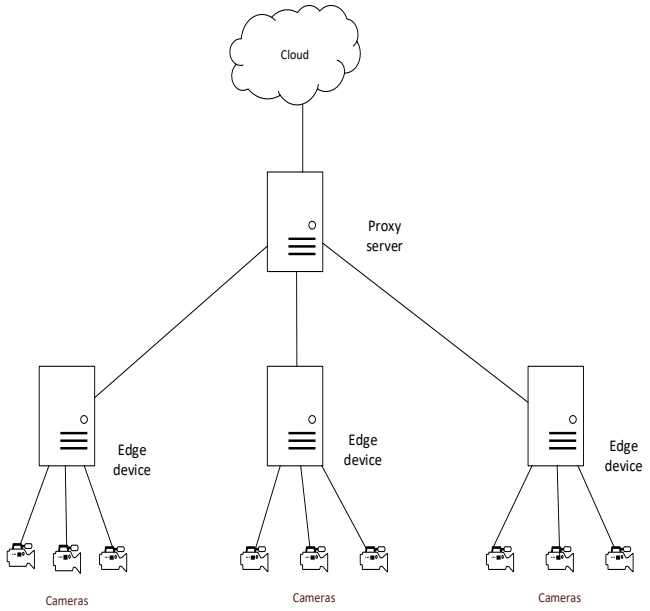

Fig. 7. Physical Experimental Set Up.

C. Defining Simulation Data

The video data is the input of video analytic system. The CNN-based object tracking method explained in previous chapter has video data as an input. For the simulation of video analytic model, we will input the video data similar to our CNN-based object tracking model into our simulator. The videos are not automatically read into iFogSim simulator. Therefore, in this simulation, moving vehicle video data first need to convert into pixel file. The data rate is bits per pixel i.e. bpp. This CSV file contains bpp of videos. There are 13 attributes which are tuple id, total number of pixels, dilation, x coordinate, y coordinate, z coordinate, frame difference, moving rate frame per second, motion ptz, contours, grey color, black color and final contours. There are 65535 rows of data on that file.

Based on the topology designed for the simulation, various fog devices are created and assigned on the nodes of the topology. The fog devices are represented as uniquely working nodes such as camera, edge device, proxy server and cloud. Each fog device has its own processing capacity and configuration like CPU, RAM, upload/download bandwidth, power consumption which setting is defined in Table IX. In this simulation, camera is the sensor that creates the tuples and passed into another device. The different application modules such as motion detection, object detection and so on are assigned to the fog devices according to their capacity. Following are the details about the device configuration parameters:

CPU (MIPS): It is the processing capacity of a CPU given on millions of instructions per second. Higher the processing capacity of a device, higher will be the task execution rate.

RAM: It is the temporary data storage medium where the processing data are stored while the device is online.

Up/Down Bandwidth: It is the speed of the device at which the data are uploaded and downloaded to and from the device.

Power Consumption: It is the electrical power in watt that a device consumes while operating.

The tuple CPU length is the size of data to be passed from one module to another module. The network length (bandwidth) is the rate of transfer of data from one module to another module. In case of object tracking module, after completed a tracking process, the size of data transferred into another module is tuple CPU length and transferred rate is network bandwidth which setting is defined in Table VII.

D. Parameter Settings and Network Configuration

The choice of configuration values is based on the minimum requirement of video surveillance in the real-world scenarios that is referred from the iFogSim Simulator. The Table VII below outlines the configuration of application module components in the video surveillance application. Table VIII presents the latencies configuration between the source and the destination nodes. It explains how the communication between the nodes is managed. Table IX describes the capacities of fog nodes and cloud. It presents the size of different devices in our physical topological structure.

TABLE VII. VALUES FOR MONITORING APPLICATION

Tuple types	Tuple CPU length (MIPS)	Network length (Bandwidth)
Raw video stream	1000	2000
Motion video stream	2000	2000
Detected objects	1000	100
Tracking points	1000	800

TABLE VIII. NETWORK LATENCIES CONFIGURATION

Between	Latency (Milliseconds)
Cloud to Proxy server	100
Proxy server to fog devices	50
Fog devices to camera	1

TABLE IX. CAPACITY OF FOG NODES

Device	CPU (MIPS)	RAM (Bytes)	Up bandwidth (Mbps)	Down bandwidth (Mbps)	Power consumption (Watt)
Cloud	40000	40000	1000	10000	450 (B) 250 (I)
Proxy server	2800	40000	10000	10000	200(B) 100 (I)
Fog device	2500	4000	1000	10000	100 (B) 83 (I)
Camera	500	1000	100	100	87(B) 82 (I)

E. Performance Evaluation

The evaluation of the performances in our proposed model is resource utilization, bandwidth, latency, and power consumption. These performances are measured in three test cases which was explained in the above section. Test cases are a) Using cloud and edges, b) Using only edges and c) Using only cloud. There are four configurations for simulation results. The number of areas and number of cameras in each place is varied on these configurations. The setting of the configuration is presented in Table X.

The network performance is calculated by the configuration of Table X. We test the results in three scenarios. The first scenario is testing our proposed model in the combination of fog devices and cloud. The second scenario is testing our proposed model in fog/edge devices only. The third scenario is testing our proposed model in cloud devices only. The four parameters of the performance matrix are measured. They are resource utilization, latency, bandwidth, and energy consumption. The resource utilization is the how much resources are utilized to process the data. The Latency/loop delay is the time taken for an application loop to execute. In the application, this loop starts with the camera sensors producing the video stream, goes through the motion detector, object detector, object tracking, and finally pattern recognition. The maximum amount of data which can be transmitted over the network on specific time is called bandwidth. Energy consumption refers to the amount of energy used to process in the system.

TABLE X. CONFIGURATION FOR SIMULATION

Configuration	Areas	Cameras
Config1	3	1
Config2	5	2
Config3	8	4
Config4	10	6

The results are explained in Fig. 8 to Fig. 11 in three scenarios cloud and edges, only edges, and only cloud. The Fig. 8 presents the comparison of network bandwidth. The edge and cloud architecture saved the network bandwidth by 81% in comparison with only cloud-based architecture. The Fig. 9 presents the comparison of resource utilization in these three scenarios. The edge and cloud architecture saved more resources which is around 88.3% in comparison with only cloud-based architecture. Fig. 10 describes the comparison of latency in these three scenarios. The edge and cloud architecture has less latency in comparison with only cloud-based approach; the latency has saved by 97.4%. Similarly, only fog-based approach is slightly better than the combination of cloud and fog devices. Fig. 11 presents the comparison of energy consumption in these three scenarios. The energy consumption in the system is around same for all scenarios.

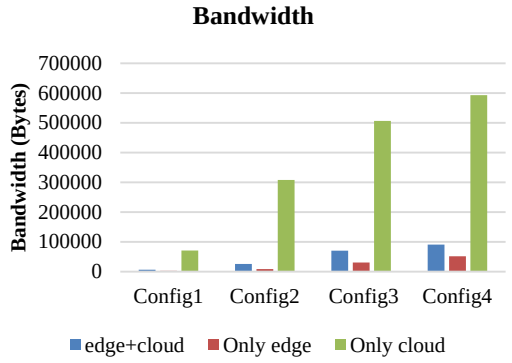

Fig. 8. Comparison of Bandwidth.

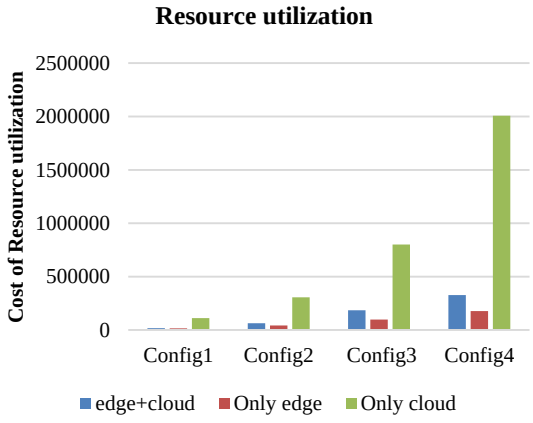

Fig. 9. Comparison of Resource Utilization.

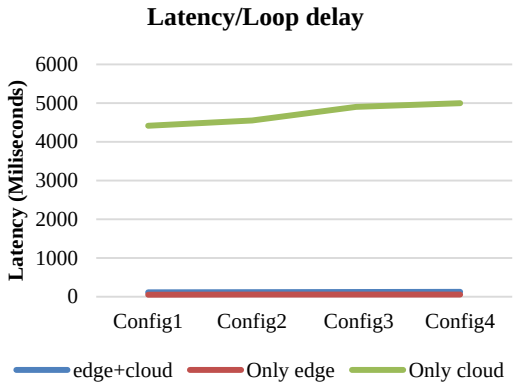

Fig. 10. Comparison of Latency.

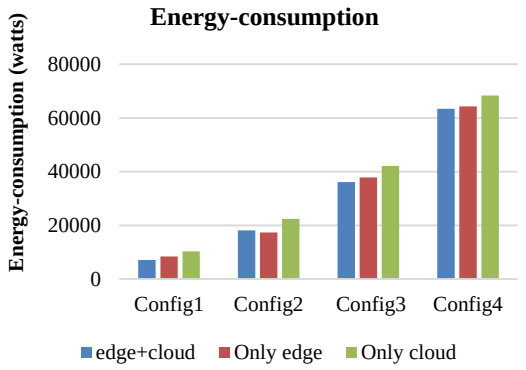

Fig. 11. Comparison of Energy Consumption.

VII. CONCLUSION AND FUTURE WORK

The combination of edge computing and the cloud computing is the main paradigm for video analytic system to build smart city application. In this paper, we developed the new approach for video analytic application to process the data in edge devices and the cloud for smart city in which modules are working as pipeline. We proposed the dimensionality reduction of data in the consecutive steps of video analytic application to increase the network performance. The proposed edge computing technique for video analytic will result in less traffic on the internet because only a small portion of the data will pass into the cloud. One contribution is separating the job into pipeline of sub-tasks and another contribution is implementing the sub-tasks by using deep learning methods. This research also proposed scalable object detection and tracking of moving objects based on CNN. The large numbers of moving vehicles can be tested by our prototype model. Dividing a video analytic job into pipeline of sub-tasks will help to process large number of videos with low latency and low network bandwidth and less cost of resource utilization. The experimental results show that proposed tracking by detection method is more than 96% accurate.

We simulated a proposed model using iFogSim, the result shows that latency, bandwidth, and resource utilization are 97.4%, 81% and 88.3% efficient than only the traditional cloud- based approach.

We simulated our model on large number of fog devices using iFogSim simulator, but we tested a smaller number of videos in CNN-based model. This is a limitation of this work.

The future work will improve the deep learning-based object detection and tracking method. We will improve the performance and compare with different Convolutional Neural Network architecture with large number of video files.

REFERENCES

[1] A. Yousefpour, G. Ishigaki, and J. P. Jue, "Fog Computing: Towards Minimizing Delay in the Internet of Things," in *Proceedings of IEEE 1st International Conference on Edge Computing*, pp.68–73, 2017.

[2] A. Anjum, T. Abdullah, M. F. Tariq, Y. Baltaci, and N. Antonopoulos, "Video Stream Analysis in Clouds: An Object Detection and Classification Framework for High Performance Video Analytics," *IEEE Transactions on Cloud Computing*, vol. 7, no. 4, pp. 1152-1167, 2019.

[3] D. Pudasaini and A. Abhari, "Scalable Object Detection, Tracking and Pattern Recognition Model Using Edge Computing," in *Proceedings of Spring Simulation Conference (SpringSim)*, pp. 1-11, 2020.

[4] G. Ananthanarayanan, P. Bahl, P. Bodik, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha, "Real-Time Video Analytics: The Killer App for Edge Computing," *IEEE*, vol. 50, no. 10, pp. 58-67, 2017.

[5] D. Pudasaini and A. Abhari, "Scalable Pattern Recognition and Real Time Tracking of Moving Object," *Spring Simulation Conference (SpringSim)*, pp. 1-11, 2019.

[6] N. Wojke, A. Bewley, and D. Paulus, "Simple online and real-time tracking with a deep association metric," in *Proceedings of IEEE International Conference on Image Processing*, pp. 3645-3649, 2017.

[7] G. Javadzadeh and A.M. Rahmani, "Fog Computing Applications in Smart Cities: A Systematic Survey," *Wireless Networking*, vol. 26, pp. 1433–1457, 2019.

[8] N. Mohamed, J. Al-Jaroodi, I. Jawhar, S. Lazarova-Molnar, and S. Mahmoud, "SmartCityWare: A Service-Oriented Middleware for Cloud and Fog Enabled Smart City Services," *IEEE* , vol.. 5 pp. 17576-17588, 2017.

[9] M. S. Alam, B. V. Natesha, T. S. Ashwin, and R. M. Guddeti, "UAV based cost-effictive real-time abnormal event detection using edge computing," *Multimedia tools and Applications*, vol. 78, pp. 35119-35134, 2019.

[10] D. N. Jha, K. Alwasel, A. Alshoshan, X. Huang, R. K. Naha, S. K.Battula, et al., "IoTSim-Edge: A Simulation Framework for Modeling the Behaviour of IoT and Edge Computing Environments," *arXiv:1910.03026*, pp. 1-19, 2019.

[11] H. Wu, Z. Zhang, C. Guan, K. Wolter, and M. Xu, "Collaborate edge and cloud computing with distributed deep learning for smart city internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8099-8110, 2020.

[12] C. Qi, W. Ouyang, H. Li, X. Wang, B. Liu, and N. Yu, "Online Multi-Object Tracking using CNN-Based Single Object Tracker with Spatial-Temporal Attention Mechanism," *International Conference of Computer Vision*, pp. 4836-4845, 2017.

[13] Y. Fengwei, W. Li, Q. Li, Y. Liu, X. Shi, and J. Yan, "POI: Multiple Object Tracking with High Performance Detection and Appearance Feature," *European conference of computer vision*, pp. 36-42, 2016.

[14] G. Chandan, A. Jain, H. Jain, and Mohana, "Real Time Object Detection and Tracking Using Deep Learning and OpenCV," in *Proceedings of* IEEE *International Conference on Inventive Research in Computing Applications (ICIRCA)*, pp.1305-1308, 2018.

[15] F. Zhu, Y. Lu, N. Ying, and G. Giakos, "Fast vehicle detection based on evolving convolutional neural network," *IEEE International Conference on Imaging Systems and Techniques (IST)*, pp. 1-4, 2017.

[16] T. Stone, N. Stone, P. Jain, Y. Jiang, K. Kim, and S. Nelakuditi, "Towards Scalable Video Analytics at the Edge," *Annual IEEE International Conference on Sensing, Communication, and Networking*, pp. 1-9, 2019.

[17] D. Zhao, H. Fu, L. Xiao, T. Wu, B. Dai, "Multi-Object Tracking with Correlation Filter for Autonomous Vehicle," *Sensors (Basel, Switzerland)*, vol. 18, no. 7, pp. 1-17, 2018.

[18] L. Yang, L. Wang, and S. Wu, "Real-time Object Recognition Algorithm Based on Deep Convolutional Neural Network," in *Proceedings of the 3rd IEEE conference on cloud computing and big data analysis*, pp. 331- 335, 2018.

[19] T. He, H. Mao, and Z. Yi, "Moving Object Recognition using Multi-View Three-dimensional Convolutional Neural Networks," *Neural Computing and Applications*, vol. 28 no. 12, pp. 3827-3835, 2017.

[20] L. M. Vaquero and L. Rodero-Merino, "Finding your Way in the Fog: Towards a Comprehensive Definition of Fog Computing," *Computer Communication Review*, vol. 44, no. 5, pp. 27-32, 2014.

[21] J. Redmon and A. Farhad, "YOLOv3: An Incremental Improvement," Retrieved from: https://pjreddie.com/media/files/papers/YOLOv3.pdf, 2018.

[22] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016, Retrieved from: https://arxiv.org/pdf/1506.02640.pdf.

[23] S. Zagoruyko and N. Komodakis, "Wide residual networks," *BMVC*, pp. 1–12, 2016.

[24] Retrieved from: https://www.kaggle.com/pengcw1/market-1501.

[25] F. Yu, W. Li, Q. Li, Y. Liu, X. Shi, and J. Yan, "POI: Multiple Object Tracking with High Performance Detection and Appearance Feature," *arXiv:1610.06136*, 2016.

[26] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in the Internet of Things, Edge and Fog Computing Environments," *Software Practice & Experience*, vol. 47, no. 9, pp. 1275- 1296, 2016.