IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received November 6, 2019, accepted November 20, 2019, date of publication December 30, 2019,
date of current version January 17, 2020.

Digital Object Identifier 10.1109/ACCESS.2019.2963159

mVideo: Edge Computing Based Mobile Video Processing Systems

HUI SUN[1], **YING YU**[1], **KEWEI SHA**[2], **(Senior Member, IEEE), AND BENDONG LOU**[1]
[1]School of Computer and Science, Anhui University, Hefei 230039 , China
[2]Department of Computer Science, University of Houston-Clear Lake, Houston, TX 77058, USA

Corresponding authors: Hui Sun (sunhui@ahu.edu.cn) and Kewei Sha (sha@uhcl.edu)

This work was sponsored in part by The University Synergy Innovation Program of Anhui Province (GXXT-2019-007), in part by the Natural Science Research Projects at Higher Institutions in Anhui Province (KJ2017A015), in part by the the National Natural Science Foundation of China under Grant 61702004, Anhui University Funding for Doctoral Research (J01003214), and in part by the Key Technology R&D Program of Anhui Province (1704d0802193).

ABSTRACT Computer vision is widely used to detect anomalies in video processing systems for public safety. Applying Deep Neural Networks (*i.e.,* DNNs) in computer vision can achieve a high detection accuracy but it requires a huge amount of computing power, storage space, and video data. Thus, DNNs-based video analytics is mostly deployed in the cloud with video data steaming from a set of stationary cameras. There are mainly three issues in this setting. First, steaming a huge amount of video data from cameras to cloud leads to high bandwidth consumption and latency. Second, when DNNs are deployed on resource-limited devices like edge nodes to reduce communication costs, it is hard to achieve a high detection accuracy. Third, stationary cameras can only collect a limited amount of video data that covers a small area, so it barely satisfies the needs of the real-time analytics in applications like public safety. We propose a mobile edge computing-based video stream processing platform, mVideo, which conducts video analytics making full use of resources at the collaborative edge and cloud nodes. On the mVideo, a mechanism is designed to partition a video analysis task based on available resources on the mobile edge node. Then, the edge nodes pre-process video data using a lightweight DNN model and upload the results to cloud nodes for further analysis. Thus mVideo not only collects video data that covers a large area, but also reduces the communication costs. To validate the proposed platform, a face recognition application is deployed on the mVideo prototype. Experimental results reveal that compared with the existing cloud computing model, mVideo reduces video data volume transmitted to the cloud nodes and power consumption by up to 99.5% and 96.2%, respectively. mVideo also improves the execution time by 90.0% to optimize mobile video analytics performance.

INDEX TERMS Mobile video analytics, edge computing, mobile cameras, public safety.

I. INTRODUCTION

Recently, researchers payed more attention to video processing in public safety [1]–[3]. A lot of object detection approaches, such as face recognition [4], bio-metric recognition [5], [6], gait detection [7], etc., are employed to monitor and identify individuals. For example, face recognition, action recognition, and license plate recognition are used in video analytics for public safety because of their availability and scalability [8]–[10]. Action recognition and license plate recognition are also employed to detect public places and traffic safety [8], [11], [12].

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Yu.

DNN-based algorithms achieve a higher detection accuracy [13], [14] in video analytics compared with traditional machine learning and image processing algorithms. To apply DNNs for video analytics in public safety, there are three models. The most popular approach is the cloud computing-based model [15], which trains DNNs using powerful cloud resources [4], [16], including GPU, FPGA, etc. This method provides a high detection accuracy because of the computing capability in the cloud with video data from a set of stationary cameras. There are some deficiencies in this model. Video cameras with blind spots cannot monitor the whole area. Besides, the important object information may not be obtained due to various angles in cameras. A large amount of video data from stationary cameras transfers to the cloud where the inference is executed, which burdens the network bandwidth and prolongs the latency of video processing. The large-scale video stream requires computing resources and time to process video in the cloud. Second, the DNNs are deployed on industrial personal computers (IPC) [4]. This approach offers reconstructive models for object detection and recognition for video analytics. However, the cost of hardware platform makes it expensive to deploy DNNs in a large-scale video processing system. Third, an embedded application specific integrated circuits (ASICs) implements the inference phase of the DNNs into the chip [17]. This approach achieves a smaller latency than others, but it is not only non-reconfigurable but also difficult for a wide deployment because of the inflexibility of hardware in ASICs.

To overcome these challenges, we propose a mobile video stream processing platform named **mVideo** based on the collaborative computing between the edge nodes and the cloud. mVideo includes the mobile edge nodes with cameras, the cloud, and the data transmission module. The video frame can be captured by the mobile edge node. A lightweight DNNs can be deployed between the edge nodes and the cloud to pre-process video stream. The intermediate result and status information in each node can be sent to the cloud by the transmission module.

In our proposed mVideo, the performance improvement are evaluated in terms of video data volume for the transmission, execution time, and power consumption. We conducted extensive experiments on mVideo platform to compare the performance of face detection and recognition in the cloud computing and edge computing models. Experimental results validate the benefits of mVideo in terms of the above three metrics. In detail, mVideo reduces the video data volume transmitted to the cloud nodes and power consumption by up to 99.5% and 96.2%, respectively. mVideo also improves the execution time by 90.0% to optimize mobile video analytics performance.

The contributions of the paper are summarized as follows.

- mVideo, a mobile video stream processing platform, is proposed to avoid blind spots and expand the monitoring area in the densely populated places. It is composed of a mobile edge computing unit with cameras and the cloud, which makes full use of the computing resources on the collaborative edge and cloud nodes.
- A framework that is designed based on the mVideo can conduct the partitioned lightweight DNNs in each node for video analytics. The intermediate result rather than raw video data are transmitted among the edge and the cloud nodes, It significantly reduces the communication cost compared with the cloud model.
- We take face recognition as an example and perform extensive experiments to validate the benefits of mVideo in terms of transmission data volume, execution time, and power consumption.

The rest of the paper is organized as follows. Section II discusses the background and motivation. Section III presents the framework of mVideo. Evaluation environment and experimental evaluation are revealed in Section IV. Section V presents related work. Further work and conclusion on our work are given in Section VI.

II. BACKGROUND AND MOTIVATION

A. BACKGROUND

In this section, we take the face recognition as an example to compare the performance of face detection and recognition in the edge computing and cloud computing models on our tested platform. This tested platform is composed of a mobile edge node and a cloud server. The edge node captures 4,000 video frames from a mobile camera and sends them to a cloud server via the Wi-Fi network. The frames vary with zero, two, and four faces. An access point (*i.e.,* AP) is employed in the Wi-Fi network. Two types of network bandwidth between the edge node and the cloud server (*i.e.,* high-bandwidth and low-delay network, and low-bandwidth and high-delay network) are configured in this experiment, the definition for these two types of network are listed as:

1) HIGH-BANDWIDTH AND LOW-DELAY NETWORK (HBLD)

The network bandwidth between the edge node and the cloud presents a high bandwidth and low delay for the video transmission.

2) LOW-BANDWIDTH AND HIGH-DELAY NETWORK (LBHD)

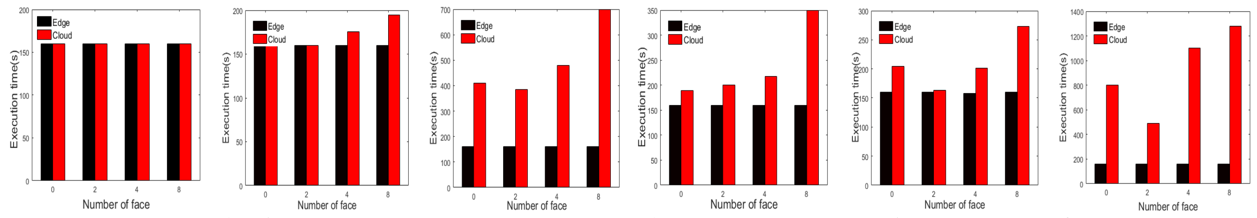
The network bandwidth between the edge node and the cloud presents a low bandwidth and high delay for the video transmission.

The process of face detection and recognition includes two sub-process, *i.e.,* face detection and face recognition. Face detection which is the first sub-processing in the recognition process is used to detect the location of a face in an image. Face recognition matches and extracts a person from the face database and identifies who he/she is. In our test, the multi-task cascaded convolutional networks (*i.e.,* MTCCN) [18] is for face detection to find candidates. Face recognition that consumes computational resources is conducted using the FaceNet [19].

Two types of computing models are employed in our test, *i.e.,* edge computing and cloud computing. In the cloud computing model, the mobile edge node captures video stream and sends to a cloud server through a transmission protocol (*e.g.,* Zero-MQ [20]). Then, the MTCNN and FaceNet are performed on the server. In the edge computing model, MTCNN and FaceNet are deployed and conducted in the mobile edge node. In addition, the cloud server collects the results and the status of nodes.

B. MOTIVATION

We conducted extensive experiments to study the performance of the cloud computing- and edge computing-based models for face detection and recognition. In the cloud computing model, a large amount of video data is transferred to



(a) (320*240):HBLD (b) (640*480):HBLD (c) (1280*720):HBLD (d) (320*240):LBHD (e) (640*480):LBHD (f) (1280*720):LBHD

FIGURE 1. Execution Time in the HBLD- and LBHD-based network and Different Resolutions of 4,000 Frames. The notation (640*480 : HBLD) means the execution time of face recognition under the resolution 640*480 in the HBLD-based network. *Cloud* and *Edge* represents execution time in the cloud computing- and edge computing-based model, respectively.

the cloud for face detection and face recognition, because the mobile edge node cannot pre-process video data. This method introduces a high overhead on the network bandwidth. In the edge computing model, the MTCNN and FaceNet are performed for face detection and face recognition at the edge node, respectively. This method avoids the transmission of large-scale video data between the edge node and the cloud server. In addition, it must be noted that the size of video data varies with its resolutions because different resolution consumes various amounts of computing resources in video analytics. Thus, the performance of face detection and recognition varies in the cloud and at the edge node, respectively. Three types of video resolution (*i.e.,* 1280*720, 640*480, and 320*240) are used in this test.

Fig. 1 presents the cloud- and edge-model execution time under resolutions 320*240, 640*480, and 1280*720 in the HBLD- and LBHD-based network. In the edge model, the execution time is between 160 ms and 550 ms in the two types of networks. In contrast, the execution time in the cloud model is between 160 ms and 1200 ms. To some extent, performance in the cloud model is more sensitive to network conditions.

In the HBLD network, the execution time of the cloud and edge-model is about 175.42 ms and 317.405 ms under resolution 640*480 and four-face cases, respectively, see Fig. 1(c). When network conditions get worse, the advantages of the edge model becomes obvious. Under resolution 640*480 and four faces, the execution time of the cloud model is approximately 1.2x of that in the HBLD-based network while it is basically unchanged in the edge model in the LBHD-based network, see Fig. 1(e). Meanwhile, under resolution 1280*720, the performance of the edge model is always better than that of the cloud model.

Compared with the edge model, a large amount of video data is sent to the cloud in the cloud model. The overhead of the network is high, which hardly provides a real-time result. In the edge model, face detection and recognition is offloaded to the edge node. This method reduces the volume of video streams that is sent to the cloud, and it can achieve better performance than the cloud model in most cases. We find that the edge model has obvious advantages over the cloud model when the network condition is not ideal by reducing network transmission. It motivates us to exploit an edge-cloud

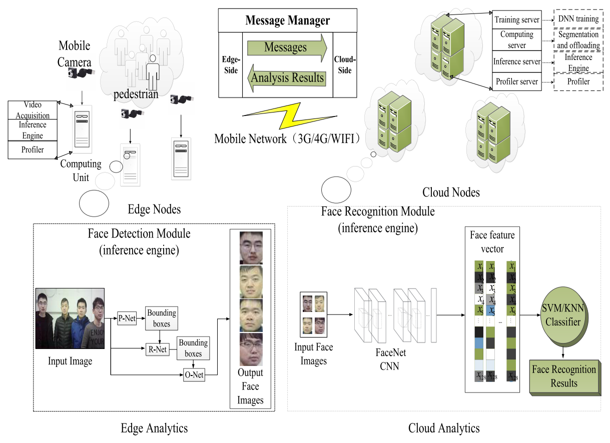


FIGURE 2. Overview System of mVideo Framework.

collaboration for video analytics (*e.g.,* face recognition) in public safety where we can make full use of the computing power of edge devices to reduce network bandwidth.

III. FRAMEWORK OF mVideo

In this section, we present the details of components of our proposed mVideo platform and its implementation.

A. DESIGN OF mVideo

Our proposed mVideo is an edge computing-based platform that includes the edge and cloud nodes. The video analytics can be conducted in real time on this platform. The DNN models with various sizes and functions could be deployed in these nodes to improve the performance of video analytics. The overview of mVideo is shown in Fig. 2.

An edge node is composed of a mobile camera and an edge computing unit. The edge computing unit that can be an edge device (*e.g.,* Raspberry Pi) captures video stream from the connected mobile camera. Then, data stream can be processed locally or directly transmitted to the cloud through 3G/4G/WiFi network for further video analysis.

The cloud nodes that are distributed on the mVideo conduct data processing. The function of a cloud node varies with its computing resources for multi-level video analytic. Thus, DNN models can be partitioned and offloaded to each node to conduct video analytic based on its available computing resources. The intermediate result from the edge level node can be processed locally. Then, the new result is sent to a cloud node for being further processed. Meanwhile, each node returns the analysis result to the edge and the cloud nodes in real time.

In the cloud, there are four types of servers for data analytics including a training server, an inference server, a computing server, and a profiler server. The training server can train DNN models for various multi-level nodes at the offline mode. The inference server conducts the models' inference for video processing. The computing servers partition and offload DNN models to an appropriate edge or cloud node at the online mode. The profiler server collects the status of each node on the mVideo platform, (*e.g.,* CPU usage, performance, power consumption, bandwidth, etc.) and store them in a server. Using these servers conducts, tasks of video analytics can be dynamically distributed among collaborative nodes to improve the system performance. Furthermore, the feedback information in the cloud can also be processed and returned to a node.

The mVideo supports several types of networks, *e.g.,* Wi-Fi, 4G, etc. The transmission module in each node conducts data compression and recovery. The processed video stream is transmitted among nodes. The transmission protocol on the mVideo platform is critical for the performance of data analytics. It is important to guarantee data transmission reliability and real time.

B. mVideo IMPLEMENTATION

We present the implementation of the software stack on each node of the mVideo in this section.

An edge node can pre-process video data by means of computer vision algorithms and DNN models. It is composed of four components including video acquisition, inference engine, profiler, and message manager.

In the video acquisition module, a mobile camera connects to the computing unit via the USB, Type-C, Bluetooth, etc. The mobile camera follows a protocol and requires a universal driver to be installed in the computing unit. When an application uses video data, the computing unit can capture video streams from the camera for a preview. The inference engine of a lightweight DNN model conducts inference on video streams at local. Generally, a lightweight DNN model is deployed based on the resource on the edge node. On the mVideo, face detection is performed on captured video streams using the MTCNN model. A profiler module can collect the status of an edge node (*e.g.,* CPU usage, bandwidth, data volume, power consumption, etc.) and send status information to the cloud node at intervals, see Fig. 4. Meanwhile, it receives a database including objective face images for the inference model from the cloud in the real time. The message manager is responsible for the data receiving/sending at local. The result (*e.g.,* feedback message) is sent to a cloud node.

The video processing for face detection using the MTCNN module is shown in Fig.3. The video pre-processing component primarily reverses the format and size of frames and configures the frequency of MTCNN on captured video

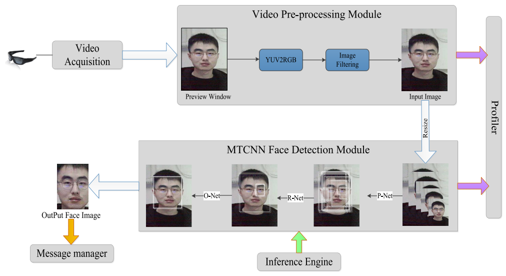


FIGURE 3. Video Pre-Processing at the Edge Node on the mVideo Platform.

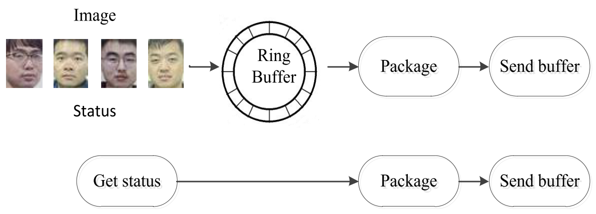


FIGURE 4. Two-type Data at the Edge Node for transmission.

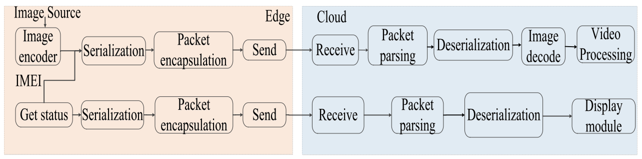
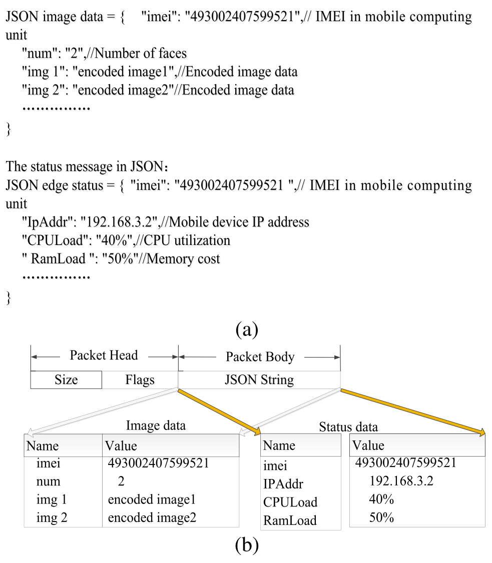


FIGURE 5. Transmission Component on the mVideo platform.

images. The candidate images are processed by three cascaded lightweight CNNs in this module. The detected face images and status (see Fig. 4) are packaged and uploaded to the next node. The data transmission at the edge node is shown in Fig.5.

The transmission component deployed at the edge and cloud guarantees the availability and scalability for data transmission among nodes on the mVideo, see Fig. 5. There are two types of data transmission among nodes on the mVideo, *i.e.,* video data from a mobile camera and edge device status, see Fig. 4. The process of data transmission mainly includes serialization, packet encapsulation, transmission (send/receive), packet parsing, deserialization. In the cloud model, video streams is pushed to the cloud for processing. However, mVideo distributes data processing on each node. Thus, the communication among nodes becomes important. First, the bandwidth and latency for data transmission between two nodes determine different types of networks. A mobile edge node can communicate with other nodes via 4G or Wi-Fi network. The performance of the whole system is dependent on the quality of the network among two nodes. Second, a scalability and high-performance network transmission protocol is also important for mVideo. On the mVideo, the Zero-MQ message protocol is employed in the system according to requirements and scenarios in real-world applications.

For the transfer data, the JavaScript Object Notation (*i.e.,* JSON) format is used for serialization. The transmitted



```
JSON image data = {    "imei": "493002407599521",// IMEI in mobile computing
unit
    "num": "2",//Number of faces
    "img 1": "encoded image1",//Encoded image data
    "img 2": "encoded image2"//Encoded image data
    ...............
}

The status message in JSON:
JSON edge status = { "imei": "493002407599521 ",// IMEI in mobile computing
unit
    "IpAddr": "192.168.3.2",//Mobile device IP address
    "CPULoad": "40%",//CPU utilization
    " RamLoad ": "50%"//Memory cost
    ...............
}
```

(a)

Packet Head		Packet Body
Size	Flags	JSON String

Image data

Name	Value
imei	493002407599521
num	2
img 1	encoded image1
img 2	encoded image2

Status data

Name	Value
imei	493002407599521
IPAddr	192.168.3.2
CPULoad	40%
RamLoad	50%

(b)

FIGURE 6. JSON Message of an Image (Fig.6(a)). The Format of Message in Transmission System (Fig. 6(b)).

messages are divided into two types, *i.e.,* image data from the mobile camera and edge device status. The latter includes International Mobile Equipment Identity (i.e., IMEI), IP address, and CPU cost, memory usage, etc. The format of JSON that is used in our transmission system is listed in Fig.6(a). The head of the format (see Fig.6(b)) is composed of data size and the flag of the data type. The size area represents the length of the message. For the message that is no longer than 255 bytes, a byte is used to store metric and eight bytes are used to store the size after setting 0xff. The flag denotes the type of the data message, *e.g.,* full or partial data. The data body is set in the format of JSON string, which ensures the range of the data package according to the data size.

It must be noted that Zero-MQ has the following benefits on the mVideo comparing with TCP. Zero-MQ sends/receives data using the message unit rather than the byte-stream model in TCP protocol. The format of JSON serializes the data transferred to the cloud from the edge device. Zero-MQ can send messages by a middleware, which is conducive to the expansion of the network. On the mVideo, the transmission data include face images and edge device status. Thus, the Zero-MQ protocol that transmits data in the serialization mode can achieve accurate data transmission. In addition, mVideo can use the structure of the Zero-MQ protocol to extend the scalability of the whole system.

The **cloud node** not only manages the nodes on the mVideo but also conducts the large-scale inference model due to its huge of computing resources. DNN training, segmentation and offloading, inference engine, profiler, and message manager exist in this component. The DNN training component trains different models at an edge and a cloud node. It integrates algorithms into a microprogram which can run on each node on the mVideo. Due to the limited resources at an edge node, a segmented inference model runs in the node. Thus, the DNN segmentation component partitions the DNN model and trains it. It can configure various models for each device according to its available resources. The DNN offloading module updates DNN models and offloads them to different edge nodes. In the cloud, the inference engine component conducts the final processing for the DNN model that requires the computing and storage resources. Taking the face detection and recognition as an example, the edge node can conduct the inference model for face detection. The face recognition completes in a high-computing cloud server. Fig.7 presents the face recognition using FaceNet network in the cloud. This configurable component is composed of face feature embedding and SVM classifier. For face feature embedding, we convert the frame including face image into the feature vector. The distance between the feature vectors of different face images represents the similarity. In the SVM classifier, the feature vector of the face image is set as input parameters of the trained SVM classifier. to obtain the category of the candidate face image.

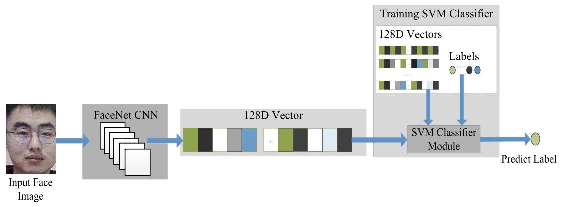


FIGURE 7. Face recognition in the cloud node on the mVideo platform.

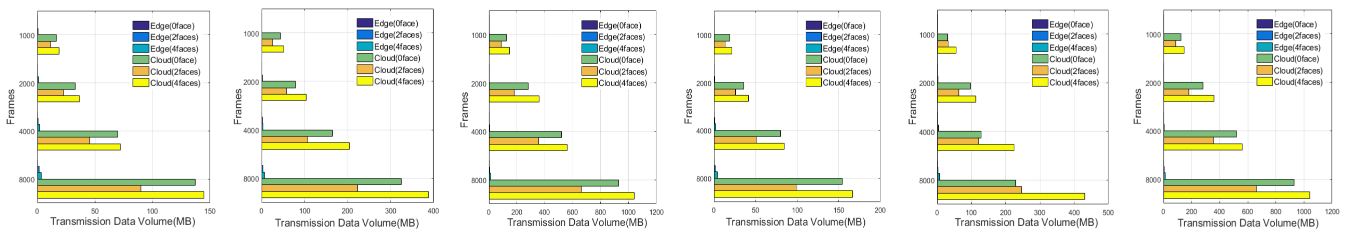
In the cloud node, the message manager receives the result from each device and performs real-time feedback to it. This module also offloads the updated information of the library to the edge node. When a suspicious face, vehicle, or abnormal behavior is detected by video analytics, a warning is sent to the edge node. It also updates the object features database on edge nodes.

IV. EXPERIMENTAL EVALUATION

In this experiment, we exemplify a face detection and recognition system on mVideo platform. A smart mobile phone (computing unit) with a mobile camera is used as an edge node. Meanwhile, we employ a cloud server as a cloud node. A Wi-Fi router is applied as the components of the access point on the mVideo platform. MTCNN and FaceNet models are employed for face detection and face recognition, respectively. Both edge computing- and cloud computing-based collaborative models are employed in this experiment. It must be noted that the notation *edge* and *cloud* represent edge-cloud collaborative computing and cloud computing for short, respectively. We study the benefits of mVideo in terms of video data volume for the transmission, execution time, and power consumption as follows.

A. EXPERIMENTAL SETUP

In the cloud model, the mobile edge node captures video data and sends it to the cloud without video pre-processing.



(a) (320*240):HBLD (b) (640*480):HBLD (c) (1280*720):HBLD (d) (320*240):LBHD (e) (640*480):LBHD (f) 1028*720:LBHD

FIGURE 8. Data Volume transmitted to the cloud from the edge in both HBLD and LBHD network under Different Resolutions. It must be noted that the notation (640*480: HBLD) means the transmission data volume under resolution 640*480 in the HBLD network.

The face detection and recognition is conducted in the cloud. We present the edge-cloud-based collaborative methods as the following steps.

Step 1. A mobile edge computing unit connects the camera to acquire video data and sends the face images to the cloud. These candidate images from the mobile cameras are obtained using the computing resources on the mobile computing unit. After face detection in local, the high quality of face image is determined according to the requirements for an angle, posture, lighting, etc. Then, the pre-processed face information is sent to the cloud node.

Step 2. In a cloud node, we use a server with a GPGPU to conduct face recognition on images by means of FaceNet model. The image with any face is sent by the edge node to the cloud node. A profiler in the cloud node collects the performance, status information, data transmission, and power consumption during the process of face detection and recognition.

Step 3. In the cloud, the offline and online modules are implemented in a cloud node. The offline module trains the models of MTCNN and FaceNet for face detection and recognition, respectively. The online collaboration module offloads the MTCNN and FaceNet models to the mobile edge node and the cloud server, respectively. The profiler in the cloud collects the status information in terms of video data volume for transmission, execution time, and power consumption from the edge and cloud nodes on the mVideo platform.

Besides, two types of network (*i.e.,* (1) the high-bandwidth and low-delay network; (2) the low-bandwidth and high-delay network) are configured in our test to study the impact of the network bandwidth on the above metrics. Note that our experiments focus on the performance of mVideo. Considering the wireless signal strength of the mobile edge node from the Wi-Fi access point, we fix the location of the cloud node and Wi-Fi access point and move the edge node in a range. Two types of networks (please refer to Section II.A for details) are used in this test.

B. PERFORMANCE EVALUATION

In the experiment, we present the performance of face detection and recognition on the mVideo in terms of video data volume for transmission, execution time, and power consumption. In this section, we refer edge computing- and cloud computing-based face processing as the edge model and cloud model, respectively.

1) DATA TRANSMISSION

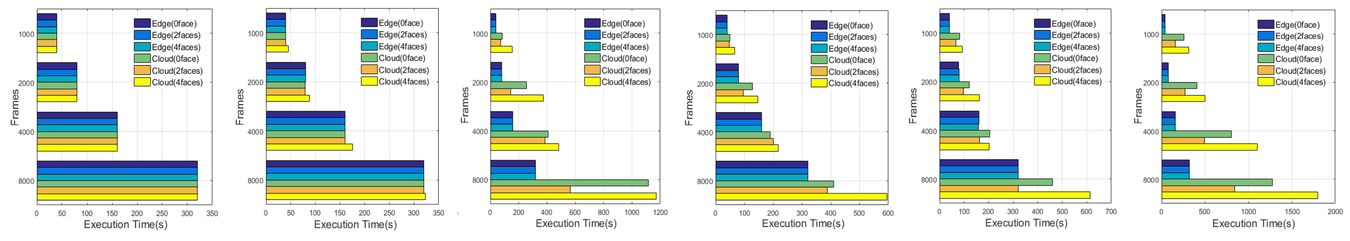
Fig. 8 shows the data transmission volume between the cloud and the edge under different networks. Notation *Edge(2face)* (Cloud(2face)) in figures represents that there are two faces in a candidate from the edge node (or cloud server) in the edge (or cloud) mode, respectively.

In the case of zero face in an image, the data transmission consumption mainly depends on the creation of the connection between the edge node and the cloud server. Thus, it is considered as the baseline in our experiment. The data size varies with the number of faces in a candidate image. We find that the data size of a frame with 0, 2, and 4 faces. In Fig. 8, there is an obvious gap between the edge and the cloud under three types of resolution and two types of network. The transmission data volume in the cloud is much more than that in the edge. In the HBLD-based network, the amount of data for processing 4 faces in 8,000 frames is about 7.34 MB in the edge under resolution 640*480, see Fig. 8(b). However, the cloud-based data volume increases to about 387.84 MB, which is 50x of that in the edge.

In Fig. 8(e), the results are close to that of the HBLD-based network. The reason is that the face image in a frame is extracted by the pre-processing module in the edge node before it is sent to the cloud server. The edge-based data transmission is significantly lower than that of the cloud because all video stream that is captured by the mobile camera is sent to the cloud node to conduct face detection and recognition. With the same set of frames, the higher the resolution is, the larger the amount of data will be transmitted in the same network.

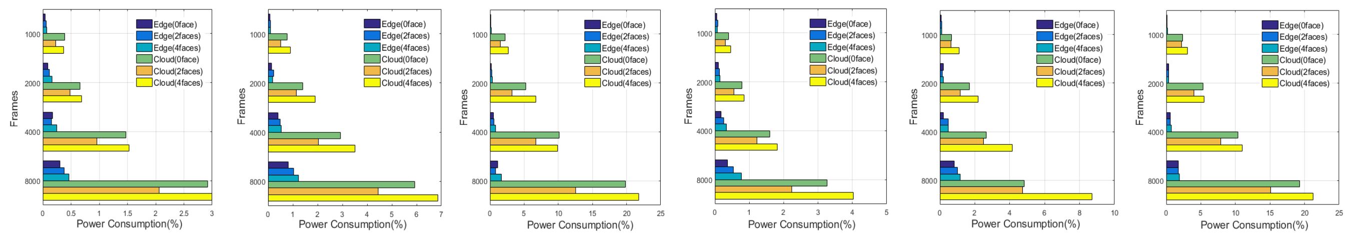
2) EXECUTION TIME

With the same set of frames, Fig. 9 shows the comparison of execution time in the edge and the cloud under different scenarios. Results reveal that the cloud-based execution time is always higher than that of the edge when having the same network, frames, and faces in a frame under different resolutions. In the HBLD-based network, the cloud-based execution time of face detection and recognition is similar to that in the edge under resolution 320*240 and



(a) (320*240):HBLD (b) (640*480):HBLD (c) (1280*720):HBLD (d) (320*240):LBHD (e) (640*480):LBHD (f) (1280*720):LBHD

FIGURE 9. Execution Time in HBLD and LBHD network under Different Resolutions. It must be noted that the notation (640*480 : HBLD) means the execution time in the HBLD network and resolution 640*480.



(a) (320*240):HBLD (b) (640*480):HBLD (c) (1280*720):HBLD (d) (320*240):LBHD (e) (640*480):LBHD (f) (1280*720):LBHD

FIGURE 10. Power Consumption in the HBLD- and LBHD-based network and Different Resolutions. It must be noted that the notation (640*480: HBLD) means the power consumption in the HBLD-based network and under resolution 640*480.

640*480, respectively. In these cases, video data analytics is processed in real time in both cloud and edge models. The reason is that the amount of data transmitted is small in the cloud. When the resolution increases, the cloud cannot achieve the same tendency. In Fig. 9, the video processing in the edge always achieves near real time. Under resolution 1280*720 in the HBLD-based network, the edge-based execution time of 8,000 frames is about 319.77 ms, which almost achieves real time, as illustrated in Fig. 9(c). In comparison, the cloud-based execution time is about 1172.11 ms, nearly reaching 3.7x of that in the edge. In Fig. 9(d), 9(e), and 9(f), the cloud-based execution time of 8,000 four-face frames is 1117.84 ms, about 566.22 ms, and about 1172.11 ms under 320*240, 640*480 and 1280*720, respectively. In the same case, the execution time of the edge model is about 319.77 ms under resolution 320*240, 640*480 and 1280*720, respectively.

From the results, we can conclude that the execution time is mostly consistent in the edge while it rises considerably with the network bandwidth increment in the cloud, because each frame is transmitted to the server in the cloud, which significantly increases transmission time. However, the edge model makes full use of the computing resources in the edge node to perform face detection; and then, the face image is transmitted to a server in the cloud. The average amount of data transmission volume reduces under the same condition.

3) POWER CONSUMPTION

Fig. 10 shows the power consumption in the edge node including the mobile camera and computing unit in the edge, respectively. In Fig. 10(a), 10(b), and 10(c), the HBLD-based results show the difference between edge and cloud in three scenarios, respectively. The results based on the LBHD are presented in Fig. 10(d), 10(e), and 10(f), respectively. We can see that the power consumption in the edge is less than 2.0%, which is lower than that in the cloud. Take the resolution 640*480 in the HBLD-based network as an example, the power consumption of the edge for processing 8,000 frames with zero-, two-, and four-faces are 0.82%, 1.03%, and 1.22%, respectively. While the cloud-based power consumption is 5.92%, 4.44%, and 6.85%, which are much higher than that in the edge. In the LBHD-based network, there is a similar result.

The result of power consumption is largely determined by the transmission data and execution time mentioned above. In summary, the edge has better performance in terms of transmission data volume, execution time, and power consumption.

V. RELATED WORK

As mentioned in the work of Shi [21], the emerging edge computing is defined as 'the enabling technologies allowing computation to be performed at the edge of the network, on downstream data on behalf of cloud services and upstream data on behalf of Internet of Everything service'. Edge computing model is promising for the killer applications with latency-sensitive requirements [22]. Edge computing-based video processing system can perform video data analytics at the edge node that is close to the cameras [23], [24]. Edge Video analytics can reduce video data transmission [25] on the network and improve the latency of video processing [26].

There are mostly two types of cameras, *i.e.,* stationary cameras and mobile cameras. The former is employed in building and aside road in video surveillance systems [27]. Mobile cameras can capture video stream at different locations [28].

For example, a project using Google Glass for paramedics was launched in 2014, but Google Glass showed a lack of connection stability and very short battery life [29]. Wu et al. [30] studied the application of wearable sensing, smart mobile devices, and video technology. Then, they also proposed an efficient, intelligent real-time emergency system for a pre-hospital emergency medical services (EMS) to improve the quality of EMS using live video streaming. Additionally, some glass-enabled applications have been developed [28], [31], [32], and the wearable mobile camera allows users to keep working while performing a remote video.

Glasses-based cameras that can be used as an edge device sends/receives video data through the WiFi network created by the smartphone. The smart glasses are also connected to the monitoring system and can display video. But the video data cannot be used to perform video anlaytics in the mobile smartphone or unit [33]. Then, the quality of the video is very important for mobile applications to acquire accurate on-site information.

Several deep learning frameworks, such as mxnet [34] and caffe [35] are suitable for cloud computing-based face detection and recognition, which costs a huge amount of memory and computing resources. Due to the limited resources in the mobile devices, an alternative approach is to upload the video data that is acquired from the mobile camera to the cloud node for detection and identification.

Currently, researcher [36] are motivated by edge computing to study the techniques that can conduct video analytics on the mobile edge nodes. These approaches aims to design lightweight target detection algorithms for mobile devices, which aims to improve the quality of the mobile video stream, especially in bad communication condition. In this work, we proposed an edge-cloud collaborative computing-based platform called mVideo for the mobile video processing. mVideo timely partitions and offloads different DNN-based video analytics tasks. Extensive experimental results validate that our proposed platform can provide accuracy and real-time data analytics in mobile video-based applications.

VI. DISCUSSION AND CONCLUSION

We design a new mobile video stream processing platform *i.e.,* mVideo, based on edge computing and exemplify with face detection and recognition. A two-level system (*i.e.,* the edge node and the cloud node) is illustrated in our work. During the mobile video processing, the face detection module is conducted in the edge, which aims to pre-process the image compacted from front-end cameras. Then, results are sent to the cloud node to perform face recognition, which can provide the *who is* to users. The cloud module is a *profiler* to collect the status of nodes in the video processing system and *offloader* to partition and offload different components of face detection and recognition to edge node and cloud node.

The mVideo is featured with a good scalability when several edge nodes are deployed in the system. In future work, we will extend the mVideo to a distributed system by adding several edge devices between edge nodes and cloud services. Second, we plane to combine multiple detection and recognition fusion methods for the mobile video processing system in the city scale. Third, we will propose a model to evaluate the edge computing framework for the collaborative tasks processing.

REFERENCES

[1] T. D. Räty, "Survey on contemporary remote surveillance systems for public safety," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 40, no. 5, pp. 493–515, Sep. 2010.

[2] P. L. Venetianer, A. J. Lipton, A. J. Chosak, M. F. Frazier, N. Haering, G. W. Myers, W. Yin, and Z. Zhang, "Video surveillance system employing video primitives," U.S. Patent 7 868 912, Jan. 11, 2011.

[3] T. Zhang, A. Chowdhery, P. V. Bahl, K. Jamieson, and S. Banerjee, "The design and implementation of a wireless video surveillance system," in *Proc. 21st Annu. Int. Conf. Mobile Comput. Netw.*, 2015, pp. 426–438.

[4] D. N. Parmar and B. B. Mehta, "Face recognition methods & applications," 2014, *arXiv:1403.0485*. [Online]. Available: https://arxiv.org/abs/1403.0485

[5] J. D. Woodward Jr, C. Horn, J. Gatune, and A. Thomas, "Biometrics: A look at facial recognition," Rand Corp., Santa Monica, CA, USA, Tech. Rep., 2003.

[6] M. Haghighat, S. Zonouz, and M. Abdel-Mottaleb, "CloudID: Trustworthy cloud-based and cross-enterprise biometric identification," *Expert Syst. Appl.*, vol. 42, no. 21, pp. 7905–7916, Nov. 2015.

[7] W. Wang, A. X. Liu, and M. Shahzad, "Gait recognition using WiFi signals," in *Proc. ACM Int. Joint Conf. Pervas. Ubiquitous Comput.*, 2016, pp. 363–373.

[8] United States Government, Department of Justice. *Amber Alert*. Accessed: Apr. 10, 2019. [Online]. Available: http://www.amberalert.gov/faqs.htm

[9] A. H. M. Amin, N. M. Ahmad, and A. M. M. Ali, "Decentralized face recognition scheme for distributed video surveillance in IoT-cloud infrastructure," in *Proc. Region Symp.*, 2016.

[10] S. Hanckmann, Patrick. (Jan. 2017). *Video Access System and Method Based on Action Type Detection*. [Online]. Available: http://www.freepatentsonline.com/9554081.html

[11] B. Nemade, "Automatic traffic surveillance using video tracking," *Procedia Comput. Sci.*, vol. 79, pp. 402–409, 2016.

[12] A. Ben Mabrouk and E. Zagrouba, "Abnormal behavior recognition for intelligent video surveillance systems," *Expert Syst. Appl.*, vol. 91, pp. 480–491, Jan. 2018, doi: 10.1016/j.eswa.2017.09.029.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016.

[14] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017.

[15] A. Vinay, V. S. Shekhar, J. Rituparna, T. Aggrawal, K. B. Murthy, and S. Natarajan, "Cloud based big data analytics framework for face recognition in social networks using machine learning," *Procedia Comput. Sci.*, vol. 50, no. 1, pp. 623–630, 2015.

[16] N. Stekas and D. van den Heuvel, "Face recognition using local binary patterns histograms (LBPH) on an FPGA-based system on chip (SOC)," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops (IPDPSW)*, May 2016, pp. 300–304.

[17] C. S. S. Prasanna, N. Sudha, and V. Kamakoti, "A principal component neural network-based face recognition system and asic implementation," in *Proc. 18th Int. Conf. VLSI Design Held Jointly, 4th Int. Conf. Embedded Syst. Design*, Apr. 2005, pp. 795–798.

[18] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Process. Lett.*, vol. 23, no. 10, pp. 1499–1503, Oct. 2016.

[19] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 815–823.

[20] P. Hintjens, *ZeroMQ: Messaging for Many Applications*. Newton, MA, USA: O'Reilly Media, 2013.

[21] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[22] 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," *Computer*, vol. 50, no. 10, pp. 58–67, 2017.

[23] T. X. Tran, P. Pandey, A. Hajisami, and D. Pompili, "Collaborative multi-bitrate video caching and processing in mobile-edge computing networks," in *Proc. 13th Annu. Conf. Wireless Demand Netw. Syst. Services (WONS)*, Feb. 2017, pp. 165–172.

[24] Q. Zhang, Q. Zhang, W. Shi, and H. Zhong, "Firework: Data processing and sharing for hybrid cloud-edge analytics," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 9, pp. 2004–2017, Sep. 2018.

[25] P. Hu, H. Ning, T. Qiu, Y. Zhang, and X. Luo, "Fog computing based face identification and resolution scheme in Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 13, no. 4, pp. 1910–1920, Aug. 2017.

[26] M. Quinn. (2017). *ETHAN Telemedicine Program Has Prevented 6,000 Unnecessary ED Transports in Houston*. Accessed: Jan. 12, 2019. [Online]. Available: https://www.emsworld.com/news/12334469/ethan-telemedicine-program-has-prevented-6-000-unnecessary-ed-transports-in-houston

[27] H. Sun, X. Liang, and W. Shi, "Vu: Video usefulness and its application in large-scale video surveillance systems: An early experience," in *Proc. Workshop Smart Internet Things (SmartIoT)*, New York, NY, USA, 2017, pp. 6:1–6:6, doi: 10.1145/3132479.3132485.

[28] Schaer, Melly, Muller, and Widmer, "Using smart glasses in medical emergency situations, a qualitative pilot study," in *Proc. IEEE Wireless Health (WH)*, Oct. 2016, pp. 54–58.

[29] A. Widmer and H. Müller, "Using Google glass to enhance pre-hospital care," *Swiss Med. Informat.*, vol. 30, pp. 1–4, Sep. 2014.

[30] X. Wu, R. Dunne, Z. Yu, and W. Shi, "STREMS: A smart real-time solution toward enhancing EMS prehospital quality," in *Proc. IEEE/ACM Int. Conf. Connected Health, Appl., Syst. Eng. Technol. (CHASE)*, Jul. 2017, pp. 365–372.

[31] A. Widmer, R. Schaer, D. Markonis, and H. Müller, "Facilitating medical information search using google glass connected to a content-based medical image retrieval system," in *Proc. 36th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Aug. 2014, pp. 4507–4510.

[32] U.-V. Albrecht, U. Von Jan, J. Kuebler, C. Zoeller, M. Lacher, O. J. Muensterer, M. Ettinger, M. Klintschar, and L. Hagemeier, "Google glass for documentation of medical findings: Evaluation in forensic medicine," *J. Med. Internet Res.*, vol. 16, no. 2, p. e53, Feb. 2014.

[33] T. Elgamal, B. Chen, and K. Nahrstedt, "Teleconsultant: Communication and analysis of wearable videos in emergency medical environments," in *Proc. ACM Multimedia Conf.*, 2017, pp. 1241–1242.

[34] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems," 2015, *arXiv:1512.01274*. [Online]. Available: https://arxiv.org/abs/1512.01274

[35] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22nd ACM Int. Conf. Multimedia*, 2014, pp. 675–678.

[36] P. Liu, B. Qi, and S. Banerjee, "Edgeeye: An edge service framework for real-time intelligent video analytics," in *Proc. 1st Int. Workshop Edge Syst., Anal. Netw. (EdgeSys)*, New York, NY, USA, 2018, pp. 1–6, doi: 10.1145/3213344.3213345.

HUI SUN received the Ph.D. degree from Huazhong University Science and Technology, in 2012. He is currently an Assistant Professor of computer science with Anhui University. His research interests include computer systems, edge computing, performance evaluation, non-volatile memory-based storage systems, file systems, and I/O architectures.

YING YU was born in 1996. She is currently pursuing the M.S. degree with Anhui University. Her main research interests include computer systems, edge computing, and storage systems.

KEWEI SHA (Senior Member, IEEE) received the Ph.D. degree in computer science from Wayne State University, in 2008. He is currently an Associate Professor of computer science and the Associate Director of the Cyber Security Institute, University of Houston-Clear Lake (UHCL). His research interests include the Internet of Things, cyber-physical systems, edge computing, security and privacy, and data management and analytics.

BENDONG LOU was born in 1995. He is currently pursuing the M.S. degree with Anhui University. His main research interests include computer systems, edge computing, and storage systems.