Future Generation Computer Systems ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Cloud-based scalable object detection and classification in video streams

Muhammad Usman Yaseen [a,*], Ashiq Anjum [a], Omer Rana [b], Richard Hill [a]

[a] College of Engineering and Technology, University of Derby, UK
[b] School of Computer Science and Informatics, Cardiff University, UK

HIGHLIGHTS

- A cloud-based, automated video analysis system is presented.
- The system reduces the involvement of human factor during video analysis.
- The objects can be classified without requiring any metric learning stage.
- The system locates an object from one month of recorded video streams in 6.52 h.
- The GPU powered infrastructure took 3 h to accomplish the same task.

ARTICLE INFO

Article history:
Received 2 August 2016
Received in revised form 16 January 2017
Accepted 4 February 2017
Available online xxxx

Keywords:
Unsupervised object classification
Cloud computing
GPUs
High performance video analytics

ABSTRACT

Due to the recent advances in cameras, cell phones and camcorders, particularly the resolution at which they can record an image/video, large amounts of data are generated daily. This video data is often so large that manually inspecting it for useful content can be time consuming and error prone, thereby it requires automated analysis to extract useful information and metadata. Existing video analysis systems lack automation, scalability and operate under a supervised learning domain, requiring substantial amounts of labelled data and training time. We present a cloud-based, automated video analysis system to process large numbers of video streams, where the underlying infrastructure is able to scale based on the number and size of the stream(s) being considered. The system automates the video analysis process and reduces manual intervention. An operator using this system only specifies which object of interest is to be located from the video streams. Video streams are then automatically fetched from the cloud storage and analysed in an unsupervised way. The proposed system was able to locate and classify an object of interest from one month of recorded video streams comprising 175 GB in size on a 15 node cloud in 6.52 h. The GPU powered infrastructure took 3 h to accomplish the same task. Occupancy of GPU resources in cloud is optimized and data transfer between CPU and GPU is minimized to achieve high performance. The scalability of the system is demonstrated along with a classification accuracy of 95%.

1. Introduction

The increasing availability and deployment of video cameras has resulted in the generation of thousands of high resolution videos streams. Such videos can be sub-divided into a number of frames of interest. Various types of information can be extracted from these video frames, such as classification of moving objects corresponding to a specific area of interest. The term video analytics refers to the optimized processing of these video frames by using intelligent approaches such as a machine learning, so that clusters of information can be automatically extracted from them.

Video analytics systems mainly perform object detection and recognition. Object detection refers to the detection of all instances of an object belonging to a known category, such as faces or cars, within a sequence of frames. Often a video may contain a number of objects. These objects can reside at any location within a frame, requiring the detection process to investigate different parts of a frame to locate the object of interest. Object recognition, on the other hand, refers to the identification of detected objects. A video stream and some known labels are provided to the system. It then assigns the correct labels to the detected objects in a video stream. [1–3] describe how video frame analysis can be used to support detection, tracking and recognition of objects. However, these systems are expensive in terms of processing time and cost [4], and require human monitoring and intervention [5] and

* Corresponding author.
E-mail addresses: m.yaseen@derby.ac.uk (M.U. Yaseen), a.anjum@derby.ac.uk (A. Anjum), ranaof@cardiff.ac.uk (O. Rana), r.hill@derby.ac.uk (R. Hill).

http://dx.doi.org/10.1016/j.future.2017.02.003
0167-739X/

address challenges that are often relevant for still images [6]. These systems are also resource intensive. Due to cognitive limitations, an operator cannot focus on recorded video streams for more than 20 min, making it challenging to perform efficient and robust large scale video analysis. Scaling such analysis to large data volumes remains a challenge. Additionally, to gain greater insights into the analysed video content, computationally intensive algorithms (e.g. deep learning algorithms [7]) with large storage requirements are needed. This work utilizes the advantages of machine learning based classification approaches to develop an automated video analysis system which overcomes these challenges. The focus of this work is to build a cloud-based robust and scalable solution for the processing of large number of video streams. We employed the detection and classification algorithms in combination to combine the benefits of both supervised and unsupervised learning domains. The Haar Cascade Classifier [8] has been demonstrated to be highly accurate for object detection, especially for detecting faces in still images [9]. We have therefore investigated its use for video sequences. Similarly, the Local Binary Pattern Histogram [10] classification algorithm is widely used, primarily because of its computational simplicity and high accuracy. Our system requires minimum human interaction for identifying objects in a large number of video frames. The system is based on a very simple object matching concept based on local binary patterns. After the extraction of desired objects, we employ an object matching algorithm to perform object recognition. This enabled us to perform classification without any metric learning algorithm and labelled training data.

An operator using the system only specifies which object of interest is to be located. The video streams are then automatically fetched from cloud storage and processed frame by frame. The object is first detected in a frame to provide a reference for the location of the object which can be tracked in the subsequent frames. It is cropped and saved as a separate image, so that the recognition step will have to process a smaller sized image. The moving object is then passed on to the subsequent object recognition phase for identification.

The recognition phase first analyses the marked input object. It extracts and stores features from it. This marked object is then compared with all of the other frames. If the same object is identified in any other frame its instance is updated and its corresponding time and location is saved. If the comparison fails then it means that the marked object is not present in the video stream which is currently being processed. This marked object is then fed to the next video stream and the same process is repeated. Depending upon the features being considered, a decision is made whether the object is present in the analysed video stream. If the object is located in the video stream, its time and location is saved and updated. This mechanism is performed for all the video streams and cumulative time and locations are stored in a database. Statistical similarity measures are used to compare extracted frames. To support scalability and throughput, the system is deployed on compute nodes that have a combination of CPU and GPU, within a cloud system. This also enables on-the-fly and on-demand analysis of video streams.

The main contributions of this paper are as follows: Firstly, a robust video analysis system is proposed which employs two learning algorithms in combination, to perform quick analysis on large numbers of video streams. Secondly, we perform object classification on the extracted objects in an automated and unsupervised way. No training or manually labeled dataset is required in our approach. Thirdly, the proposed system is scalable with high throughput as it is deployed on a cloud based infrastructure that have a combination of CPU and GPU. The paper is structured as follows: Section 2 compares our work with related approaches, providing a survey of the most recently used features and classifiers for object detection and recognition. The proposed approach and its architecture are explained in Sections 3 and 4 respectively. The implementation of the proposed system is described in Section 5. Section 6 details the experimental setup and Section 7 reveals the results obtained from implementation in terms of accuracy, scalability, performance and throughput. The conclusions drawn from the work and the future directions are presented in Section 8.

2. Related work

Significant literature already exists for image and video processing. However, the effective use of these techniques for analysing a large volume of video data, the size of which may not be known 'a priori', is limited. Additionally, carrying out such analysis on scalable/elastic infrastructures also remains limited at present.

Object Classification Approaches: Object classification has been an area of great interest for the past decade. Yuanqing et al. [11] proposed an automated fast feature extraction approach for large scale image classification using Support Vector Machines. Similarly Nikam et al. [12] developed a scalable and parallel rule based system to classify large image datasets and concluded that the system is reliable, with computation time decreasing as the number of nodes increase.

Giang et al. [13] used CNN to differentiate between pedestrians and non-pedestrians in an urban environment. They scan input images at different scales, and at each scale all windows of fixed size are processed by a CNN classifier to determine whether an input window is pedestrian or not. Feature extraction and classification phases were integrated in one single fully adaptive structure. All three layers of CNN i.e. convolution layers, sub-sampling layers, and output layers were used to perform classification. This work showed that it is possible to lower training time while maintaining a threshold classification rate.

In another study, Masayuki et al. [14] implemented a parallel cascade of classifiers consisting of a large number of stages. The first stage contains a subset of features selected from training data that efficiently distinguish two classes. The cascade is then applied on the training data where more false positives are observed. A new training set is formed by combining the misclassification which are then used for second stage of cascade. This procedure continues until an acceptable performance in a training sequence is achieved. According to the authors, the later stages are not executed too often, so only early stages were executed in parallel, leading to a reduced total processing time. To make such a cascade of classifiers more effective, Xusheng et al. [15] exploited the use of genetic algorithms as a post optimization procedure for each stage classifier and achieved a speedup of 22%.

Xing et al. [16] used multiple independent features to train a set of classifiers online, which *collaborate* with each other to classify the unlabelled data. This newly labelled data is then used to update classifiers using co-training. The independent features which were used are Histogram of oriented Gradients(HoG) and colour histograms. A Support Vector Machine (SVM) was trained by each feature and final classification results were produced by combining the outputs of all SVMs.

Principle components of a face image generated from principle component analysis (PCA) [17] are known as eigen-faces and are used in various works [18,19] for classification. Facial recognition performed by PCA is insensitive to facial expressions. However, performance degrades in extreme lighting conditions. Linear discriminant analysis (LDA) [20] was used to generate Fisher Faces and proved to outperform PCA in face recognition tasks under complex conditions. LDA provided a way to overcome the shortcomings of the PCA approach but it can face the small sample size problem. Independent component analysis (ICA) [21] is a generalization of principle component analysis and was used in various works for

Please cite this article in press as: M.U. Yaseen, et al., Cloud-based scalable object detection and classification in video streams, Future Generation Computer Systems (2017), http://dx.doi.org/10.1016/j.future.2017.02.003.

facial recognition. The objective of ICA is same as PCA but it generates spatially localized features. In contrast to PCA, no information in images is destroyed by using this technique. But one also has to compromise on redundant information present in the images which makes this technique computationally expensive.

The analysis of video streams has been the focus of many commercial vendors recently. An intelligent system Vi-System [22] was developed for the surveillance and monitoring of objects in crowds. It was based on analytical rules and was capable of generating alerts on defined parameters. SmartCCTV [23] on the other hand is mainly used in UK transportation systems and provides optical based survey solutions and video incident detection systems. Project BESAFE [24] and Intelligent Vision [25] are tools to perform intelligent video analysis for fully automated video monitoring of premises. Their services include tracking of abnormal behaviour of people and detection of their activities. One of the embedded video analytic systems is IVA 5.6 [26] which facilitates the detection and tracking of moving objects. It has the capability to detect inactive and removed objects, as well as loitering and object trajectories. However, most of these commercial systems have the limitation of scalability for a large number of streams and require high bandwidth for video stream transmission.

Object Classification in the Clouds: When object classification is needed to be performed on large scale datasets, it requires large storage and computational resources. Efficient object classification using cloud systems has also been explored in the literature, by managing distribution of video streams and load balancing among various available cloud nodes [27]. A pervasive cloud computing infrastructure was utilized in [28] to recognize food images. Cloud computing was used to process images of different kinds of foods using differing lighting conditions, in various colours and viewing angles. However, the authors concluded that it is not promising to use the cloud computing paradigm for small datasets as job preparation overheads reduce the performance of the system.

A Hadoop based object classification system was implemented in [29] by using two dimensional principle component analysis. In another study, a massively parallel cloud computing architecture was presented [30] to classify astronomical images. A large scale video processing system was demonstrated by [31,32] using MapReduce clusters. However, no enhancement in the video processing routines was presented in these studies.

Recently, the use of GPUs as a high performance resource for the processing of large scale video data has become an active research area [33], as GPUs support a multi-threaded architecture and offer abundant computational power. They have been used for various large scale video processing tasks such as object detection [34], motion estimation [35], and object recognition by using deep belief networks [36] and sparse coding [37]. It has been demonstrated in these studies that a speedup of 5 to 15 times can be achieved as compared to the use of standard CPUs [38].

The accuracy and performance of an object classification system is highly dependent on the similarity metric along with good visual representation. Most of the recent object classification approaches do provide good visual representation but also necessitate learning a dataset specific metric. It helps to learn and understand the underlying regularities of a specific dataset which in turn results in improved performance. However, this phenomenon is time consuming because a large number of training examples must be collected and labelled manually. The collection of large number of training examples and their labelling is itself a major challenge. Although these training examples enable the system to capture variations in object appearances, they also burden the training process [39,40]. Machine learning approaches such as semi supervised learning and unsupervised learning are a way to reduce the time required for the training process. They train the system with a small number of completely labelled examples and another set of unlabelled examples which reduces computation time.

The focus of this paper is to propose a cloud based video analysis system that has a combination of CPU and GPU-based compute nodes to identify objects of interest from a large number of video streams. The proposed system requires minimum human interaction and performs object classification in an unsupervised way. It is scalable and supports processing of large number of recorded video streams as compared to existing cloud based video analytics approaches.

3. Video analysis approach

We present the approach behind our video analysis system in this section. Each video stream is first decoded to extract individual video frames. The objects of interest are extracted from the video frames by detecting and cropping around the area of detection. The local patterns of each extracted object are then generated and stored in the associated buffer. Object matching is then performed on the generated local features. The generated results are then stored in the database. Algorithm 1 shows the approach used in our object classification system.

Algorithm 1 Object Classification

```
for all streams in the database do
    for all decoded frames from stream do
        Launch object detection module
        Extract (crop) desired object from frame
        Generate local patterns for the extracted object
        Store generated patterns in an associated buffer
    end for
        for all object recognition patterns in
            the database do
            Launch object matching module
            Compare stored patterns with marked objects
            Generate matching scores for each object
            Store results in the database
        end for
end for
```

The system applies multiple machine learning algorithms for detection and recognition. The algorithms are employed in such a way that the results produced by one algorithm are processed further by the following algorithm. The first algorithm is used to extract the object of interest from the whole frame in such a way that it narrows down the image area. The rest of the frame which contains unwanted information is discarded to save processing time and resources. This algorithm independently operates on all the frames in a sequence. This results in the extraction of all the desired objects from all the video frames. Fig. 1 presents the process followed in our approach.

3.1. Object detection and classification

We have used the Haar Cascade Frontal Face Classifier algorithm for the extraction of human faces. The extraction of desired faces from the frames helps to improve the performance of the system in two ways: (i) Since the frame area is reduced so the analysis algorithm now has to process a smaller sized frame as compared to original one. This reduces the processing time of individual frames and in turn reduces the overall processing time of the whole video. (ii) As the frame has been narrowed down to only object(s) of interest, by removing the unwanted area of the frame, it now contains only the desired object. The illumination effects and noise which have the possibility to be present in the unwanted area will not reflect in the object recognition process. This will lead to improvements in the accuracy of overall system.

The extracted objects are then processed via the object recognition phase, which generates local binary patterns of all the extracted objects. These local binary patterns serve as features which

Please cite this article in press as: M.U. Yaseen, et al., Cloud-based scalable object detection and classification in video streams, Future Generation Computer Systems (2017), http://dx.doi.org/10.1016/j.future.2017.02.003.

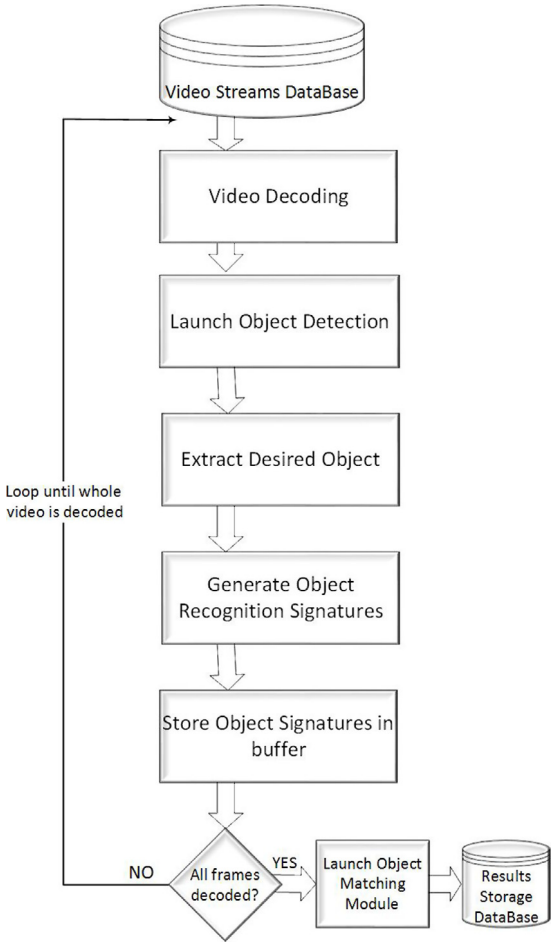

Fig. 1. Video processing workflow.

can be used for the recognition of a known object. These features represent the extracted objects in such a way that they become highly discriminative to various grey-level changes in the objects.

We have used the extended version of the local binary pattern operator which makes the use of uniform patterns. The use of uniform patterns helps to decrease the size of feature vector. Since we are calculating the local binary patterns of a huge dataset, the use of uniform patterns helps to lower the computation cost. Uniform patterns work on the phenomenon that some of the patterns occur more frequently than other patterns. A pattern is said to be uniform if there are a maximum of two bit-wise transitions from 1 to 0 or vice versa. The patterns 01110000 and 11001111 have two transitions and are thus uniform. These uniform patterns are used during the computation of LBP labels with a separate label for each uniform pattern. The rest of the non-uniform patterns are labelled with a single label.

The computation of the local pattern features is a compute intensive procedure as it involves the manipulation of every pixel in the video frame. The porting of this compute intensive procedure to GPUs is performed to reduce the computation demands. A GPU kernel is developed for this purpose. It performs the procedure of local pattern feature generation in parallel instead of sequential processing as in a CPU. Fig. 2 shows the relation between Haar Cascade Classifier and the LBPH algorithm.

The extracted human faces from the video streams are represented in the form of histograms by their binary pattern representation. The comparison of the marked face has then been made with the faces extracted from the video streams by simply computing the similarity measure between them. The proposed system does not require learning a data specific metric in order to compare faces. The representation of the faces is capable enough to distinguish the underlying irregularities of the dataset. The proposed face matching algorithm (Algorithm 2) has proved to be generic and is not adapted to any dataset. It is capable of identifying faces from the video streams without requiring any complex similarity metric-learning algorithm, pre-labelled dataset, any other supervised learning model or any outside data from other sources.

An object matching algorithm is applied on the local patterns of detected objects for recognition. The recognition process is performed by comparing the detected object features with the stored object information. The comparison is made on the basis of histogram intersection which is used as a distance measure. The histogram intersection can be calculated as [41]:

$$D(S, M) = \sum_{b-1}^{B} min(S_b, M_b) \quad (1)$$

where 'S' and 'M' are a pair of histograms of two video frames containing 'B' bins.

Each comparison generates a score of each individual registered in a database. These scores obtained after performing the histogram intersection determine the recognition of a marked person which was being searched in the video streams. We have used a threshold of 90% match in our experiments. We obtained over 90% accuracy rate in case of matching individual objects. The matching scores for unmatched individuals is 70% or below. The matching scores along with locations and time of presence are stored in the database. This module is totally unsupervised and is independent of any metric learning stage. The recognition is performed only on the basis of similarity measure between the features of two objects.

The performance of any object classification system can be affected by the facial structure constraints (gender, ethnicity) and the viewing parameters such as illumination and viewpoint. In addition, a number of perceptual complications can occur due to the movement of objects in video streams. The facial movements of a person can be classified as rigid or non-rigid. The rigid movements include tilting, nodding or shaking around the vertical axis. These movements can change the angle of a face from a static point. On the other hand, non-rigid movements take place due to facial expressions and eye-gaze during speech. These movements can distort the identifying features of the face. A smiling facial expression can strongly differ from a surprised facial expression. This difference occurs due to the relative change in position of the eyebrows with respect to nose, mouth or other features.

It was observed during the experiments that because of the discriminative power of the LBPH operator, it is capable to perform well at low level of perceptual complications. The LBPH operator has shown its performance for various rigid and non-rigid movements by providing high accuracy rates. Also, since the dataset is generated under controlled conditions, it does not pose significant changes to illumination or viewpoint.

Algorithm 2 Object Matching

```
1: procedure ObjectMatching
2:     Compute LBP Histogram of Marked Object
3:     Compute LBP Histograms of Objects in Video Streams
4:     for all Objects in Video Streams do
5:         Compute HistogramIntersection of MarkedObject with
   Objects in Streams
6:         if IntersectionResult > 0.9 then
7:             ObjectFound
8:         else
9:             ObjectNotFound
10:    end for
```

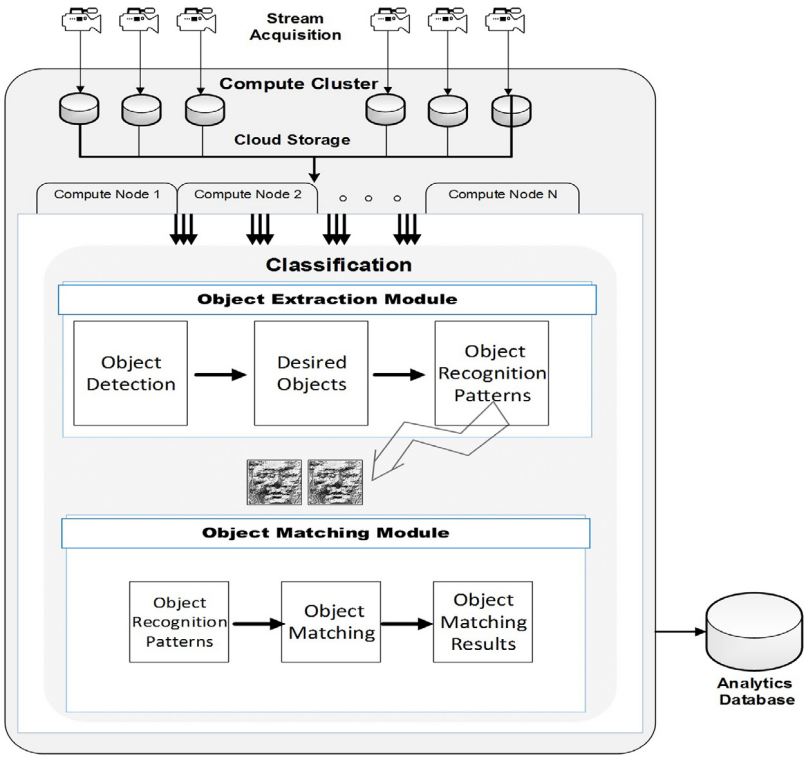

Fig. 2. System architecture.

4. System architecture

The overall architecture of the system is illustrated in Fig. 2. The proposed system provides scalable and automated classification of objects in a large number of video streams in an unsupervised way. It is independent of the need of labelled training data and metric learning stage. The use of GPU-enabled cloud nodes enables the system to achieve high throughput. Scalability challenge is also addressed by leveraging the benefits of GPU mounted servers in the cloud. The transfer time overhead of moving the video data from the camera to cloud storage is not considered in this work. This overhead is dependent on the speed of the network connecting the camera/data capture source to the cloud system.

The video streams are first fetched from cloud storage and are decoded to extract individual video frames. The decoded individual video frames are stored in the input frame buffer. This buffer is a temporary storage in main memory for decoded video frames. The recorded video streams are encoded with the H.264 encoder to save storage space. Each video stream is recorded at 25 frames per second, with 3000 (120*25) video frames for a video stream of 120 s length. The number of decoded video frames is dependent upon the length of video stream being analysed.

Each frame is then processed individually for object detection and recognition. The objects of interest are first detected using the Haar Cascade Classifier algorithm. This detection helps to extract only the desired objects from the overall frame. The extracted objects are stored in a memory buffer for further processing.

We have used the already trained frontal face classifier for detection of human faces from video streams. Training is not performed separately and saves the computation cost of training. The computation cost of the detector is highly dependent on the number of features being evaluated. The small number of features means low computational cost but the classifier will also be less accurate. A classifier with more features results in higher classifier accuracy. It was noted during the experiments that a frontal face classifier built on 25 feature stages provides a detection rate of 95%. The computation time depends on the resolution of the video frame. So there is a trade-off between the computation cost and accuracy of the classifier. Fig. 3 shows the elapsed time for various stages of the object detector within a CPU node. We have also analysed the execution time of a video stream within a CPU node for various processing stages including local binary pattern histogram. Fig. 3 also depicts the execution time of these stages.

The next module after the extraction of desired objects is a feature generation module. This module generates the local patterns against each detected object. These local patterns serve as features which are further used to recognize the marked object. This module mainly consists of the execution of local binary pattern histogram. A histogram of each of the detected objects is created and stored in the buffer as an output of this module. We have used a profiling mechanism to identify the compute intensive steps of our system. The generation of local pattern features is a compute intensive process. This compute intensive feature generation process has been ported to GPUs, through the design/ implementation of a kernel which performs generation of local patterns on GPUs. Each pixel of the video frame is mapped to a thread. This thread is then responsible for launching kernel for each pixel and processing it in parallel. The size of the thread block is dependent on the size of the frame. These threads work in a synchronous way to process frame data in parallel. A high level of parallelism is achieved since each pixel in the video frame is processed in parallel. Once the processing of frame buffer is completed, the resulting processed frame is stored in an output buffer.

5. System implementation

This section provides a description of the system components, their functionality and implementation. The operations employed to process video streams to support object detection and recognition are also described.

Please cite this article in press as: M.U. Yaseen, et al., Cloud-based scalable object detection and classification in video streams, Future Generation Computer Systems (2017), http://dx.doi.org/10.1016/j.future.2017.02.003.

Function Name	Elapsed Inclusive Time %	Elapsed Exclusive Time %
main	82.29	0.00
ObjectDetection::detectObjects	71.43	0.00
ObjectDetection::ScaleImage_Invoker	70.89	11.36
ObjectDetection::runCascadeClassifier	59.53	32.98
evalWeakClassifier	18.26	18.26

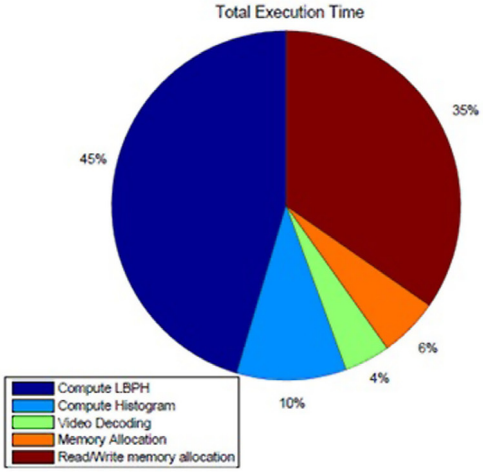

Fig. 3. Execution time of various modules.

Fig. 4. Extracted faces from video streams.

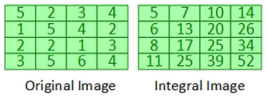

Fig. 5. Original and integral image.

5.1. Video decoding

The video streams are decoded to extract individual video frames. These frames are then transferred to the processing module to enable the detection and recognition process to be carried out. Hence, each frame can be processed independently of each other. This approach enables the processing of individual frames on cloud resources, leading to high throughput and scalability.

5.2. Object extraction

After the frame is decoded from the video stream, the next step is to extract faces from frames using an object detection algorithm. We have used Haar Cascade Classifier for this purpose. The input image is cropped automatically around the output of Haar Cascade algorithm for the next step i.e. face recognition. This helps to narrow down the area of image to a small rectangle containing the desired face. Fig. 4 shows some of the extracted faces from the video streams. We also monitor the persistence of an object across multiple frames of a video stream. In this way, although each frame is individually processed, tracking an object across multiple frames enables us to monitor its presence over a particular time period.

The Haar Cascade Classifier is constructed on top of Haar features which are extracted from objects present in video frames. In order to make the classifier scale-invariant, a frame pyramid approach [42] has been used. The pyramid represents the same frame in multiple scales and enables the detector to be scale invariant. Objects with varying image sizes can easily be detected through the pyramid approach. An object pyramid can be constructed by using a down-sampling approach which samples the frame by one scale in each iteration. An integral image for each scale in the pyramid is then calculated to speed up the process of generating a pixels sum. Integral image [43] helps to compute the summation of pixels present in a rectangular region by utilizing only four pixel corners. This approach of using integral images is highly efficient, especially for the cases in which the pixel sum of many rectangular regions of the same image need to be computed. Since the detector uses the sliding window approach and a pixel sum for each shifted window is required, this approach reduces the complexity of the overall process. Fig. 5 shows a representation of the integral image.

The sliding window is used, pixel by pixel, on the whole frame in search of an object (e.g. a face). The area under the sliding window is passed to the cascaded classifier. As most of the image area is a non-face region it groups the features into different stages based on the classifiers used. The region that passes all stages of the cascaded classifier is a face. The area under the sliding window is required to be passed through all stages of the cascade classifier. If at any stage, the input region is unable to pass the stage by not meeting the required threshold, it is immediately rejected. If the region passes all the stages successfully, then it is considered to be the face. On detection, an object recognition algorithm is invoked.

5.3. Local feature generation

Each detected object of interest is then analysed, by using LBPH. The algorithm computes local binary patterns in order to generate feature vectors. In order to compute LBP features, the examined

Please cite this article in press as: M.U. Yaseen, et al., Cloud-based scalable object detection and classification in video streams, Future Generation Computer Systems (2017), http://dx.doi.org/10.1016/j.future.2017.02.003.

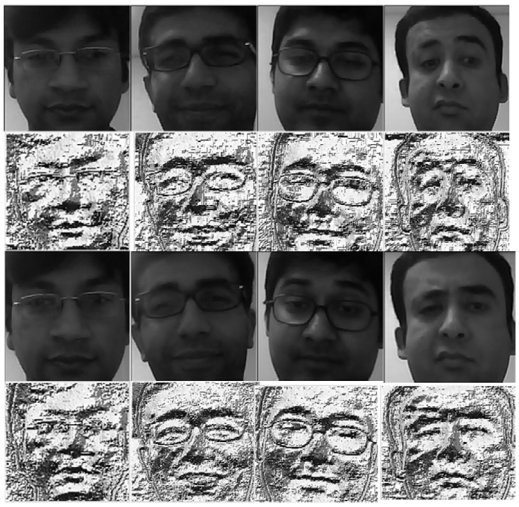

Fig. 6. Original and LBP faces.

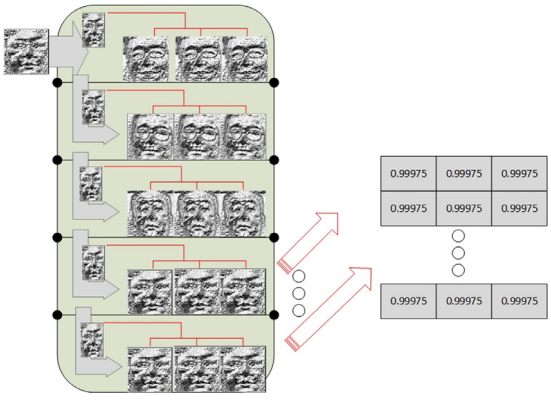

Fig. 7. Visualization of matching process.

window is divided into multiple cells. Each cell contains a sub-block of 3 ×3 pixels. Then each pixel in the sub-block is compared to its neighbouring pixels. If the value of centre pixel is greater than its neighbour pixel, 1 is stored at the location of that pixel. If the values of centre pixel is less than the neighbouring pixel, the grey value of that pixel is replaced with 0. This makes the sub-block a binary block containing 0 and 1 depending upon its pixel values. This is known as the labelling of pixels. These labelled pixels generate a binary pattern which is then converted into one decimal value. The grey value of centre pixel is then replaced with the decimal value. This procedure is repeated on the whole image and an LBP image is obtained. A histogram is then calculated over the frequency of each number occurrence. This histogram gives a feature vector of the window.

In order to perform face recognition, the face image is divided into multiple blocks or regions. Then for each block or region, an LBP histogram is computed as explained above. The feature vector of the whole image is a combination of all LBP histograms of all regions in an image. Fig. 6 shows the original faces and the LBP computed faces from video streams.

5.4. Similarity measure

This procedure of LBP histogram generation is performed for all the video frames and the image which is to be matched. Matching is performed by comparing the LBP histogram of the marked object frame with all the frames of a video stream. The histogram intersection is used as a distance measure to calculate the similarity between two frames. After a person's face is authenticated correctly, the matching score associated to it is stored in a database. This phenomenon can be visualized in Fig. 7.

5.5. Local pattern feature generation on gpus

Generation of local patterns from a video frame is a compute intensive procedure. It is therefore ported to GPUs to reduce the computation demands. A GPU kernel is designed and implemented to perform this procedure. The processing of pixels is sequential in CPU based implementation. The processing time even increases exponentially as the number of video frames increases.

Conversely, the GPU implementation works in a parallel fashion. GPU implementation is known as GPU kernel and is executed by a number of threads generated by a GPU. The number of threads that a GPU can generate depends upon the processing cores of a GPU, memory and registers. It is also dependent upon the size of thread block and grid. Since each pixel is mapped to an individual thread, the number of generated threads should be equal to the number of pixels in a video frame. The availability of frame data in GPU memory enables the parallel processing of each pixel. Upon completion of the frame data processing, the processed frame data is copied back to a CPU memory buffer (host) from GPU memory buffer (host).

We have used Compute Unified Device Architecture (CUDA) to implement and generate local pattern features on GPUs. It uses SIMD (Single Instruction Multiple Data) parallel programming model and provides a collection of APIs to execute instructions on a GPU. A CUDA program initiates on a CPU and processes data on a GPU through CUDA kernels. The GPU memory is first allocated, so that frame data can be transferred from CPU to GPU. The size of the GPU memory is allocated according to the size of video frame. Three different data transfer mechanisms including page-able memory, pinned memory and zero copy have been implemented and tested in this work. Upon successful completion of frame processing, the results are transferred back to the CPU memory.

The GPU kernel is executed by a number of threads. There can be a maximum of 32 threads in a warp and each thread block has numerous warps. Thread blocks are further grouped into grid. It is the responsibility of the CUDA Work Distributor (CWD) to allocate thread blocks on a GPU. At the first step of kernel execution, these thread blocks are allocated. Kernel execution is performed in parallel with the help of CUDA streams.

The proposed system works partially on CPUs and partially on GPUs. The decoding of frames from video streams and extraction of faces is performed on a CPU. The compute intensive process of generation of local features is performed on a GPU using the CUDA kernel. The processed results are then transferred back to CPU. The results section provides a more detailed analysis of the accuracy of recognized objects and the processing time of the system.

6. Experimental setup

This section provides the details of our experimental setup used to evaluate the proposed system. The parameters used to evaluate the performance of the system are the accuracy of the algorithms, processing speed-up achieved, resource consumption, scalability, and processing time of each video frame. The purpose of cloud based deployment is to evaluate the scalability of the system. The cloud deployment with GPUs evaluates the performance, throughput, resource consumption and processing time of video streams.

The configuration of the cloud resources is as follows: the cloud instance has Ubuntu LTS 14.04.1 and is running OpenStack

Please cite this article in press as: M.U. Yaseen, et al., Cloud-based scalable object detection and classification in video streams, Future Generation Computer Systems (2017), http://dx.doi.org/10.1016/j.future.2017.02.003.

Icehouse. There are six server machines and each server machine is equipped with 12 cores. Each server is running with 6-core Intel Xeon Processors at 2.4 GHz. It has a storage capacity of 2 TB with 32 GB RAM. The cloud instance is configured with 192 GB RAM, storage capacity of 12 TB and 72 processing cores. OpenStack provides a dashboard to manage and control the resources such as storage, network and pool of computers.

A cluster consisting of 15 nodes is configured to evaluate the proposed system. The configuration of each node is as follows: 4 VCPU running at 2.4 GHz with 8 GB RAM. Each node is configured with a storage capacity of 100 GB. The evaluation parameters to measure the performance of the system include total analysis time of the system, impact of task parallelism on each node and the variations of compute nodes in the cloud. This experimental setup helps to measure the performance of the system for scalability and robustness with varying cloud configurations.

The Hadoop MapReduce framework is utilized to evaluate the system in cloud resources. Hadoop comes with Yarn which is responsible for managing resources and scheduling jobs for the running processes. It further facilitates with a NameNode in charge for the management of nodes, a Data/ComputeNode to process and store the data, and a JobTracker for the tracking of running jobs. These components of Hadoop MapReduce framework help to schedule and analyse tasks on the available nodes in parallel.

The accuracy and performance of the proposed system is evaluated on cloud nodes with 2 GPUs. The nodes are equipped with Intel Core i7 3.60 GHz processors with 16 GB RAM. Each node is supported with an ASUS GeForce GTX 780 GPU. This Kepler architecture based GPU is enriched with 12 Streaming Microprocessors (SM). It has 2304 CUDA cores and a memory of 3 GB. A total of 2048 threads can be generated in parallel by each streaming processor. These threads are executed in 64 warps and each warp has the capability to execute 32 threads in parallel. A local memory of 512 KB is possessed by each thread and there are 255 registers per thread. Each streaming microprocessor (SM) uses 16 thread blocks with 2048 bytes of shared memory per block.

The GT610 GPU has 48 CUDA cores with a memory of 1 GB. The architecture of this GPU is Fermi-based and has one streaming microprocessor. The streaming microprocessor can support a total of 8 thread blocks. It can support 48 warps per SM and each warp contains 32 threads. Each thread has a total of 63 registers and a local memory of 512 kb.

The dataset is self-generated consisting of videos of human faces of various individuals. The video streams recorded for the experiments are relatively simple (captured under controlled environmental conditions with faces posing towards a camera) and does not pose challenges such as illumination or head pose. The total video data used for the experimentation consists of one month of video streams. Each video stream has a duration of 120 s. The video streams are encoded with H.264 format. The frame rate for each video stream is 25 fps. The data rate and bitrate for each video stream are 421 kbps and 461 kbps respectively. The decoding of each video stream generates a frame set of 3000 video frames. Each video frame holds a data size of 371 kb.

7. Experimental results

This section explains the results obtained by executing the experiments with the dataset and the experimental setup with two different configurations described in Section 6. This section is further divided into three subsections. The first subsection explains the accuracy of the object classification system and the speedup achieved by the cropping process. The second subsection explains the throughput and performance of the system for video stream decoding, transfer of data between CPU to GPU and vice versa and performance gains achieved by utilizing the GPUs for compute intensive parts of the algorithm. The third subsection explains the scalability and robustness of the whole system by analysing decoded video streams and transferring the video data from local storage to cloud nodes. It also measures the time required to analyse video data on the cloud nodes and gathering the results after the completion of analysis. A discussion of the observations from these results is also provided in this section.

Table 1
Matching results of a person in multiple video streams.

Frames	VideoStream1	VideoStream2	VideoStream3
1	1	0.7437	0.7624
2	0.9613	0.7424	0.7594
3	0.9629	0.7434	0.758
4	0.963	0.7351	0.7546
5	0.9646	0.7339	0.7552
6	0.9665	0.7271	0.7573
7	0.9573	0.7266	0.7575
8	0.9525	0.7308	0.7512
9	0.9619	0.7285	0.7512
10	0.9453	0.7272	0.7626
AVG (120 s)	0.943	0.745	0.756
STD (120 s)	0.0256	0.0127	0.0128

7.1. Performance of the unsupervised object classification

The performance of the unsupervised object classification system is evaluated by measuring the accuracy to classify objects and the speedup achieved by the cropping process.

7.1.1. Object classification accuracy

The marked object which is to be identified in the video stream is matched with all the frames of a video stream. Each video stream (among three testing video streams) contains video frames of a single individual. The target face is present in the first testing video stream, the other two testing video streams have different individuals. The video streams recorded for the experiments are relatively simple (captured under controlled environmental conditions with faces posing towards camera) and do not pose challenges such as illumination or head pose. All the three testing video streams have different individuals in each video stream. It is to be noted that for a video stream with a frame per second rate of 25, we decoded only 5 frames per second. It is obvious that no change can occur in such a short interval of time, so processing all the frames would only increase the processing time. Table 1 shows the matching results of a marked object with multiple frames of multiple video streams.

The values in the columns represent the distance measure of marked object against different objects of multiple video streams using the LBPH algorithm. The values near to 1 depict a closer match of marked object. It can be seen from the table that all values in the column of video stream 1 are above 90%. This shows that the marked object is present in the video stream 1. On the other hand, all values in the second and third video streams are below 90% and depict that the marked object is not present in these video streams. We have used a threshold of 90% to distinguish between the matched and unmatched objects. Fig. 8 shows the video streams in which the marked object is most likely to reside.

It can be seen from the figure that video stream 1 has the highest probability of having the marked object. The other two streams are not probable to contain the marked object. Local binary pattern histogram hence provides a good measure for the presence of marked objects in video streams.

7.1.2. Cropped frame processing time

A significant amount of speedup is achieved in the processing time of each frame due to the object detection approach. Cropping

Table 2
Data transfer time from CPU to GPU and GPU to CPU.

CUDA streams	Data transfer time (in milliseconds)					
	CPU to GPU			GPU to CPU		
	Pageable	Pinned	Zero copy	Pageable	Pinned	Zero copy
1	0.113	0.104	0.001	0.123	0.1	0.001
2	0.212	0.117	0.025	0.16	0.151	0.011
3	0.321	0.208	0.12	0.311	0.233	0.0869
4	0.36	0.215	0.126	0.374	0.293	0.126
5	0.42	0.286	0.197	0.438	0.415	0.196
6	0.471	0.313	0.216	0.489	0.502	0.275
7	0.56	0.373	0.267	0.597	0.686	0.328
8	0.612	0.431	0.316	0.65	0.83	0.38
9	0.643	0.499	0.322	0.795	0.878	0.485
10	0.733	0.517	0.397	0.872	0.982	0.509

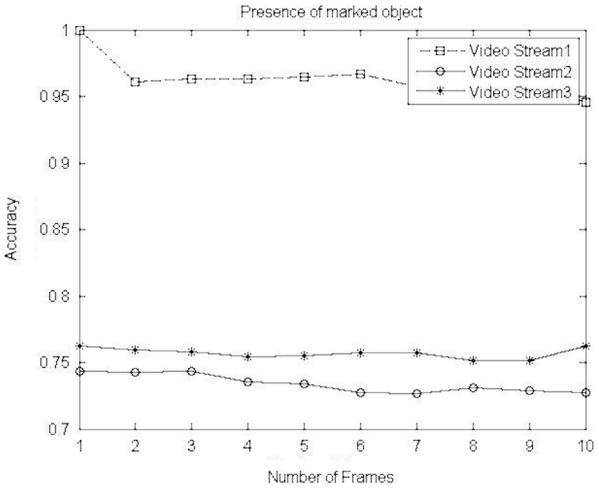

Fig. 8. Presence of marked object in multiple streams.

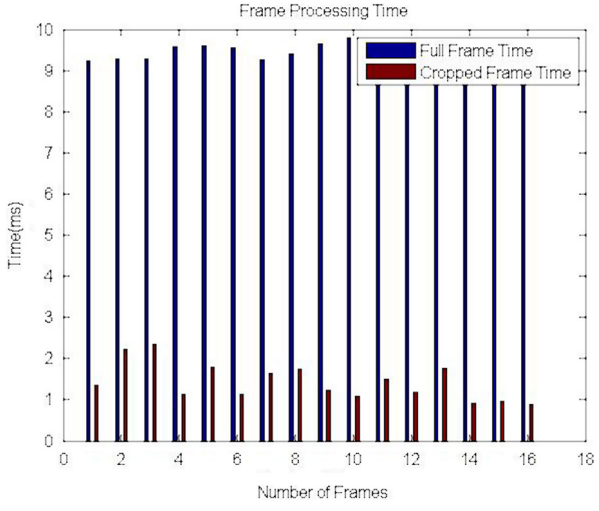

Fig. 9. Frame processing time of individual frames.

of a video frame around the detected object helped to reduce the processing area for the LBPH algorithm. The resolution of overall video frame is decreased which in turn reduced the overall processing time of each frame. The processing time of each individual frame before cropping and after cropping is calculated and is shown in Fig. 9.

The decrease in processing time is because of the fact that the resolution is reduced significantly because of cropping. The video used in this experiment had a frame resolution of 640 ×480. However, the detected object which was extracted from the whole frame and later used by LBPH for comparison had a resolution of around 160 ×160 in most of the cases. This decrease in resolution improved the total frame processing time by almost 90%.

7.2. Object classification on GPUs

This section describes the throughput and performance of the object classification system. The analysis of object classification system on GPU can be divided into two major steps (i) time required for decoding a video stream and transferring it from CPU to GPU memory, (ii) time required to process the video frame data for object classification. The performance measures of these two major steps are explained in the rest of this subsection.

7.2.1. Data transfer time

We have tested three different memory allocation techniques to transfer data from CPU to GPU and then back from GPU to CPU. The three techniques are page-able memory, pinned memory and zero copy. The effect of these three techniques has been demonstrated by varying the number of video streams from 1 to 10. A total memory allocation of 371.712 KBs is required by each video frame with a resolution of 704 × 528. A video stream recorded at 25 frames per second has a data transfer rate of 10.89 MB per second. For a varying number of video streams from 1 to 10, the data transfer per second varied from 10.89 to 108.9 MB. It has been observed that zero copy memory allocation technique remained fastest among the three techniques for transferring video frame data from CPU to GPU and vice versa. The time taken by each technique is summarized in Table 2.

7.2.2. Frame processing time

The total time taken to process an individual frame of a video stream is calculated by using the three memory allocation techniques discussed in the previous section. The total time required to process an individual video frame is the sum of time required to read and decode a frame, transfer time from CPU to GPU and GPU to CPU and the time required to compute local binary pattern of frame. Fig. 10 depicts the elapsed time of different frame processing operations by each memory allocation technique. It has been observed that zero copy remained the most efficient mechanism because of direct video frame data access from GPU to CPU. GPU memory address space is mapped to CPU memory address space in the zero copy mechanism, so a GPU can access CPU memory as its own address space. This mapping also enables the GPU to access a particular memory location in host memory whenever data is copied from host to device. The same procedure is followed to copy data back to the host from GPU memory.

Another way to quantify the performance of the system is to measure the number of frames processed per second. The number

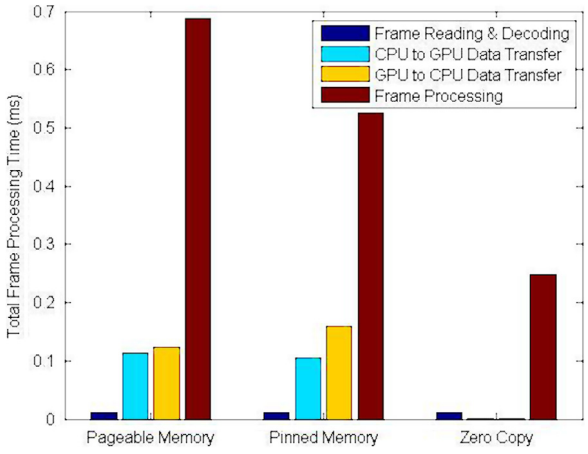

Fig. 10. Comparison across different memory allocation techniques.

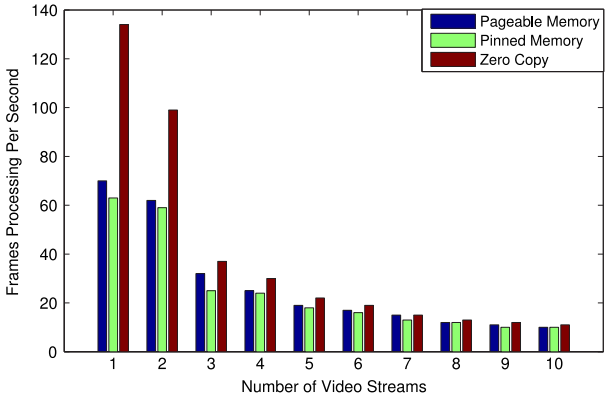

Fig. 11. Frame processing time and number of video streams.

of frames processed per second using the three memory allocation mechanisms is calculated and depicted in Fig. 11. As it was predicted, the highest throughput is achieved by the zero copy mechanism with varying number of video streams. It is observed that two video streams per GPU provided the most optimum performance by processing almost 100 frames per second. The data transfer time from CPU to GPU and GPU to CPU remained optimized with two CUDA streams as described in Table 2.

7.2.3. Computation time with varying video resolutions

The processing time of a video frame is highly dependent on the resolution of a video frame. For a high resolution video frame, more computation time is required as more data is needed to be processed. We have tested different video streams with varying resolutions on the system and computed the total processing time. This time includes the time required to process the frame as well as the video decoding time. The generated results are also compared with the results produced by stand-alone CPU node as depicted in Fig. 12.

It has been observed that optimum utilization of GPUs can be achieved by having the videos with high resolution. The processing of low resolution videos on GPUs will not generate much speedup as compared to CPUs. This is because of the fact that a CPU processes each pixel sequentially. On the other hand a GPU performs the processing of pixels in parallel by mapping each pixel to individual thread. This elevates the processing speed of individual frames. However, if the video frame is of low resolution, no significant speedup in the processing time of video frame is observed as compared to CPU due to data transfer overheads.

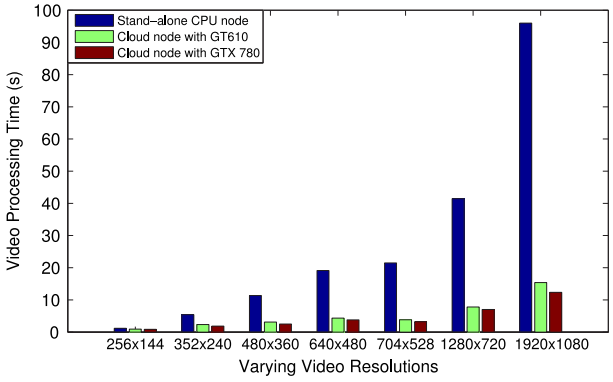

Fig. 12. Video processing comparison on different platforms.

7.3. Object classification on the cloud

In order to evaluate the scalability of our approach, we have executed it on the cloud infrastructure described in the experimental setup section. The evaluation is performed on the following three parameters. (i) Time taken to transfer video stream data from storage server to the cloud nodes, (ii) Analysis time of video streams on cloud nodes, (iii) Time required to collect results from cloud nodes. Hadoop File System (HDFS) is used for storing files. The MapReduce framework is used to analyse video streams by executing unsupervised object classification algorithm explained in Section 3. The analysis results are then stored in the database.

7.3.1. Hadoop sequence file creation

The video streams are first decoded to extract individual video frames from the input video. The total size of one month of recorded video streams is 175 GB. Each video stream is recorded at 25 frames per second. The number of decoded video frames is dependent upon the length of video stream being analysed. These individual frames are not suitable for directly processing on the compute nodes with the MapReduce framework. This is because of the fact that MapReduce is designed to process large files. Processing small files will only result in the decrease of overall performance. These small files are bundled into a large file referred to as Hadoop sequence file and then transferred to the cloud nodes for processing. The sequence file is then moved to cloud storage for unsupervised object classification.

7.3.2. Hadoop sequence file creation time

The time required to generate a sequence file is directly proportional to the size of dataset. Multiple datasets of varying sizes from 5 to 175 GB have been used in this paper to generate results. The dataset of varying sizes helped to evaluate numerous aspects of our system. The time taken to create a sequence file for sizes ranging from 5 to 175 GB varied from 6.15 min to 10.73 h respectively. The larger the dataset, more time it will require to generate the sequence file. However, it is a one-time process and once the sequence file has been generated, it can be stored in the cloud data storage for future analysis tasks.

7.3.3. Sequence file transfer time

The generated sequence file is moved to cloud data storage as object classification will be performed on cloud nodes. The transfer time required to transfer the file to cloud data storage depends on various parameters. These parameters include network bandwidth, data replication factor and cloud data storage block size. The data transfer time varies with the size of the dataset. For the dataset sizes reported in this paper (5–175 GB), the data transfer

Please cite this article in press as: M.U. Yaseen, et al., Cloud-based scalable object detection and classification in video streams, Future Generation Computer Systems (2017), http://dx.doi.org/10.1016/j.future.2017.02.003.

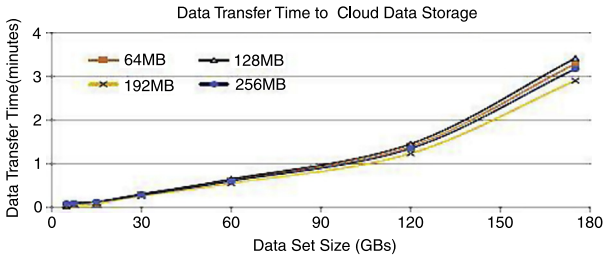

Fig. 13. Data transfer time to cloud storage.

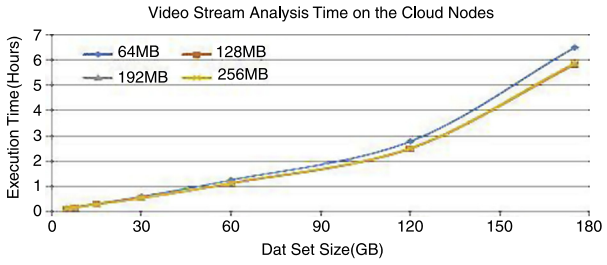

Fig. 14. Video stream analysis time on cloud nodes.

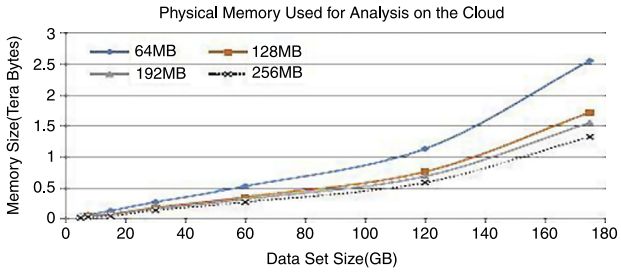

Fig. 15. Memory consumed for analysis in the cloud.

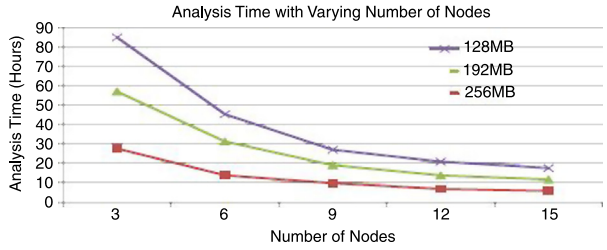

Fig. 16. Analysis time with varying number of cloud nodes.

time varied from 2 min to 3.17 h. Fig. 13 depicts the data transfer time of various dataset sizes with varying cloud storage block size.

7.3.4. Object classification on cloud nodes

We have evaluated the scalability and robustness of the system by executing object classification on large numbers of video streams. The datasets have also been varied from 5 to 175 GB to observe the effects on the cloud nodes. The HDFS block sizes have also been varied to measure the execution time and resources consumed during the analysis tasks on cloud nodes. The performance of the system is measured by monitoring the time required to analyse the dataset of various sizes and the resources consumed during the analysis task.

We have varied the block size from 64 to 256 MB, in order to observe the effect of varying block size on Map task execution. It has been observed that the execution time of Map task increases by increasing the size of dataset as depicted in Fig. 14. But the variation in block sizes has no major impact on the execution time of Map/Reduce tasks. For the dataset size varying between 5 and 175 GB, the total execution time varied between 6.38 min and 5.83 h.

The memory consumption of all the block sizes remained the same except for the 64 MB block. The requirement of physical memory for the 64 MB block size is higher than other block sizes as depicted in Fig. 14. The default block size of cloud storage is 128 MB. A 64 MB block size thus produces more data blocks which are needed to be processed by cloud nodes causing memory overhead. More memory is required to process small block sizes as the number of map tasks turn out to be deficient with the smaller block sizes. Fig. 15 shows the memory required with varying datasets for analysis on the cloud.

7.3.5. Robustness with changing cluster size

The robustness of the system is evaluated by measuring the total analysis time and the speed-up achieved by increasing the number of cloud nodes. We have measured the total time required for the analysis of dataset with varying number of nodes. The total analysis time of whole dataset decreases as the number of nodes increases in the cloud. Table 3 shows the execution time required to analyse the dataset with varying nodes.

Table 3
Analysis task execution time with varying cloud nodes.

Nodes	Tasks per node	Tasks execution time (h)
15	94	5.83
12	117	7.10
9	156	7.95
6	234	14.01
3	467	27.80

We have also measured the total time required for analysis of whole dataset with varying number of nodes and block sizes. Fig. 16 depicts that the execution time decreases as the number of nodes in the cloud increases. A decreasing trend has been observed in the analysis of whole dataset. A total execution time of 27.80 h was required for the processing of 175 GB dataset with 3 nodes, whereas, it took only 5.83 h to process the same amount of data with a 15 node cloud.

7.3.6. Task parallelism on compute nodes

The total number of analysis tasks executing on a compute node is directly proportional to the number of input splits. The number of input splits are further dependent on the dataset size, cloud data storage block size and available physical resources. The dataset size of 175 GB gives rise to 1400 map/reduce tasks with a default cloud storage block size of 128 MB. It has been observed during the experiments that the number of analysis tasks on each node increases as the number of nodes decreases. We varied the number of nodes between 3 and 15 in these experiments. As the number of tasks per node increases, the performance of the overall system degrades. This is because of the fact that the increase in number of tasks per node saturates resources and each subsequent task has to wait longer for scheduling and execution. A summary of task execution time corresponding to a varying number of nodes is shown in Table 3.

We have also calculated the analysis time of varying datasets with varying block sizes. It is observed that if the block size is large, less computation time will be required to analyse the data

Please cite this article in press as: M.U. Yaseen, et al., Cloud-based scalable object detection and classification in video streams, Future Generation Computer Systems (2017), http://dx.doi.org/10.1016/j.future.2017.02.003.

as compared to smaller block size. The large block size will have less number of map tasks, reduced memory requirement and management overhead as compared to small block size. This will result in the faster processing of dataset. However, it is to be noted that varying block sizes does not affect the execution time of Map task. The block size of 512 MB required the same processing time as 256 MB block size for the 175 GB dataset. The same phenomenon is observed with other block sizes as well. However, the time required to transfer the data with larger block sizes is greater and required larger compute nodes to process the data.

8. Conclusion & future work

A cloud based video analysis system based on Haar Cascade Classifier and the Local Binary Pattern Histogram is presented in this paper. The proposed system requires minimum human interaction and provides automated object classification from large number of video streams. The system performs classification under unsupervised learning domain and without requiring any metric learning stage or labelled training dataset. An accuracy of more than 95% is achieved when the application is tested on multiple video streams.

The proposed system is capable of coping with the challenges of increased volume of data. The objects are detected and classified from one month of video data comprising a size of 175 GB. It took 6.52 h to analyse this data on a 15 node cloud. By increasing the number of nodes in the cloud, a decreasing trend in processing time is observed in analysing the video data. A reduction from 27.80 to 5.83 h is observed, when the number of cloud nodes increased from 3 to 15. However, the analysis time is also dependent on the amount of data being analysed. The analysis time varied from 6.38 min to 5.83 h for the dataset sizes ranging from 5 to 175 GB in the cloud.

The processing time further reduced to 3 h for 175 GB data when the video stream analysis is performed on GPU mounted cloud nodes. Several factors contributed to achieving high throughput such as optimized resource utilization of GPUs, efficient and optimal data transfer techniques, improved occupancy and efficient memory allocation. The mapping of each pixel of a video frame to individual light-weight GPU threads played a major role in achieving high performance in the system.

In future, we would like to make the system more generic by detecting and recognizing other objects from different object classes such as cars, bicycles and pedestrians. The optimization of detection and recognition algorithms by analysing them in the frequency domain will also be the focus of our future work. We would also like to achieve more speed-up and scalability by using in-memory processing cluster coupled with the computation power of GPUs. This will help to overcome the delays which occur due to various I/O operations.

References

[1] L. Zhang, M. Yang, X. Feng, Sparse representation or collaborative representation: Which helps face recognition, in: IEEE International Conference on Computer Vision, 2011, pp. 471–478.
[2] X. Zhu, D. Ramanan, Face detection, pose estimation, and landmark localization in the wild, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2012, pp. 2879–2886.
[3] H. Cevikalp, B. Triggs, V. Franc, Face and landmark detection by using cascade of classifiers, in: IEEE International Conference on Face and Gesture Recognition, 2013, pp. 1–7.
[4] J.S. Bae, T.L. Song, Image tracking algorithm using template matching and PSNF-m, Int. J. Control Autom. Syst. 6 (2008) 413–423.
[5] Project BESAFE (Behavior lEarning in Surveilled Areas with Feature Extraction), http://imagelab.ing.unimore.it/besafe/. (Last accessed: 20 February 2016).
[6] A. Jain, B. Klare, U. Park, Face recognition: Some challenges in forensics, in: IEEE International Conference on Automatic Face and Gesture Recognition and Workshops, 2011.
[7] Y. Taigman, M. Yang, L. Wolf, DeepFace: Closing the gap to human-level performance in face verification, in: IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1701–1708.
[8] H. Wang, X. Gu, X. Li, Z. Li, Occluded face detection based on adaboost technology, in: IEEE Eighth International Conference on Internet Computing for Science and Engineering, 2015, pp. 87–90.
[9] R. Lienhart, L. Liang, A. Kuranov, A detector tree of boosted classifiers for real-time object detection and tracking, in: International Conference on Multimedia and Expo, 2003, pp. 277–280.
[10] A. Suruliandi, K. Meena, R. Reena, Local binary patterns and its derivatives for face recognition, IET Comput. Vis. (2012) 480–488.
[11] Y. Lin, F. Lv, S. Zhu, M. Yang, L. Cao, Large-scale image classification: Fast feature extraction and SVM training, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2011, pp. 1689–1696.
[12] A. Zamani, M. Zou, J. Diaz-Montes, I. Petri, O. Rana, A. Anjum, M. Parashar, Deadline constrained video analysis via in-transit computational environments, IEEE Trans. Serv. Comput. (2017).
[13] G.H. Nguyen, S.L. Phung, A. Bouzerdoum, Reduced training of convolutional neural networks for pedestrian detection, in: The 6th International Conference on Information Technology and Applications, 2009, pp. 61–66.
[14] H. Tan, L. Chen, An approach for fast and parallel video processing on apache hadoop clusters, in: IEEE International Conference on Multimedia and Expo, 2014, pp. 1–6.
[15] X. Tang, Z. Ou, T. Su, P. Zhao, Cascade AdaBoost classifiers with stage features optimization for cellular phone embedded face detection system, Adv. Nat. Comput. 3612 (2005) 688–697.
[16] B. Zhou, W. Wang, X. Zhang, Training backpropagation neural network in MapReduce, in: International Conference on Computer, Communications and Information Technology, 2014, pp. 22–25.
[17] A. Hervé, L. Williams, Principal component analysis, in: Wiley Interdisciplinary Reviews: Computational Statistics, 2010, pp. 586–591.
[18] M.A. Turk, A.P. Pentland, Face recognition using eigenfaces, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1991, pp. 586–591.
[19] F. Tsalakanidoua, D. Tzovarasb, M.G. Strintzis, Use of depth and colour eigenfaces for face recognition, Pattern Recognit. Lett. (2003) 1427–1435.
[20] A.M. Martinez, A.C. Kak, PCA versus LDA, IEEE Trans. Pattern Anal. Mach. Intell. (2001) 228–233.
[21] B. Marian, M. Javier, S. Terrence, Face recognition by independent component analysis, IEEE Trans. Neural Netw. (2002) 1450–1464.
[22] http://v-i-systems.com/. (Last accessed: 24 November 2016).
[23] http://www.smartcctvltd.com/. (Last accessed: 24 November 2016).
[24] https://www.besafe.uk.com/. (Last accessed: 24 November 2016).
[25] http://www.intelligentvision.com.au/. (Last accessed: 24 November 2016).
[26] http://www.boschsecurity.com/. (Last accessed: 24 November 2016).
[27] R. McClatchey, A. Anjum, H. Stockinger, A. Ali, I. Willers, M. Thomas, Data intensive and network aware (DIANA) grid scheduling, J. Grid Comput. (2007) 43–64.
[28] W. Wang, W. Zhang, F. Gong, P. Zhang, Y. Rao, Towards a pervasive cloud computing based food image recognition, in: IEEE International Conference on Cyber, Physical and Social Computing, 2013, pp. 2243–2244.
[29] J. Fan, F. Han, H. Liu, Challenges of big data analysis, Natl. Sci. Rev. (2013) 293–314.
[30] S. Sotiriadis, N. Bessis, A. Anjum, R. Buyya, An inter-cloud meta-scheduling (icms) simulation framework: Architecture and evaluation, IEEE Trans. Serv. Comput. (2015).
[31] L. Zhu, X. Zheng, P. Li, Y. Wang, A cloud based object recognition platform for IOS, in: International Conference on Identification, Information and Knowledge in the Internet of Things, 2014, pp. 68–71.
[32] A. Anjum, T. Abdullah, M.F. Tariq, Y. Baltaci, N. Antonopoulos, Video stream analysis in clouds: An object detection and classification framework for high performance video analytics, IEEE Trans. Cloud Comput. (2015) 1–14.
[33] R. Raina, A. Madhavan, A.Y. Ng, Large scale deep unsupervised learning using graphics processors, in: 26th ACM Annual International Conference in Machine Learning, 2009, pp. 873–880.
[34] W. Wang, Y. Zhang, S. Yan, H. Jia, Parallelization and performance optimization on face detection algorithm with OpenCL: A Case Study, Tsinghua Sci. Technol. 17 (3) (2012) 287–295.

Please cite this article in press as: M.U. Yaseen, et al., Cloud-based scalable object detection and classification in video streams, Future Generation Computer Systems (2017), http://dx.doi.org/10.1016/j.future.2017.02.003.

[35] Z. Jing, J. liangbao, C. Xuehong, Implementation of parallel full search algorithm for motion estimation on multi-core processors, in: IEEE 2nd International Conference on Next Generation Information Technology, 2011, pp. 31–35.
[36] A. Mohamed, G. Dahl, G. Hinton, Acousting modeling using deep belief networks, IEEE Trans. Audio, Speech Lang. Process. 20 (1) (2011) 14–22.
[37] S. Gao, I. Tsang, L. Chia, Laplacian sparse coding, hypergraph laplacian sparse coding and applications, IEEE Trans. Pattern Anal. Mach. Intell. (2012) 92–104.
[38] D. Oro, C. Fernandez, J.R. Saeta, X. Martorell, J. Hernando, Real time GPU based face detection in HD video sequences, in: IEEE International Conference on Computer Vision Workshops, 2011, pp. 530–537.
[39] C. Rosenberg, M. Hebert, H. Schneiderman, Semi-supervised self-training of object detection models, in: IEEE International Conference on Application of Computer Vision, 2014.
[40] B. Pfahringer, G. Holmes, E. Frank, Classifier chains for multi-label classification, in: Machine Learning, Springer, 2011, pp. 333–359.
[41] G. Sharma, F. Jurie, A novel approach for efficient SVM classification with histogram intersection kernel, in: British Machine Vision Conference, 2013.
[42] D. Nguyen Ta, W. Chen, N. Gelfand, K. Pulli, SURFTrac: Efficient tracking and continuous object recognition using local feature descriptors, in: IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 2937–2944.
[43] T. Wu, A. Toet, Speed-up template matching through integral image based weak classifiers, J. Pattern Recognit. Res. 12 (2014).

Muhammad Usman Yaseen is a Ph.D. student at the University of Derby UK. His research interests include video analytics, big data analysis, machine learning and distributed systems.

Ashiq Anjum is a professor of Distributed Systems at the University of Derby UK. His research interests include Data Intensive Distributed Systems, Parallel Computing and High Performance Analytics platforms. Currently he is investigating high performance distributed platforms to efficiently process video and genomics data.

Omer Rana is a professor of Performance Engineering at the Cardiff University UK. His research interests include Problem Solving Environments (PSEs) for computational science and commercial computing, Data analysis and management for large scale computing, and scalability in high performance agent systems.

Richard Hill is a professor of Intelligent Systems at the University of Derby UK. His research interests include Cloud computing, big data analytics, data science, multi-agent systems, High Performance Computing.

Please cite this article in press as: M.U. Yaseen, et al., Cloud-based scalable object detection and classification in video streams, Future Generation Computer Systems (2017), http://dx.doi.org/10.1016/j.future.2017.02.003.