Real-Time Load Reduction in Multimedia Big Data for Mobile Internet

KUN WANG, JUN MI, CHENHAN XU, and QINGQUAN ZHU, Nanjing University of Posts and Telecommunications
LEI SHU, Guangdong University of Petrochemical Technology
DER-JIUNN DENG, National Changhua University of Education

In the age of multimedia big data, the popularity of mobile devices has been in an unprecedented growth, the speed of data increasing is faster than ever before, and Internet traffic is rapidly increasing, not only in volume but also in heterogeneity. Therefore, data processing and network overload have become two urgent problems. To address these problems, extensive papers have been published on image analysis using deep learning, but only a few works have exploited this approach for video analysis. In this article, a hybrid-stream model is proposed to solve these problems for video analysis. Functionality of this model covers Data Preprocessing, Data Classification, and Data-Load-Reduction Processing. Specifically, an improved Convolutional Neural Networks (CNN) classification algorithm is designed to evaluate the importance of each video frame and video clip to enhance classification precision. Then, a reliable keyframe extraction mechanism will recognize the importance of each frame or clip, and decide whether to abandon it automatically by a series of correlation operations. The model will reduce data load to a dynamic threshold changed by σ, control the input size of the video in mobile Internet, and thus reduce network overload. Through experimental simulations, we find that the size of processed video has been effectively reduced and the quality of experience (QoE) has not been lowered due to a suitably selected parameter η. The simulation also shows that the model has a steady performance and is powerful enough for continuously growing multimedia big data.

CCS Concepts: • **Networks** → **Hybrid-stream model**; **Data path algorithms**; • **General and reference** → Design

Additional Key Words and Phrases: Multimedia, big data, mobile Internet, real-time, load reduction, networking

ACM Reference Format:
Kun Wang, Jun Mi, Chenhan Xu, Qingquan Zhu, Lei Shu, and Der-Jiunn Deng. 2016. Real-time load reduction in multimedia big data for mobile Internet. ACM Trans. Multimedia Comput. Commun. Appl. 12, 5s, Article 76 (October 2016), 20 pages.
DOI: http://dx.doi.org/10.1145/2990473

Support was provided by NSFC (61572262, 61100213, 61571233, 61373135, 61572172); SFDPH (20113223120007); NSF of Jiangsu Province (BK20141427), NUPT (NY214097); Priority Academic Program Development of Jiangsu Higher Education Institutions; Open Research Fund of Key Lab of Broadband Wireless Communication and Sensor Network Technology (NUPT), Ministry of Education (NYKL201507); Jiangsu Qing Lan Project; Educational Commission of Guangdong Province (2013KJCX0131); Guangdong High-Tech Development Fund (2013B010401035); International and Hong Kong, Macao & Taiwan collaborative innovation platform and major international cooperation projects of colleges in Guangdong Province (No. 2015KGJHZ026); and the 2014 Guangdong Province Outstanding Young Professor Project and 2013 Top Level Talents Project in the Sailing Plan of Guangdong Province.
Authors' addresses: K. Wang, J. Mi, C. Xu, and Q. Zhu, Key Lab of Broadband Wireless Communication and Sensor Network Technology, Nanjing University of Posts and Telecommunications, Nanjing, 210003, China; emails: kwang@njupt.edu.cn, junmiia@163.com, xchank@outlook.com, njuptzqq@163.com; L. Shu, Guangdong Provincial Key Lab of Petrochemical Equipment Fault Diagnosis, Guangdong University of Petrochemical Technology, Guangdong, 525000, China; email: lei.shu@ieee.org; D.-J. Deng, National Changhua University of Education, Taiwan; email: djdeng@cc.ncue.edu.tw.

DOI: http://dx.doi.org/10.1145/2990473

1. INTRODUCTION

We are now living in the era of multimedia data. Data is undergoing an unprecedented upsurge. Mobile devices are becoming mainstream, and Internet traffic has experienced an exponential growth in volume as well as in heterogeneity [Hernandez 2010]. With the growth trends, though we benefit significantly from such data resources, many problems may appear simultaneously, for example, data processing and network overload. Many aspects of data, including storage, transmission, processing and application, are confronted with great challenges. Data processing and analyzing are the most urgent problems.

Multimedia big data consist of both structured and unstructured data [Han et al. 2016; Pan et al. 2015]. One of the challenges in dealing with image, video, and audio is the much more complex algorithms used for content analysis than ever before. Generally, 1s of a video displays at least 25 images [Lee et al. 2015; Chen et al. 2015]. Furthermore, the video size inevitably increases if higher resolution is required when users ask for better quality of experience (QoE) [Dong et al. 2015; K. Wang et al. 2015; Zhang et al. 2011]. A novel data-processing method that can be applied to reduce data volume and network load is now urgently needed.

To address these problems, methods related to traffic analysis and anomaly detection become critical in mobile Internet. Generally, network-overload problems caused by big data can be addressed by two strategies. One is dispersing blocked traffic in advance by optimizing route selection; the other is recognizing abnormal traffic and abandoning it before transmission. The vertical handoff (VHO) decision algorithm has a good performance in heterogeneous traffic [Xia et al. 2012], but when new metrics are introduced, the video high-density (VHD) process will become more complex. The Vehicle Classifier and Traffic flow analyzeR (VECTOR) is useful for online traffic flow analysis [Morris and Trivedi 2008], but one challenge is how to apply it to video analysis. When it comes to the other strategy, the primary task is to correctly classify the videos. The Convolutional Neural Network (CNN), a typical feed-forward neural network, outperforms in classifying 2D shapes [He et al. 2015]. In literature, training of a CNN is rather time-consuming due to the high complexity of the model. The hybrid convolutional neural network (HCNN), a model which combines the original CNN and winner-takes-all mechanism, was proposed to simplify the original CNN model, so that it can acquire improved speed and better precision of pattern recognition [Mrazova and Kukacka 2008; Zheng et al. 2015]. The main problem of the model is that it is only suitable for 2D shapes.

Compared with extensive studies on using deep learning for image analysis, only a few works have exploited this approach for video analysis [Ji et al. 2013; Karpathy et al. 2014]. A two-stream CNN structure was proposed to process video by dividing the original video information into spatial information and temporal information [Simonyan and Zisserman 2014]. Then, an improved two-stream model aimed at dealing with video classification was proposed [Ye et al. 2015]. The model achieves competitive performance by training two CNNs, but the fusing multiple network model still needs to be improved when in a strong network, in which the traffic is very heavy.

In this article, we address the problem of mobile Internet overload [Dong et al. 2014], especially for video in multimedia big data [Li et al. 2015]. We take the second strategy discussed earlier. Abnormal traffic can be considered as unimportant frames or clips in the video streams, which can be abandoned without affecting users' QoE. Specifically, in contrast to conventional single input [H. Wang et al. 2015], we work with a two-stream model that divides the input information into spatial and temporal information, and we set two inputs to separately deal with input videos' different information. One input deals with static information, such as scenes and objects; the other deals with dynamic information, such as motion. We consider that the video stream is made up of numerous

frames or clips, and each frame or clip can be viewed as images. Then, we can analyze and monitor the abnormal traffic in mobile Internet by recognizing these images, and thus solve the problem of network overload. Based on these considerations, we propose a hybrid-stream model that contains a reliable keyframe extraction mechanism and an improved CNN classification algorithm to enhance classification precision and relieve the network overload.

However, there are several challenges when performing video analysis. The primary challenge is how to improve the video classification's precision. Generally, most video classification algorithms use low-dimensional representations for the sake of reducing the videos' dimensionality [Baktashmotlagh et al. 2014; B. Gu and V. Sheng 2016]. In our model, we propose an improved CNN algorithm in which we add a time dimension, forming a 3-dimensional matrix, comparing with conventional CNN algorithm's 2-dimension image analysis, to improve classification precision. How to deal with the classified outcomes to reduce network overload becomes another problem. In our model, we put forward a keyframe extraction mechanism to solve the problem. Through the mechanism, we try to find a suitable output number that can get a reduced load and does not affect users' QoE.

For video processing [Wang et al. 2016; Xia et al. 2016], we consider four steps in this article: preprocessing, classification, recognition, and extraction. The main goal of preprocessing is to make input video streams suitable for our model. With classification we aim to describe the importance of each frame or clip; recognition and extraction are supposed to realize network overload reduction. In summary, we mainly focus on real-time video processing to relieve network overload in mobile Internet. The contributions of our article are as follows:

—A hybrid-stream model considered to solve the network overload problem in mobile Internet is proposed. We make use of different types of input to improve classification precision.
—An improved CNN classification algorithm, which is used to recognize video information, is proposed. In the algorithm, we add a time dimension, forming a 3-dimensional matrix to classify video frames and clips.
—A keyframe extraction mechanism is proposed. We aim to realize load reduction and better QoE performance by adjusting suitable parameters.

The rest of this article is organized as follows. In the next section, we review the related works. Section 3 outlines our system model. In Sections 4 and 5, we describe the classification module and load-reduction module in detail, respectively. In Section 6, different types of comparative results are provided to show system performance. In Section 7, we present our conclusions.

2. RELATED WORK

In this section, we summarize existing methods concerning network traffic-load balancing. We also discuss some solutions about abnormal traffic detection. Finally, we summarize the major methods proposed to realize real-time problems.

2.1. Network Traffic Load Balancing

In mobile Internet, in order to pursue higher throughout and reduce unexpected end-to-end delay, a common approach is to balance traffic among each mobile node. Some popular themes about load balancing have been well studied recently. One is based on flow hashing [He et al. 2015], and another is based on swarm intelligence (SI) [Rango and Tropea 2009].

In terms of flow hashing, Equal Cost Multipath Routing (ECMP) has drawn many researchers' attention [He et al. 2015] because congestion easily occurswhen hash

collisions happen. A centralized scheme is proposed [Al-fares et al. 2010; Fu et al. 2015], in which the main idea is collecting the state of the network, then rerouting overflows when the network is busy. The method is simple and easy to realize, but the main problem is its large time constraints and extra network infrastructure. Another theme that concerns in-network reactive distributed load balancing is proposed that solves the time-constraint problem, but still has special requirements on networking hardware [Alizadeh et al. 2014].

In terms of SI, the Ant-Based Control algorithm (ABC algorithm) shows good performance on a network in which its dynamic data have a very fast change [Kroon 2002]. Other works based on SI have been proposed in the literature, such as AntNet [Kroon 2002], an ant-routing algorithm for mobile ad-hoc network (ARAMA) [Hussein et al. 2005]. AntNet is an adaptive agent-based routing algorithm that has a good performance on packet-switched communications networks. ARAMA is a biologically based routing algorithm, which aims to balance the distribution of energy usage and optimize the number of hops. However, these algorithm cannot adopt different metrics. Combined with these algorithms, load balancing and energy aware ARAMA (LBE-ARAMA), a novel probabilistic routing algorithm based on SI, is proposed to solve these problems through a cooperation mechanism [Rango and Tropea 2009].

When the network traffic mainly comprise videos, a queue-occupancy-based load-balancing solution (ViLBaS) is proposed [Hava et al. 2015]. It is a new selective load-balancing solution for video; to acquire better QoE, it reroutes flows selectively around congested nodes. However, unbalanced traffic distributions reduce the efficiency of a network because of overloading in some network nodes, and classic routing solutions take no consideration of load; thus, the network may get overloaded rapidly. If we use a load-balancing strategy, time constraint and extra network infrastructure problems should be considered. In this article, we use another strategy that is similar to abnormal traffic detection.

2.2. Abnormal Traffic Detection

Traffic flow is an important feature for monitoring network anomalies, and it is easily affected due to the randomness of networks. Abnormal traffic in multimedia big data causes a bad QoE for users [Wang et al. 2014]. Therefore, research on abnormal traffic detection is important and necessary, and attracts many researchers. A bidirectional flow model aimed to extract pivotal traffic metrics and reduce flow records in large-scale networks [Qin et al. 2010; Wang and Yu 2013]. By monitoring the dynamic changes of traffic patterns, the model obtains main traffic flow patterns, then detects potential anomalies to reduce the flow records. Another method using the core concept of origin-destination flow (ODFLOW) is useful for large-scale network attack detection, but has no prominent effect on traffic volumes [Lakhina et al. 2005]. Lakhina et al. [2005] mainly finished the classifying works on traffic flows by analyzing the ODFLOW matrix using principle component analysis (PCA). However, studies on the remaining work, such as separating abnormal and normal network traffic, are very challenging. A blind source separation method was presented to separate the behavior indices [Qin et al. 2009]. In this method, the behavior indices contained in the traffic patterns correspond to the unobserved input signals, and the observed output signals are traffic features extracted from measurement. Another feature of this method is in no need of supervised learning process. In another method based on volume, Zeb et al. [2014] investigated the long-range dependence (LRD) behavior of Internet traffic to detect volume-based anomalies. Song and Liu [2014] proposed a dynamic k-NN cumulative-distance abnormal detection algorithm. It shows good performance in a high-speed network. In this article, in contrast to almost abnormal traffic detection discussed earlier, we aim to detect high correction traffic and process it to lower the whole traffic volume in mobile Internet.

2.3. Real-Time Systems

The method of detecting abnormal information in massive traffic data in real time is not well studied. Almost all real-time systems are good at operating on continuous data queries, but they have time-constraint problems in data attributes [Chiang et al. 2015; Wang et al. 2016]; due to the dynamic nature of data transmission, real-time data queries tend to need more unexpected execution cost. However, nonreal-time systems easily acquire fairly good QoE, while real-time systems face many challenges. One of these fundamental challenges is the enormous data size in mobile Internet. The common solution is maximizing a compression rate. Based on that idea, many video- and image-compression algorithms have been proposed. In addition, it is not the only solution for real-time multimedia to stream over mobile Internet [Dong et al. 2015; Zhao et al. 2015]. Distinguishing important and unimportant aspects of video frames to minimize the data size is another solution. In terms of real time, using compression to decrease the data size is far from sufficient, because the mobile environment has the feature of changeability. Lee et al. [2015] proposed a content-based algorithm for fine granular scalable coding. The proposed algorithm analyzes video contents to preserve the more important part of the video and discard the less important part. The problem with this algorithm is that it is not fully real time. In addition, an adaptive bitrate streaming algorithm can divide multimedia contents into small pieces, and cognitive streaming abstraction (CoSA) applies this algorithm in mobile Internet [Lee et al. 2015]. It works without degrading the QoE compared with other real-time systems. Chiang et al. [2015] presented an online scheduling algorithm to maximize the total number of satisfied users in asymmetric communication environments with time requirements. It not only focused on a data-broadcasting approach, but also considered the data items with time constraints. In a real-time system, the arriving data are unpredictable, which leads to variable input volumes. What we do in this article is process these data in advance to obtain a controlled data volume.

3. SYSTEM DESIGN

In this section, we first clarify the problem that video analysis always encounters. Then, we propose a model to solve the problem. Next, we give a brief system overview.

3.1. Design Goals

We aim to establish a model for videos to realize video analysis and data load–reduction processing. In order to achieve these goals, we first need to preprocess raw video resources and improve the conventional CNN algorithm for video analysis. In addition, we need a rule mechanism to complete video volume reduction. The following metrics are also important for processing performance.

Accuracy of classification: In the model, the classification accuracy directly affects the following data load–reduction processing.

Time-efficiency: With large amounts of data resources, the algorithm should be more powerful and fast.

Data-Load Reduction: Since the final goal is to reduce network overload, the processed data volume of transmission should be evaluated.

3.2. System Overview

The primary problem of mobile Internet is the increasing data. During transmission, too much data traffic inevitably causes network overload. We focus on the network load problem and try to conduct a study on video analysis and processing. Generally, a video sequence normally contains a large number of frames. In order to avoid network overload and ensure people's watching quality simultaneously, a mechanism that can

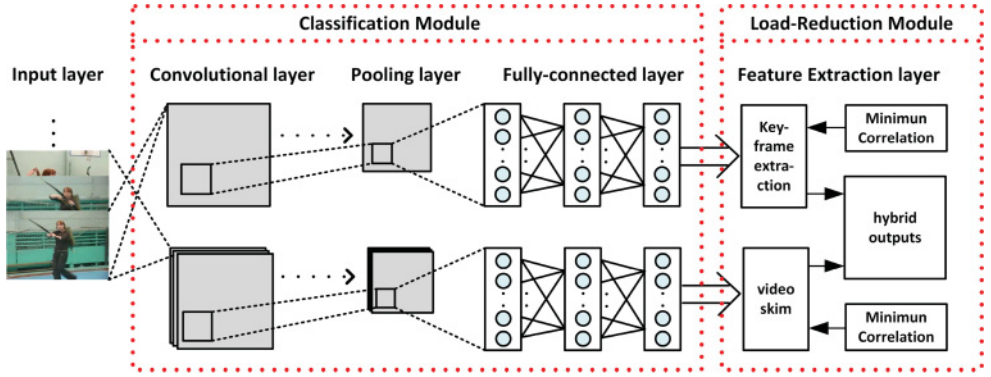

Fig. 1. Hybrid-stream model.

recognize the importance of the frame and then decide whether to drop the frame should be established. However, when it comes to how to represent the importance of a frame or a clip, an improved CNN algorithm that differs from conventional image-processing algorithms should be designed to acquire more accurate classifications.

We design a hybrid-stream model, as shown in Figure 1. The functionality of the hybrid-stream model can be summarized as three parts: Data Preprocessing (Input Module), Data Classification (Classification Module) and Data Load–Reduction Processing (Load-Reduction Module). Data Preprocessing deals with raw video resources. In order to improve classification precision, we divide raw video resources into two input forms: video frames and video clips. Video clips generally consist of several video frames. Data Preprocessing transforms the raw video resources into another form, which is suitable for Data Classification. Then, Data Classification Processing mainly further processes the two input video streams, then outputs values that can represent every frame's or clip's information. In the end of the classification module, we get two types of value, which can be combined in a load-reduction module to produce a new key feature parameter. Data Load–Reduction Processing is the final processing step for deciding whether and how to drop a frame.

Generally, video data, sequential image frames, can be divided into spatial and temporal components. The spatial component includes scenes and objects, and the temporal component contains motion information. Figure 1 shows a hybrid-stream model. In the figure, we consider the video as two type of streams: spatial streams and timing-sequence streams. Spatial streams include all kinds of scenes and objects contained in the image, and timing-sequence streams are motion information contained in video clips.

At the top of Figure 1, we show the processing of spatial streams. This structure is similar to deep CNN for image classification. A single video image frame as inputs goes through several convolutional layers, pooling layers and fully connected (FC) layers. Finally, the network outputs a value that represents the information of the video image frame. For timing-sequence streaming, in contrast to spatial streams with video image frames as input, we use several continuous frames called video clips as input to capture the motion information. Apart from input, the entire structure of processing timing-sequence streams is basically the same as the spatial stream part.

We summarize all notations containing in this article in Table I.

3.3. Definitions

In this section, we define and describe the input form of every module.

Table I. Summary of Notation

f_i	the video frame
w	the collection of all weights in the network
$w_{i,j}$	the weight of 2-dimension
$w_{i,j,k}$	the weight of 3-dimension
d	the dimension
b	the bias
λ	the scaling factor that can be learned
α	feedback of the last result
$h_{w,b}(x)$	the desired output from the network
$\|\upsilon\|$	the usual length cost function for a vector υ
ϱ	the learning rate
N	the total number of training inputs
σ	the total number of keyframes
θ	the keyframe ratio in all frames
η	the dynamic load-reduction threshold
A	the scene-change threshold
$\odot$	the excerpt assembly and integration operation
$Aver$	an averaging operation
$Corr$	a correlation computing operation

To divide the input into two streams to process, we define two types of input, video frame f_i and video clip kf_i, where k represents the video clip's length. We treat the first input, video frames, with the same method as image analysis. While dealing with the second input, since a single video clip contains several video frames in time, we extend a time dimension when dealing with video clips.

In a convolutional layer, video frames or video clips change into f_j based on weight matrices, $w_{i,j}$ or $w_{i,j,k}$. However, when the input is video frames, convolutional output can be described as

$$f_j^{i,j} = \sum_{j=1}^{J} \sum_{i=1}^{I} f_i * w_{i,j}. \tag{1}$$

The following equation is an expansion in time dimension to deal with video clips:

$$f_j^{i,j,k} = \sum_{k} \sum_{i=1}^{j} \sum_{i=1}^{I} f_i * w_{i,j,k}. \tag{2}$$

In a pooling layer, the input that comes from a convolutional layer is also organized into feature maps to realize pooling operation, and the number of feature maps is the same as the convolutional layer's, but each is much smaller. By doing so, the resolution of feature maps will be efficiently reduced. This reduction is realized by applying a pooling function in the pooling layer. However, the pooling function can be a simple function, such as averaging. In this article, we use an averaging function and define the pooling function as

$$f_m = \lambda \sum_{n=1}^{p} Aver_{i,(m-1)\times s+n}, \tag{3}$$

where p is the pooling size, s determines the overlap of the pooling windows, $Aver$ represents the averaging operation, and λ is a scaling factor that can be learned.

4. CLASSIFICATION MODULE

In this section, we detail the classification module used to label video frames' information value. We first describe a single frame CNN dealing with video frames, then discuss its extension in time to deal with video clips.

4.1. Video Frames Processing

We give a simple model for solving image classification. A Feed-Forward Neural Network (FNN) is an artificial neural network used to solve basic image classification problems. Like many neural networks, FNN usually consists of three layers: input layer, hidden layer, and output layer. On the basis of the number of hidden layers, the network can be named as a single hidden-layer network or multiple-layer network (MLNs). In the network, a neuron in a certain layer receives information from neurons in previous layers through weighted connections. It computes the weighted sum of its inputs $\sum_j w_j x_j$ and passes this value through an activation function so that its output information moves forward from the input layers, through intermediate layers, and to the output layers.

There are two principal types of activation functions:

Threshold function

$$output = \begin{cases} 0, & if \sum_j w_j x_j \leq threshold \\ 1, & if \sum_j w_j x_j > threshold \end{cases} \tag{4}$$

Sigmoid function

$$\begin{aligned} &\sigma(z) = 1/(1+e^{-z}) \\ &bias = 1 \Big/ \left(1 + exp\left(-\sum_j w_j x_j - b\right)\right). \end{aligned} \tag{5}$$

When the perceptron is used to solve some problems about image classification, the input to the network is raw pixel data. The network correctly classifies the digit by learning weights and biases. However, the learning work is difficult; a small change in the weights or bias of any single perceptron in the network can sometimes cause the output of that perceptron to completely flip. That flip may then cause the behavior of the rest of the network to completely change in a very complicated way. Thus, it is not immediately obvious how we can get a network of perceptron to learn. To overcome this problem, a conventional method is proposed.

We apply a square error cost function to finish the following derivation. We suppose C classes and N samples:

$$E^N = 1/2 \sum_{n=1}^{N} \sum_{d=1}^{c} \left(t_d^n - y_d^n\right)^2, \tag{6}$$

where t_d^n denotes the d dimensions of the n samples and y_d^n denotes the d output of the n samples. If we only have one class, we can get the following expression:

$$E^N = 1/2 \sum_{n=1}^{N} (t^n - y^n)^2 = 1/2 \|t^n - y^n\|^2. \tag{7}$$

However, by reducing the error, we improve the image classification result.

4.2. Video Clips Processing

Having discussed image classification, we return to video-image recognition. Essentially, video is made up of continuous image streams; thus, video clips can be viewed as several single images. In this article, we use a time dimensionality to connect several neighboring frames; thus, the form of output with time dimensionality can be written as

$$f_j^{i,j,k} = \sum_k \sum_{j=1}^{J} \sum_{i=1}^{I} f_i * w_{i,j,k}. \tag{8}$$

Equation (8) is an improved equation compared with Equation (1). As described earlier, researchers generally use two dimensions to represent images; in Equation (8), we add a time dimension k to extend Equation (1) to represent videos. However, we consider Equation (8) to be our 3-dimensional convolution kernel to compute feature maps pixel by pixel.

Next, we introduce our algorithm in detail. The main idea of our algorithm is decomposing the video into many single images, recognizing the importance of each image and dropping the less important or abnormal ones. This method has two advantages: (a) we can realize the purpose of dimension reduction; (b) from these images, we can choose main images to improve accuracy and reduce the runtime of our algorithm.

First, we consider a $P \times Q$ grayscale image; then, we have $P \times Q$ input neurons, with the intensities scaled appropriately between 0 and 1. We define the cost function as

$$J(W, b) = 1/2n\|h_{w,b}(x) - \alpha\|^2. \tag{9}$$

The inputs are 3-dimensionality matrixes. In our neural network, the dimensions of the output need to be determined by the actual condition. The pseudo-code of the classification module is presented in Algorithm 1.

ALGORITHM 1: Algorithm for Training 3-Dimensional Convolution Neural Network

```
Initialize w, b randomly
While cost function < threshold do
  Compute feature maps pixel by pixel according to the 3-dimensional convolution kernel
    f_j^{i,j,k} = Σ_k Σ_{j=1}^{J} Σ_{i=1}^{I} f_i * w_{i,j,k}, and feed forward
  Calculate cost function J(w, b) = 1/2n‖h_{w,b}(x) − α‖^2
  If cost function > threshold then
    Back propagation using method Stochastic Gradient Descent (SGD) method
  End if
End while
```

It is hard for us to consider a problem that the number of images correctly classified is not a smooth function of the weights and biases in the network. If making small changes to the weights and biases in the network, it will be difficult to figure out how to change the weights and biases to get improved performance. Thus, we use the mean squared error (MSE) to make the small changes that can be found easily.

From the cost function, we know that the $J(w, b)$ is nonnegative. In other words, $J(w, b) \geq 0$ and it becomes small, that is, $J(w, b) \approx 0$, when $h_{w,b}(x)$ is approximately equal to the training output, a, for all input x. Thus, we can try to find weights and biases so that we will make $J(w, b) \approx 0$. It is not good when the cost function $J(w, b)$ is large, which means that the $h_{w,b}(x)$ is not equal to the output a for a large number of

inputs x. Thus, the goal of our training algorithm will be to minimize the cost $J(w, b)$ as a function of the weights and biases. In other words, we should try our best to find a set of weights and biases that make the cost as small as possible or make $J(w, b) = 0$.

THEOREM 1. *The cost function J can be guaranteed always to decrease.*

PROOF. According to the basic conception of calculus, we can change J as the form of Equation (9), and our algorithm is related to the vector; thus, we denote the gradient vector by ∇J. The change of cost function is analyzed as follows:

$$\Delta J \approx \frac{\partial J}{\partial w}\Delta w + \frac{\partial J}{\partial b}\Delta b \tag{10}$$

$$\nabla J = \left(\frac{\partial J}{\partial w}, \frac{\partial J}{\partial b}\right)^T \tag{11}$$

$$\Delta J = \nabla J \bullet (\triangle w, \triangle b) \tag{12}$$

$$(\triangle w, \triangle b) = -\eta \nabla J = -\eta \left(\frac{\partial J}{\partial w}, \frac{\partial J}{\partial b}\right)^T \tag{13}$$

$$\Delta J = -\eta \|\nabla J\|^2 = -\eta \left\| \left(\frac{\partial J}{\partial w}, \frac{\partial J}{\partial b}\right)^T \right\|^2 \leq 0. \tag{14}$$

According to Equations (12) and (13), we can find a small, positive parameter called the learning rate that can be used to simplify the expression (11), and then we can guarantee the J always decrease. □

THEOREM 2. *A factor $1/n$ or $1/m$ in cost function helps to train data in real time.*

PROOF. Separating w and b, we get Equations (14) and (15). We set a small number m to randomly chosen training inputs.

$$w'_k = w_k - \varrho \frac{\partial J}{\partial w_k} \tag{15}$$

$$b'_k = b_k - \varrho \frac{\partial J}{\partial b_k} \tag{16}$$

$$\frac{\sum_{j-1}^{m} \nabla J_{xi}}{m} \approx \frac{\sum_x \nabla J_x}{n} = \nabla J \tag{17}$$

$$w'_k = w_k - \frac{\varrho}{m} \frac{\partial J}{\partial w_k} \tag{18}$$

$$b'_k = b_k - \frac{\varrho}{m} \frac{\partial J}{\partial b_k}. \tag{19}$$

In Equation (8), we scale the overall cost function by a factor $1/n$. People sometimes omit the $1/n$, summing over the costs of individual training examples instead of averaging. This is particularly useful when the total number of training examples is not known in advance. This can occur if more training data is being generated in real time, for instance. Also, in a similar way, the mini-batch update rule (17) and (18) sometimes omit the $1/m$ term out the front of the sums. Conceptually, this makes little difference, since it is equivalent to rescaling the learning rate ϱ. □

5. LOAD-REDUCTION MODULE

In this section, we detail the load-reduction module used to abandon less important frames. We first introduce this module's input, then describe the keyframe extraction mechanism.

5.1. Keyframes and Video Skim

After the classification module labels every video frame's and clip's information value, we consider these information values as load-reduction module input. In literature, keyframe and video skim are two basic video abstract mechanisms [Truong and Venkatesh 2007]. They are used to help to recognize the importance of each frame. In this article, we use the concept of the two video abstract mechanisms; instead of using original video as input, we change the input to the information value output from the classification module. We introduce keyframe and video skim in our article, as follows. When dealing with video frames, we use keyframes, which can also be called representative frames (R-frames). R-frames are a set consisting of a collection of salient images extracted from the information values. The keyframe set K is defined as follows:

$$K = F_{k-frames}(Video) = f_{r_1}, f_{r_2}, \ldots, f_{r_\sigma}, \tag{20}$$

where σ is the total number of keyframes and $F_{k-frames}$ denotes the concrete extraction operation. In Equation (20), we use extractive label f_{r_i} to form a keyframe set to represent the input video.

When dealing with video clips, we use video skim to help recognize the importance of each frame. Video skim consists of a collection of video clips extracted from the original video. These video clips are of significantly shorter duration. The video skim S is defined as follows:

$$S = F_{skim}(Video) = f_{r_1} \bigodot f_{r_2} \bigodot \cdots \bigodot f_{r_\sigma}, \tag{21}$$

where $\bigodot$ is the excerpt assembly and integration operation.

5.2. Keyframe Ratio

In an important feature-extraction mechanism, we set a ratio, θ, over the number of the video frames or video clips as a constraint to guarantee a suitable output number in order to reduce overload. This method is suitable for a resource shortage environment such as mobile Internet.

We now address the problem of how to set the ratio. Actually, the problem can be viewed as an optimization problem of finding a suitable set $R = r_1, r_2 \ldots, r_\sigma$, which can represent the video using the least frames or clips. We summarize the optimization problem as

$$r_1, r_2, \ldots, r_\sigma = arg\ min_{r_i}\{D(R, F)|1 \leq r_i \leq n\} \tag{22}$$

$$\sigma = \theta \cdot n, \tag{23}$$

where n is the number of frames or clips in the original video sequence, σ is the total number of keyframes or clips, D is a dissimilarity measure, and F is the output of the classification module.

5.3. Minimum Correlation Among Keyframes or Video Clips

Next, we analyze how to find a suitable set that can represent the video using the least frames or clips. Because the input, frames or clips, are always sequential elements, we consider making use of the correlation among them to finish key element extraction.

Therefore, we introduce minimal correlation to solve the problem. The minimal correlation method is select frames or clips that are dissimilar to each other and can represent the video with the least elements. Introducing the concept of minimal correlation, we can rewrite Equation (21) as

$$r_1, r_2, \ldots, r_\sigma = arg\ min_{r_i}\{Corr(f_{r_1}, f_{r_2}, \ldots, f_{r_\sigma})\}, \tag{24}$$

where $Corr$ is a correlation computing operation.

When we consider pairs of frames or clips to simplify the entire set, Equation (22) can be formulated as

$$Corr(f_{r_1}, f_{r_2}, \ldots, f_{r_\sigma}) = \left\{ \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} corr(f_{r_i}, f_{r_j})^2 \right\}^{1/2}, \tag{25}$$

where $Corr(f_{r_i}, f_{r_j})$ is the correlation coefficient of any two frames or clips (f_{r_i}, f_{r_j}).

In this article, we take sequential elements into consideration, and Equation (24) can be written as

$$\{r_1, r_2, \ldots, r_\sigma\} = arg\ min_{r_i} \left\{ \sum_{i=1}^{\sigma-1} corr(f_{r_i}, f_{r_{i+1}}) \right\}. \tag{26}$$

However, the extraction of keyframes or clips based on these equations endeavor to maximize the difference of each frame or clip rather than simply reduce the total number.

ALGORITHM 2: Load-Reduction Module, Scene Change

```
1 Inputs f_{r_i}, A
2 Procedure:
3 Begin
4   While (i < n)
5     If (Corr(f_{r_i}) < A)
6       enter into scene+1
7     Else
8       still in scene
9     End if
10  End while
```

In the model's load-reduction module, a reliable keyframe extraction mechanism is applied to recognize the importance of each frame or clip. The specific implementation process of the mechanism is presented in Algorithms 2 and 3. Algorithm 2 mainly judges whether the scene changes or not and Algorithm 3 emphasizes the load reduction. In the pseudo-codes, what we first recognize is the scene change, because different scenes have different thresholds. Through correctly recognizing scene changes or not, we can improve the final performance. What's more, if continuous frames are all in the same scene, we classify these frames into a group and distribute each group a threshold η.

6. PERFORMANCE EVALUATIONS

In this section, the performance of the hybrid-stream model is verified. In the first part, we introduce the dataset used in this model. Then, the performance of hybrid-stream model is analyzed and compared to the existing related models. Finally, performance

ALGORITHM 3: Load-Reduction Module, Keyframe Extraction

```
1 Input f_ri
2 Procedure:
3  Begin
4     η = α ∑ f/k
5     While (scene has not changed)
6       If (f_ri > η)
7         drop the frame
8       Else
9         store the frame in set S
10       End if
11    End while
```

comparisons with Basic CNN, Temporal stream ConvNet, and the Two-stream model are demonstrated [Xia et al. 2016; Ye et al. 2015].

6.1. Dataset in the Simulation

Our works aim to evaluate the hybrid-stream model in different datasets, including UCF-101, UCF-101 Expand, CCV, and HMDB-51, and prove the superiority and adaptability of this model compared with other models, such as the basic CNN model, Temporal stream ConvNet model, and Two-stream model. We briefly introduce the datasets, as follows.

UCF-101 is an action-recognition dataset [Soomro et al. 2012] which consists of 101 action categories. With 13320 videos from these action categories, UCF-101 presents the largest diversity in terms of actions.

UCF-101 Expand is an extension of UCF-101. We perform some image transformation such as fuzzification and distortion, then add the transformed videos to the original dataset. The expanded dataset aims to prevent overfitting.

CCV is a new database [Jiang et al. 2011] containing 9317 YouTube videos over 20 categories. The database mainly emphasizes people's interests and originality of video content. These videos have little textual annotation and can also apply for automatic content analysis techniques.

HMDB-51 is a large human motion database [Kuehne et al. 2011]. The dataset contains 6849 clips, which can be divided into 51 action categories; each category contains at least 101 clips. The videos are collected from various sources: most of them are from movies and the rest are from public databases, such as YouTube and Google videos.

Among the four datasets, the main differences between them are that these videos' sources are different and each dataset has different action categories. These differences can present the largest diversity of processed videos.

6.2. Simulation Setup

There are two key parameters in our model: σ and η. Specially, σ is a dynamic parameter that changes with the length of the video:

$$\sigma = \frac{Time_{video}}{Time_{sampling}},$$

where $Time_{video}$ is the duration of the input video, and $Time_{sampling}$ is the sampling time. The bigger the $Time_{sampling}$, the more video frames or clips we will obtain. In addition,

η is also a dynamic parameter that changes according to correlation among frames:

$$\eta = \frac{\sum Importance_{frame}}{number},$$

where $Importance_{frame}$ is processed frame's importance, which ranges from 0 to 1, and *number* refers to several correlative frames' number.

In simulations, we adjust parameter σ to control the size of the processed video and adjust parameter η to decide the dynamic threshold. The size of video clips is associated with parameter σ; the greater parameter σ is, the more video frames are contained in video clips. However, only when choosing a suitable parameter σ can the result's accuracy can be improved. In addition, parameter η is key to ensuring effective load reduction and better quality of experience (QoE).

The inputs are fixed to the size of 214×214. In contrast to a two-dimensional feature map in the conventional layer, we add a time dimension based on the two-dimensional feature map to process real-time videos. Therefore, we have two different inputs. The first input is video frames (single images); the second input is video clips (several continuous frames).

In the following simulation, we change several parameters in the classification algorithm to compare the algorithm's classification precision in different architectures. We also compare the model's performance in different datasets. Finally, we compare our model with other existing models. In this section, we emphasize classified precision comparison.

6.3. Simulation Results

In this section, we first compare our model in different architectures. We use UCF-101 as the dataset of our classified precision comparison. In Table II, we set three different groups of parameters in architectures A, B, and C to compare our model's accuracy in UCF-101 1 to 4 and 4 to 8 classifications. These parameters include the size of convolution kernel (Filter), the number of feature map (Nf), channels, pooling size, and input and output dimension (In and Out). Each architecture has four layers (Figure 1): Convolutional layer and Pooling layer (here, we combine them as Conv-Pooling layer), Fully connected layer, and Feature Extraction layer. In the table, it shows that the more Conv-Pooling Layers the architecture has, the higher accuracy the result acquires. In addition, the size of convolution kernel (Filter) also affects the accuracy of results: a relatively bigger size of convolution kernel (Filter) acquires better results on accuracy at the cost of time consumed. In general, when suitable parameters are selected, our classification algorithm has a good accuracy: around 90%. As a result, higher classification precision is helpful to the following load-reduction operation.

Next, we analyze the model's performance in different datasets. UCF-101, UCF-101 Expand, HMDB-51, and CCV are four different datasets that we use to make comparisons. As Figure 2 shows, our method has a good performance on the whole, and is basically superior to other models on the same dataset.

Then, we compare our model with other existing models or algorithms, and show our model's performance.

Receiver operating characteristic (ROC) is a kind of image used to describe sensitivity. In the simulation, based on the datasets UCF-101 and UCF-101 Expand, we use true-positive rate (TPR) and false-positive rate (FPR) of ROC to compare our method with the basic CNN model and Support Vector Machine (SVM).

Figure 3 shows the classification performance of our methods, Basic CNN and SVM. The various parameter settings can be seen in architecture A in Table II. As can be seen, for the same dataset, our method has a better approach to the coordinate (0, 1), which is usually called as a perfect classifier. Meanwhile, the approaching speed of our

Table II. Different Architectures for Classification

Architecture	Layer	Param	Value	Accuracy 1 to 4	Accuracy 4 to 8
A	Conv-Pooling Layer 1	Filter	5*7*8	91.30%	88.60%
		Nf	20		
		Channels	3		
		Pooling size	2*2		
	Conv-Pooling Layer 2	Filter	5*8*5		
		Nf	40		
		Channels	20		
		Pooling size	2*2		
	Fully connected Layer	In	2400		
		Out	50		
	Feature Extraction Layer	In	50		
		Out	4		
B	Conv-Pooling Layer 1	Filter	5*7*4	92.80%	90.10%
		Nf	20		
		Channels	3		
		Pooling size	1*1		
	Conv-Pooling Layer 2	Filter	5*7*4		
		Nf	20		
		Channels	20		
		Pooling size	2*2		
	Conv-Pooling Layer 3	Filter	5*7*4		
		Nf	40		
		Channels	20		
		Pooling size	2*2		
	Fully connected Layer	In	6160		
		Out	50		
	Feature Extraction Layer	In	50		
		Out	4		
C	Conv-Pooling Layer 1	Filter	5*7*5	92.60%	90.30%
		Nf	20		
		Channels	3		
		Pooling size	1*1		
	Conv-Pooling Layer 2	Filter	5*7*5		
		Nf	20		
		Channels	20		
		Pooling size	2*2		
	Conv-Pooling Layer 3	Filter	5*7*5		
		Nf	40		
		Channels	20		
		Pooling size	2*2		
	Fully connected Layer	In	6160		
		Out	50		
	Feature Extraction Layer	In	50		
		Out	4		

method is obviously faster than that of basic CNNs and SVMs. However, our method achieves an improvement in classification accuracy.

The performance measures that we used are precision, recall, and scaled Area Under the Curve (AUC) at different values of FPRs [Ji et al. 2013]. The average performance is shown in Figure 4. Under lower FPRs, we can see from Table III a higher average performance on Precision and lower average performance on Recall and AUC. When under the same FPR, our method outperforms the 2D-CNN, and has a similar performance

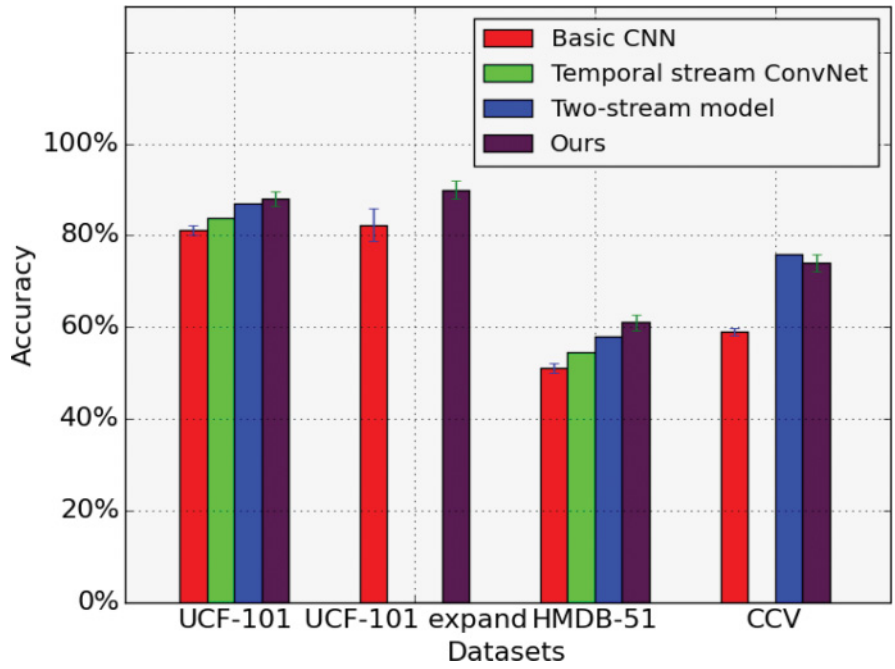

Fig. 2. Comparisons among different models under different datasets.

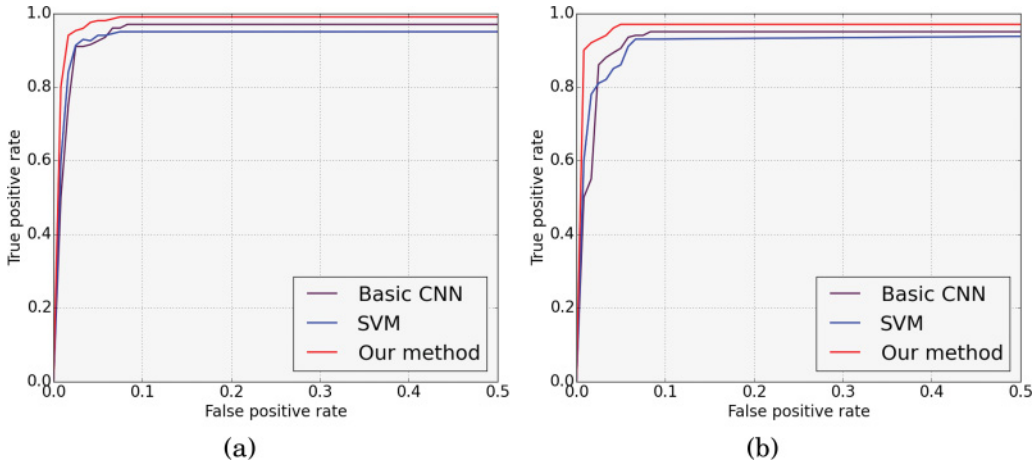

Fig. 3. Comparisons among basic CNN, SVM, and our method under (a) UCF-101 and (b) UCF-101 Expand.

with the 3D RCNN and 3D CNN. The specific data information is presented in Table III.

The performance of the load-reduction module is shown in Figure 5. The figures are both a 10s video clip cut from two different videos. The main difference between them is that the first video has an obvious scene change and the other only has a single scene. In Figure 5, we use two colors to distinguish two scenes. Since we use different dynamic threshold η to represent different scenes, it is important to first recognize

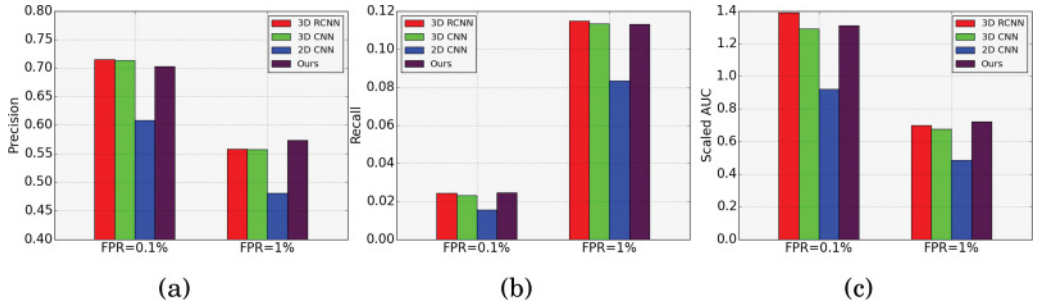

Fig. 4. Average performance comparison of the four methods under different FPRs.

Table III. Detailed Comparisons of Our Method and Three Other Models under Different FPRs

Method	FPR	Measure	CellToEar	ObjectPut	Pointing	Average
$3D-RCNN^{S}_{332}$	0.1%	Precision	0.5717	0.7348	0.8380	0.7148
		Recall	0.0211	0.0348	0.0169	0.0243
		$AUC(*10^3)$	0.0114	0.0209	0.0096	0.0139
	1%	Precision	0.3917	0.5384	0.7450	0.5584
		Recall	0.1019	0.1466	0.0956	0.1147
		$AUC(*10^3)$	0.6272	0.9044	0.5665	0.6993
$3D-CNN^{S}_{332}$	0.1%	Precision	0.6433	0.6748	0.8230	0.7137
		Recall	0.0282	0.0256	0.0152	0.0230
		$AUC(*10^3)$	0.0173	0.0139	0.0075	0.0129
	1%	Precision	0.4091	0.5154	0.7470	0.5572
		Recall	0.1109	0.1356	0.0931	0.1132
		$AUC(*10^3)$	0.6759	0.7916	0.5581	0.6752
$2D-CNN$	0.1%	Precision	0.3842	0.5865	0.8547	0.6085
		Recall	0.0097	0.0176	0.0192	0.0155
		$AUC(*10^3)$	0.0057	0.0109	0.0110	0.0092
	1%	Precision	0.3032	0.3937	0.7446	0.4805
		Recall	0.0505	0.0974	0.1020	0.0823
		$AUC(*10^3)$	0.2725	0.5589	0.6218	0.4844
ours	0.1%	Precision	0.5817	0.6944	0.8333	0.7031
		Recall	0.0266	0.0288	0.0181	0.0245
		$AUC(*10^3)$	0.0151	0.0157	0.0086	0.0131
	1%	Precision	0.3866	0.5437	0.7896	0.5733
		Recall	0.1100	0.1372	0.0920	0.1131
		$AUC(*10^3)$	0.6681	0.9189	0.5734	0.7201

Notes: The superscript s represents that the five channels are convolved separately; the subscript 332 represents that the first two convolutional layers use 3D convolution and the last convolutional layer uses 2D convolution.

scene changes with threshold η. From the two figures, we can visually observe the drop frames.

Finally, we summarize the average performance of our model in different types of video. We aim to reduce load on video frames or clips' correlation. Some videos, such as talk shows, have a high correlation among frames and clips; these videos have a smaller load reduction. On the other hand, some videos, such as TV series, have relative low correlation among frames and clips; these videos have a larger load reduction. In total, the load reduction of different types of video are shown in Table IV. Different types of video have a quite different load reduction under our model; the average load-reduction percentage is about 2.76.

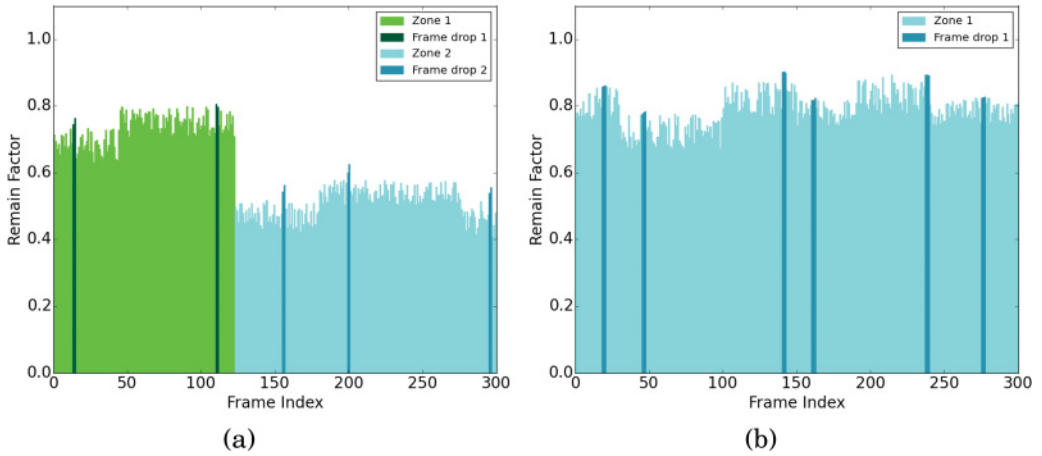

Fig. 5. (a) Two scenes in a 10s video and its drop frames. (b) One scene in a 10s video and its drop frames.

Table IV. Load Reduction in Different Types of Video

Content	Size reduction
Cinecolor	1.9%
Black-white	2.6%
Talk show	7.9%
TV series	2.5%
Music video	1.1%
Landscape	2.4%
Technique	2.0%
True man show	0.8%
Situation comedy	3.6%
Average	2.76%

7. CONCLUSIONS

We presented a novel model for enhancing classification precision and reducing network overload. The model, called the hybrid-stream model, is able to recognize the important frames and decide whether to drop the unimportant ones. In contrast to conventional methods, such as deep learning to address the image analysis problem, we improve the method to deal with video analysis. We formalize the overload problem of videos as an optimization problem and show a practical algorithm over a large amount of real-time data. The conducted simulations show that our model performs well in most of the datasets, in particular for UCF-101 and UCF-101 Expand. The proposed hybrid-stream model and the improved video-recognition algorithm can lead to fairly good video streams in mobile Internet. The model can reduce network overload and will not lower users' QoE.

REFERENCES

Y. Zheng, B. Jeon, D. Xu, Q. Wu, and H. Zhang. 2015. Image segmentation by generalized hierarchical fuzzy C-means algorithm. *Journal of Intelligent and Fuzzy Systems* 28, 2, 961–973.

M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, F. Matus, R. Pan, N. Yadav, and G. Varghese. 2014. CONGA: Distributed congestion-aware load balancing for datacenters. In *Proceedings of the 2014 ACM Conference on SIGCOMM (SIGCOMM'14)*. ACM, New York, NY, 503–514.

M. Al-fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. 2010. Hedera: Dynamic flow scheduling for data center networks. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation (NSDI'10)*. 19–19.

M. Baktashmotlagh, M. Harandi, B. C. Lovell, and M. Salzmann. 2014. Discriminative non-linear stationary subspace analysis for video classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36, 12, 2353–2366.

B. Chen, H. Shu, G. Coatrieux, G. Chen, X. Sun, and J. Coatrieux. 2015. Color image analysis by quaternion-type moments. *Journal of Mathematical Imaging and Vision* 51, 1, 124–144.

D. Chiang, C. Wang, C. Chen, and W. Lo. 2015. Scheduling management for multiple real-time data over on-demand mobile environments. In *Proceedings of IEEE International Conference on Mobile Services (MS'15)*. IEEE, New York, NY, 383–390.

M. Dong, T. Kimata, K. Sugiura, and K. Zettsu. 2014. Quality-of-experience (QoE) in emerging mobile social networks. *IEICE Transactions on Information and Systems* 97, 10, 2606–2612.

M. Dong, H. Li, K. Ota, and J. Xiao. 2015. Rule caching in SDN-enabled mobile access networks. *IEEE Network* 29, 4, 40–45.

M. Dong, X. Liu, Z. Qian, A. Liu, and T. Wang. 2015. QoE-ensured price competition model for emerging mobile networks. *IEEE Wireless Communications* 22, 4, 50–57.

Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang. 2015. Enabling personalized search over encrypted outsourced data with efficiency improvement. *IEEE Transactions on Parallel and Distributed Systems*. DOI:10.1109/TPDS.2015.2506573

B. Gu and V. Sheng. 2016. Structural minimax probability machine. *IEEE Transactions on Neural Networks and Learning Systems*. DOI:10.1109/TNNLS.2016.2544779

G. Han, W. Que, G. Jia, and L. Shu. 2016. An efficient virtual machine consolidation scheme for multimedia cloud computing. *Sensors* 16, 2, Article 246.

A. Hava, Y. Ghamri-Doudane, M. Muntean, and J. Murphy. 2015. Increasing user perceived quality by selective load balancing of video traffic in wireless networks. *IEEE Transactions on Broadcasting* 61, 2, 238–250.

K. He, X. Zhang, S. Ren, and J. Sun. 2015. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37, 9, 1904–1916.

K. He, E. Rozner, K. Agarwal, W. Felter, J. Carter, and A. Akella. 2015. Presto: Edge-based load balancing for fast datacenter networks. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM'15)*. ACM, New York, NY, 465–478.

P. Hernandez. 2010. Statistical analysis of network traffic for anomaly detection and quality of service provisioning. Ph.D. dissertation, Computer Science, Telecom Bretagne (ENST Bretagne), Brest, France.

O. H. Hussein, T. N. Saadawi, and M. Jong Lee. 2005. Probability routing algorithm for mobile ad hoc networks' resources management. *IEEE Journal on Selected Areas in Communications* 23, 12, 2248–2259.

S. Ji, W. Xu, M. Yang, and K. Yu. 2013. 3D convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 1, 221–231.

Y. Jiang, G. Ye, S. Chang, D. Ellis, and A. C. Loui. 2011. Consumer video understanding: A benchmark database and an evaluation of human and machine performance. In *Proceedings of ACM International Conference on Multimedia Retrieval (ICMR'11)*. ACM, Italy.

A. Karpathy, G. Toderici, S. Shetty, and T. Leung. 2014. Large-scale video classification with convolutional neural networks. In *Proceedings of 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'14)*. IEEE, Columbus, 1725–1732.

R. Kroon. 2002. Dynamic vehicle routing using ant based control. Masters Thesis, Delft University of Technology, Delft, The Netherlands.

H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. 2011. HMDB: A large video database for human motion recognition. In *Proceedings of IEEE International Conference on Computer Vision (ICCV'11)*. IEEE, Barcelona, 2556–2563.

A. Lakhina, M. Crovella, and C. Diot. 2005. Mining anomalies using traffic feature distributions. In *Proceedings of the 2005 Conference on Applications, Technologies, and Protocols for Computer Communications (SIGCOMM'05)*. ACM, New York, NY, 217–228.

S. K. Lee, S. Yoo, J. Jung, H. Kim, and J. Ryoo. 2015. Link-aware reconfigurable point-to-point video streaming for mobile devices. *ACM Transactions on Multimedia Computing Communications and Applications* 12, 1, Article 9.

J. Li, X. Li, B. Yang, and X. Sun. 2015. Segmentation-based image copy-move forgery detection scheme. *IEEE Transactions on Information Forensics and Security* 10, 3, 507–518.

B. T. Morris and M. M. Trivedi. 2008. Learning, modeling, and classification of vehicle track pattern from live video. *IEEE Transactions on Intelligent Transportation System* 9, 3, 425–437.

I. Mrazova and M. Kukacka. 2008. Hybrid convolutional neural networks. In *Proceedings of the 6th IEEE International Conference on Industrial Informatics* IEEE, Indin, 469–474.

Z. Pan, Y. Zhang, and S. Kwong. 2015. Efficient motion and disparity estimation optimization for low complexity multiview video coding. *IEEE Transactions on Broadcasting* 61, 2, 166–176.

T. Qin, X. Guan, W. Li, and P. Wang. 2009. Monitoring abnormal traffic flows based on independent component analysis. In *Proceedings of IEEE International Conference on Communications (ICC'09).* IEEE, Dresden, 1–5.

T. Qin, X. Guan, and Q. Huang. 2010. Characteristic measurement of the connection degree for network monitoring. In *Proceedings of the 8th World Congress on Intelligent Control and Automation (WCICA'10).* IEEE, Jinan, 147–151.

F. Rango and M. Tropea. 2009. Swarm intelligence based energy saving and load balancing in wireless ad hoc networks. In *Proceedings of the 2009 Workshop on Bio-inspired Algorithms for Distributed Systems (BADS'09).*ACM, New York, NY, 77–84.

K. Simonyan and A. Zisserman. 2014. Two-stream convolutional networks for action recognition in videos. In *Proceedings of the Conference on Neural Information Processing System (NIPS'14).* 568–576.

R. Song and F. Liu. 2014. Real-time anomaly traffic monitoring based on dynamic k-NN cumulative-distance abnormal detection algorithm. In *Proceedings of the 3rd International Conference on Cloud Computing and Intelligence System (CCIS'14).* IEEE, Shenzhen, 187–192.

K. Soomro, A. R. Zamir, and M. Shah. 2012. UCF101: A dataset of 101 human action classes from videos in the wild. CRCV-TR-12-01.

B. Truong and S. Venkatesh. 2007. Video abstraction: A systematic review and classification. *ACM Transactions on Multimedia Computing, Communications, and Applications* 3, 3, Article 3.

K. Wang and Y. Yu. 2013. A query-matching mechanism over out-of-order event stream in IoT. *International Journal of Ad Hoc and Ubiquitous Computing* 13, 3/4, 197–208.

K. Wang, H. Lu, L. Shu, and J. Rodrigues. 2014. A context-aware system architecture for leak point detection in the large-scale petrochemical industry. *IEEE Communications Magazine* 52, 6, 62–69.

K. Wang, H. Gao, X. Xu, J. Jiang, and D. Yue. 2015. An energy-efficient reliable data transmission scheme for complex environmental monitoring in underwater acoustic sensor networks. *IEEE Sensors Journal* 16, 11, 4051–4062. DOI:10.1109/jsen.2015.2428712.

H. Wang, M. Chan, and T. Ooi. 2015. Wireless multicast for zoomable video streaming. *ACM Transactions on Multimedia Computing, Communications, and Applications* 12, 1, Article 5.

K. Wang, Y. Shao, L. Shu, G. Han, and C. Zhu. 2015. LDPA: A local data processing architecture in ambient assisted living communications. *IEEE Communications Magazine* 53, 1, 56–63.

K. Wang, Y. Shao, L. Shu, Y. Zhang, and C. Zhu. 2016. Mobile big data fault-tolerant processing for eHealth networks. *IEEE Network* 30, 1, 36–42.

K. Wang, L. Zhuo, Y. Shao, D. Yue, and K. F. Tsang. 2016. Towards distributed data processing on intelligent leakpoints prediction in petrochemical industries. *IEEE Transactions on Industrial Informatics* PP, 99, 1. DOI:10.1109/TII.2016.2537788, 2016.

F. Xia, L. T. Yang, L. Wang, and A. Vinel. 2012. Internet of Things. *International Journal of Communication Systems* 25, 9, 101–1102.

Z. Xia, X. Wang, X. Sun, Q. Liu, and N. Xiong. 2016. Steganalysis of LSB matching using differences between nonadjacent pixels. *Multimedia Tools and Applications* 75, 4, 1947–1962.

H. Ye, Z. Wu, and R. Zhao. 2015. Evaluating two-stream CNN for video classification. In *Proceedings of the 5th ACM International Conference on Multimedia Retrieval (ICMR'15).* ACM, New York, NY, 435–442.

K. Zhao, W. Rao, Y. Zhang, P. Hui, and S. Tarkoma. 2015. Towards maximizing timely content delivery in delay tolerant networks. *IEEE Transactions on Mobile Computing* 14, 4, 755–769.

K. Zeb, B. AsSadhan, J. Al-Muhtadi, and S. Alshebeili. 2014. Volume based anomaly detection using LRD analysis of decomposed network traffic. In *Proceedings of the 4th International Conference on Innovative Computing Technology (INTECH'14).* IEEE, Luton, 52–57.

Y. Zhang, R. Yu, W. Yao, S. Xie, Y. Xiao, and M. Guizani. 2011. Home m2m networks: Architectures, standards, and QoS improvement. *IEEE Communications Magazine* 49, 4, 44–52.

Received December 2015; revised January 2016; accepted February 2016