symmetry

MDPI

Article

SIAT: A Distributed Video Analytics Framework for Intelligent Video Surveillance

Md Azher Uddin, Aftab Alam, Nguyen Anh Tu, Md Siyamul Islam and Young-Koo Lee *

Department of Computer Science and Engineering, Kyung Hee University Global Campus, Yongin 17104, Korea
* Correspondence: yklee@khu.ac.kr

Received: 18 June 2019; Accepted: 10 July 2019; Published: 12 July 2019

Abstract: In recent years, the amount of intelligent CCTV cameras installed in public places for surveillance has increased enormously and as a result, a large amount of video data is produced every moment. Due to this situation, there is an increasing request for the distributed processing of large-scale video data. In an intelligent video analytics platform, a submitted unstructured video undergoes through several multidisciplinary algorithms with the aim of extracting insights and making them searchable and understandable for both human and machine. Video analytics have applications ranging from surveillance to video content management. In this context, various industrial and scholarly solutions exist. However, most of the existing solutions rely on a traditional client/server framework to perform face and object recognition while lacking the support for more complex application scenarios. Furthermore, these frameworks are rarely handled in a scalable manner using distributed computing. Besides, existing works do not provide any support for low-level distributed video processing APIs (Application Programming Interfaces). They also failed to address a complete service-oriented ecosystem to meet the growing demands of consumers, researchers and developers. In order to overcome these issues, in this paper, we propose a distributed video analytics framework for intelligent video surveillance known as SIAT. The proposed framework is able to process both the real-time video streams and batch video analytics. Each real-time stream also corresponds to batch processing data. Hence, this work correlates with the symmetry concept. Furthermore, we introduce a distributed video processing library on top of Spark. SIAT exploits state-of-the-art distributed computing technologies with the aim to ensure scalability, effectiveness and fault-tolerance. Lastly, we implant and evaluate our proposed framework with the goal to authenticate our claims.

Keywords: intelligent video surveillance; distributed video processing; Apache Spark

1. Introduction

Recently, the volume of video data has increased dramatically on the internet and various sources are actively contributing toward video generation. For example, YouTube users upload more than 300 h of videos per minute [1], almost 58% of downstream traffic on the internet in video [2], and IntelliVision deployed more than four million cameras worldwide for surveillance [3]. These unstructured video data are reservoirs of knowledge and have a direct relation to the real world events, unlike other data sources. Video data has the ability to provide us with information about people's interactions and behaviors. Furthermore, real-time video streams can help in behavior analysis whether it is of traffic or human patterns. The security agencies used to process hours of surveillance video collected after some sort of unwanted event occurs to determine moments where the suspects passed the cameras. Similarly, search technology scans the meta-data saved with videos, like tags assigned during video uploads to YouTube ('science', 'kids', 'cat dancing' or other keywords). Some domains exploit computer vision techniques for video classification to detect objects, cars, or suspicious behavior in videos.

However, video content has largely been overlooked in the discussion about big data. Video data can be turned into useful big data and true video analytics are required while exploiting the concepts of computer vision.

With the current abilities of computer vision technology, we have the expertise to mine this visual data to acquire valuable insights about what is happening in the world. Video analytics has stakes from surveillance to space. The demand for intelligent video analytics is expected to be driven by many factors, such as low costs, flexibility, agility and security. According to Reference [4] intelligent video analytics market is expected to grow from USD 3.23 Billion in 2018 to USD 8.55 Billion by 2023 (a nearly 21 percent growth rate). However, video processing applications require more time and resources for processing. Extracting the insights from the large-scale video analytics system is a hectic task. Substantial amounts of video data pose a big challenge for video management and manipulation, which asks for the powerful computer to process and mine these video data. Moreover, due to the large-scale video data, an elastic solution to save and process video data is needed for potential decision making. However, most of the existing works [5,6] rely on a traditional client/server framework to perform simple tasks (e.g., face and object recognition) [7] while lacks the support for more complex application scenarios (e.g., activity recognition). Furthermore, these frameworks are rarely handled in a scalable manner using distributed computing. Even though intelligent video surveillance has been subject to tremendous advancement, there is still a lack of contributions from the domain of system engineering to the field [6]. Besides, these works only focus the consumer's (i.e., end-user) point of view, while completed overlooked the third party developers.

In order to resolve the above-mentioned issues, in this paper, we propose a layered framework for scalable video analytics in a distributed environment named SIAT. SIAT exploits state-of-the-art distributed technologies and is composed of five layers, that is, a Big Data Curation Layer (BDCL), Distributed Video Data Processing Layer (DVDPL), Distributed Video Data Mining Layer (DVDML), Knowledge Curation Layer (KCL) and Service Curation Layer (SCL). The Big Data Curation Layer (BDCL) is the base layer and is responsible for the large scale video stream, batch video, user profiles and other SIAT's metadata and verity management. BDCL provide role-based data management restful services to the above layers. Distributed Video Data Processing Layer (DVDPL) is in charge of pre-processing and extracting the important features from the raw videos which are provided as input to the Distributed Video Data Mining Layer (DVDML). The DVDM Layer is responsible for producing the high-level semantic result from the features generated by the DVDPL. Knowledge Curation Layer (KCL) generates ontology and creates knowledge based on the extracted higher level features which are obtained from DVDML. Finally, Service Curation Layer (SCL) provides services to the end users. The proposed framework is able to process both the real-time video streams and batch video analytics. For the offline video processing, in the beginning, all the batch video data are stored in the Hadoop Distributed File System (HDFS) [8] and processed on the top of Spark [9,10]. In contrast, for online video processing, we adopted Apache Kafka [11] and Spark Streaming [12]. Apache Kafka is responsible for capturing video frames from multiple sources while Spark streaming is responsible for the distributed computing of video frames. SIAT also enables the third party developers to develop customized functions and video analytics services by exploiting provided distributed video processing APIs and other machine learning libraries.

Apache Spark [9] is an on-demand distributed computing platform for large-scale data analysis. However, Spark provides almost no support for complex data-types such as video. Furthermore, Spark does not support any high-level APIs for image or video processing. To resolve this issue, MMLSpark [13] introduced a novel open source library that integrates the popular image processing library OpenCV with Spark. However, their work does not support any video processing APIs. Besides, in References [14,15], the authors introduced a Streaming Video Engine (SVE) for uploading and processing videos in a distributed manner and a framework for real-time video analysis on Spark, respectively. Despite that, existing literature lacks support for the dynamic feature extraction on a distributed environment, which is a very essential component to provide any high-level

services [10,16]. To address this issue, we introduce a distributed video processing library that provides video processing on top of Spark. Our work integrates the image and video processing library JavaCV with Spark so that we can stay in the same ecosystem as much as possible. Our work not only limited by providing basic distributed video processing APIs, but it also provides distributed dynamic feature extraction APIs which extracts the prominent information the video data. Due to the provided distributed video processing APIs, it increases the usability of the proposed framework since developers do not need to pay much attention to behind distributed video processing algorithms, which should hide the underlying technical complexities. The main contributions of this paper are:

- We propose and develop a novel layered framework for intelligent video surveillance which has the ability to process both real-time streaming videos and offline batch processing in a scalable manner. The SIAT framework can be used as the reference architecture for distributed video analytics.
- We introduce a distributed video processing library that provides video processing on top of Spark. Our work is not only limited to providing basic distributed video processing APIs but it also provides distributed dynamic feature extraction APIs which extracts prominent information from the video data.
- We also develop application scenarios for both online and offline distributed video data processing services.
- Extensive experiments are performed to ensure scalability, fault-tolerance and effectiveness.

The remainder of the paper is planned as follows: Related work is discussed in Section 2, while Section 3 thoroughly explains the proposed SIAT Framework; Section 4 presents the evaluations of our framework; in the end, conclusions and future directions are shown in Section 5.

2. Related Work

The number of applications and systems for intelligent video surveillance has briskly developed during recent years. However, only a few works have concentrated on the distributed environment. In this section, we explore the intelligent video surveillance frameworks for both academic research areas and the industrial arena.

In academic research, Zhang et al. [7] introduced a cloud-based architecture that can provide both real-time processing and offline batch data analysis of large-scale videos. This work explored Apache Kafka and Storm for real-time processing, while Hadoop based MapReduce framework is used for batch video data processing. However, this work only focuses on video processing, while video mining is not explored. Hu et al. [17] presented an evaluation platform for Intelligent Video and Image Analysis. The framework comprises four components: a set of evaluation tools, evaluation criteria, evaluation data sets and an evaluation operation system. A scalable Smart Surveillance Framework is introduced in Reference [6], which allows researchers to execute their solutions to the surveillance field. However, their work does not support distributed computing. Chao and Jun [18] introduced a distributed real-time video surveillance framework, which is employed to support the analysis of numerous surveillance data. In Reference [19], distributed real-time video processing is developed for objects and event detection but this work is limited to fewer applications. Hossain [5] presented a cloud-based client-server framework for the surveillance system that includes segmentation, motion detection, tracking activity and recognition. Zhang et al. [20] presented an online video surveillance framework that includes the distributed Kafka message queue and Spark Streaming; however, they overlooked the offline video processing. Recently, BigDL [21] a distributed deep learning framework is introduced, which is implemented on top of Apache Spark and allows users to develop deep learning applications. In this literature, they showed efficiency with object detection and image feature extraction. Furthermore, BigDL supports immensely efficient and scalable distributed training. However, they do not provide support for video processing, since the existing BigDL framework only brings out spatial information and does not consider dynamic information,

which is an important feature for processing the videos. MMLSpark [13] provides a distributed image processing library that integrates OpenCV with Spark and combines deep learning library Cognitive Toolkit, with Apache Spark. However, their work is also limited to image processing and does not provide any video processing APIs. Lv et al. [22] introduced a Spark based solution for near-duplicate video detection while, in Reference [23], the authors proposed a distributed solution for face search and classification. In contrast, Huang et al. [24] developed convolutional neural networks based method for recognizing objects in traffic video data. This information is stored and processed applying Spark. However, the raw video data are not processed using Spark. Yaseen et al. [25] introduced object classification using the Hadoop MapReduce framework. Finally, in Reference [26] the authors proposed both Hadoop and Spark based solutions for edge detection using canny operator [27] and line detection using Hough transform. Despite these, most of the existing literature lacks the support for distributed dynamic feature extraction APIs, which are a very necessary part to produce higher-level applications. While our work introduces a distributed video processing library, which is not only limited by providing basic distributed video processing APIs (e.g., edge detection, background subtraction, keyframe extractor) but it also provides distributed dynamic feature extraction APIs (e.g., dynamic texture feature extractor), which extracts the prominent information the video data. Due to the provided distributed video processing APIs, it extends the usability of the proposed framework since the developers do not need to spend much thought behind distributed video processing algorithms. In contrast, numerous organizations have already deployed the cloud-based distributed video processing system. Eagle Eye Networks [28] provides cloud-based video surveillance system for real-time streaming. They also provide RESTful APIs for recording, indexing and storing streaming videos. Intelli-Vision [29] offers AI-based video analysis for smart cameras. They provide different services that include object tracking, human detection, vehicle counter, face recognition, customer count, license plate detection, traffic analytics, video search, video summary, and so forth. Google Vision API [30] offers a Video Data management and retrieval framework. They also provide APIs for video processing but does not support an intelligent video surveillance system. IBM Intelligent Video Analytics [31] provides batch video data processing and real-time video stream processing. However, they do not support API based services.

3. SIAT Framework

The framework, depicted in Figure 1, which consists of five main layers, Big Data Curation Layer (BDCL), Distributed Video Data Processing Layer (DVDPL), Distributed Video Data Mining Layer (DVDML), Knowledge Curation Layer (KCL) and Service Curation Layer (SCL) respectively. Moreover, the proposed framework has 3 basic users with roles: Administrator, Third Party Developers and End-Users. An administrator is who provides and manages the platform. They are also responsible for developing and providing end services to the users and providing APIs to the developers. While Third Party Developer can use the existing video processing APIs provided by the framework or can develop new APIs to produce new services. Finally, the End-User (Consumers) can use the provided services after performing the registry and subscription. In the following, the SIAT framework layers are discussed in detail.

3.1. Big Data Curation Layer

SIAT is supposed to consume 24/7 real-time video streaming in the cloud from millions of multi-modal video data stream sources and a large scale of batch video data from millions of consumers with the aim of intelligent video data analytics. The volume of the video data can easily reach the petabytes scale. For the sack of distributed video analytics, SIAT's DVDP and DVDM layers can deploy different distributed video processing APIs and Spark's MLLib algorithms, that is, supervised, unsupervised or deep learning algorithms. The life cycle of a video analytics service is data intensive and a systematic method is required to efficiently apply a wide spectrum of advanced distributed video processing and mining algorithms for large-scale video analytics problems, using Big Models.

In order to manage large scale streams, batch data and to meet the demands of distributed video analytics life cycle and data management requirements of the DVDP and DVDM Layers, we introduce SIAT's Big Data Curation Layer (BDCL). BDCL is composed of three modules, that is, Video Data Acquisition, Persistent Storage and Data Access Controller.

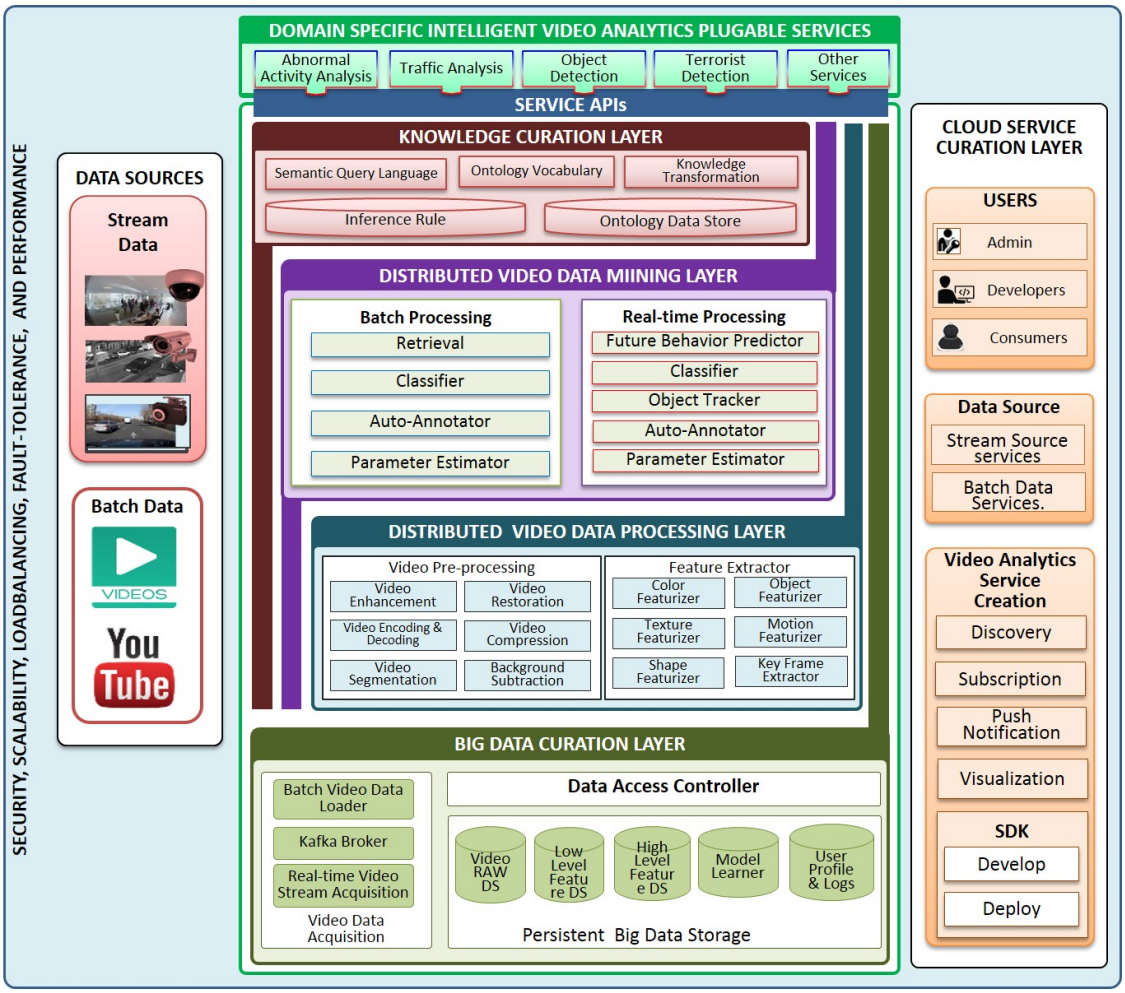

Figure 1. Proposed SIAT Framework.

3.1.1. Real-Time Video Stream Acquisition

SIAT BDCL has the ability to acquire a large scale of video streams from multi-model video data sources for on-the-fly processing. For large scale stream acquisition, technologically, we deploy Apache Kafka [11]. Apache Kafka is an open-source publishes-subscribe based messaging system responsible for transferring data from one application to another. Kafka Messaging system is comprised of three components, that is, Message Producer, Message Broker and Message Consumer [32]. Apache Kafka runs in a server cluster that stores streams of records in categories called topics or categories.

For communication from a multi-modal video data source, a JSON object is defined. The contents of this object consist of six fields, that is, user id, data source id, number of columns and number of rows in a frame, type, timestamp of the data origination at a data source and payload. This JSON object is known as Record. After the successful formation of the Record, the message needs to be sent to the Kafka Broker. The Record is then compressed while using snappy compression algorithm [33] with the aim to consume less network bandwidth and less space on the Kafka Cluster. The compressed message is then forwarded to the Producer.

The Real-time Video Stream Acquisition extends the Kafka Producer APIs with the aim to rout the mini batch of video streams to a specific topic in the Kafka Broker. Kafka Broker is an independent

application that is responsible to buffer, and deliver the Records to the consumers being received from the Producer. Kafka Broker is composed of multiple topics and each topic is a synonym to real-time video stream analytics service. A topic consists of partitions and is used to ensure height throughput and allow multiple consumers to read the mini batch of video stream data from the Kafka Broker. The Kafka Buffer can be replicated with the aim to ensure fault-tolerance.

3.1.2. Big Data Persistence

The data persistent component is responsible for providing permanent and distributed big-data persistence to both the structured and unstructured data of the SIAT framework. This module also stores and manage the meta-data of the above layers of SIAT. Technologically, Data Persistent storage is built on the top of Hadoop Distributed File System (HDFS) [8] and NoSQL data stores (HBase [34] while using Apache Phoenix [35]) as a query execution engine.

Persistent Big Data Storage is built on top of HDFS and is responsible for storing raw data, both batch video data and the streaming videos from various sources. Furthermore, it also provides storage facilities to store and manage model learner. Similarly, the Structured Data Store, that is, User Profile and Logs, is responsible for storing and managing structure data related to users, access rights, distributed video data processing layer, distributed video data mining layer, knowledge curation layer, metadata of the multi-modal video data stream sources, batch datasets, Big Models, and service subscription information.

The above layer in SIAT needs either the bulk of data for batch processing or random read-write to the data stores. For this purpose, we use Data Access Controller. Data Access Controller is responsible to provide read-write access to the underlying data in a secure way according to SIAT business logic.

3.2. Service Curation Layer

SIAT design incorporates top-notch functionality into a simple unified role bases Service Curation Layer (SCL). The SCL is built on the top of low-level access APIs. Logically, and for the ease of understandability, the services can be categorized as Users Cloud Services, Data Sources Management Services, Video Analytics Algorithms and Services and Cluster Management Services.

The SIAT provide role-based services and supports three types of Users, that is, System Administrator, Third party developer and Consumers (end-user). System Administrator has full control of the system maintenance and is responsible for developing distributed video processing and mining APIs and services for the Consumers (end-user) and publishing new APIs for the developers. The Developer role is allowed to extend the functionality of the framework by developing new plugins (such as video analytics algorithms and services) while using the SDK services as shown in the sequence diagram in Figure 2. Similarly, the end-user (i.e., Consumers) roles are authorized to attach video data source and/or to upload the video datasets and the same can be subscribed to the respective available video analytics services according to needs and demand.

In this work, we are providing our framework, SIAT, as Software-as-a-Service to the consumers (end-users) who will directly use the end services as an application on a web browser based on the service subscription. Here, the consumers (end-users) do not need to think about the insight into the framework. While for the developers we are providing our framework SIAT as Platform-as-a-Service. SIAT enables the developer to produce customized services using video analytics services by exploiting the provided distributed video processing APIs and other machine learning libraries from Spark MLlib. The main advantages of these APIs are that the developer can produce any higher level services using these APIs without considering their insights, which makes the developer's life easier and simpler. Furthermore, the developer can also build their own APIs and can integrate with the system.

Cluster Management Services are provided for administration purposes to let the system administrator monitor the health and functionality of the SIAT.

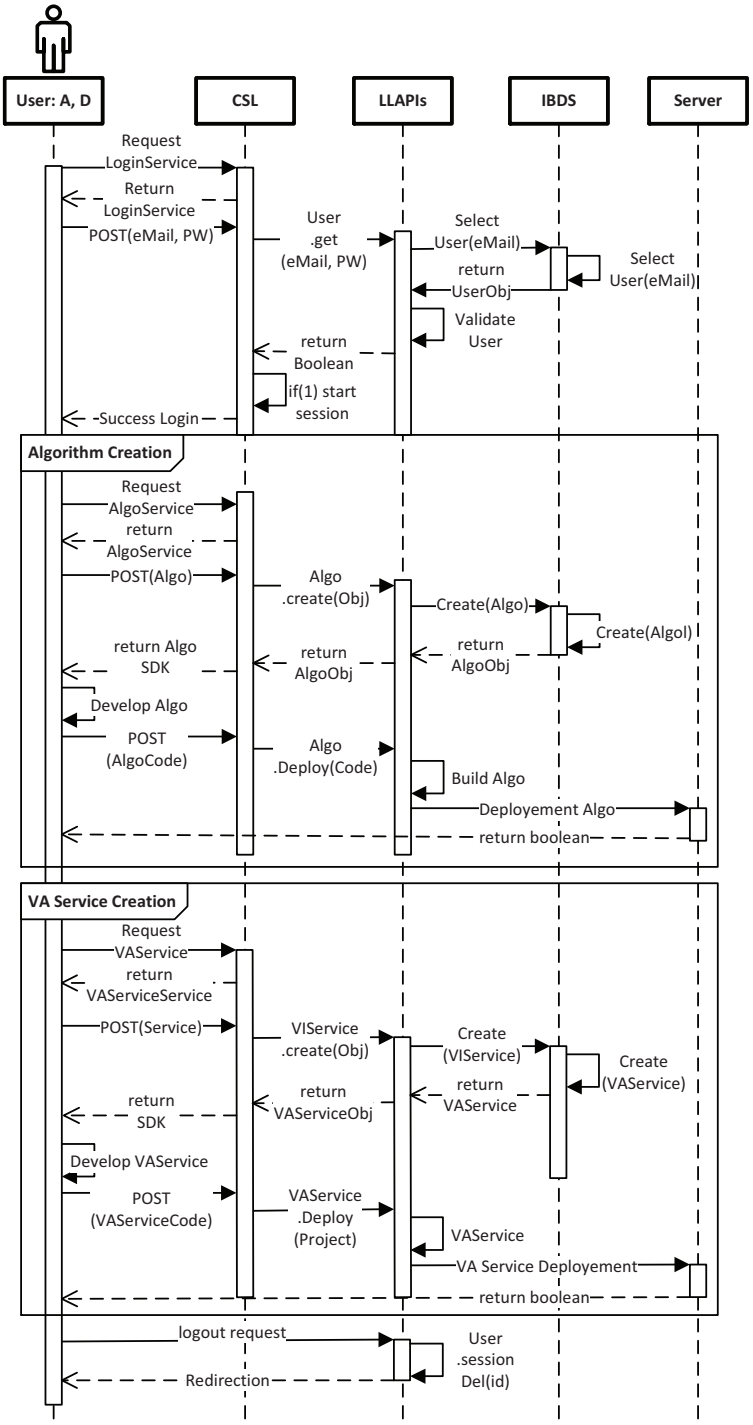

Figure 2. Sequence Diagram for Algorithm and Service Creation. In the Actor lifeline, A and D represents Admin and Developer respectively. These two roles are allowed to create new Video Analytics Algorithm and Service.

3.3. Distributed Video Data Processing Layer

Apache Spark is a distributed computing framework for large-scale data processing. Furthermore, Spark speeds up the process by supporting in-memory based computing. However, Spark provides almost no support for complex data types such as video. Furthermore, Spark does not support any high-level APIs for image or video processing. In this regard, MMLSpark [13] introduced a novel open source library that integrates the popular image

processing library OpenCV with Spark. However, their work does not provide any low-level video processing APIs. Furthermore, the existing literature also lacks support for dynamic feature extraction on the distributed environment, which is a very essential component to provide in any high-level services [10,16]. In order to resolve the above-mentioned issues, in this work, we introduce a distributed video processing library that provides distributed video processing on top of Spark. Our work integrates the image and video processing library JavaCV with Spark, so stays in the same ecosystem as much as possible. Our work was not only limited to providing basic distributed video processing APIs, but also supports distributed dynamic feature extraction APIs which extracts the prominent information in the video data. In our work, the Distributed Video Data Processing Layer (DVDPL) is mainly in charge of pre-processing and extracting the important features from the raw videos which are provided as input to the Distributed Video Data Mining Layer. It is composed of two main components, namely, Distributed Video Pre-processor and Distributed Feature Extractor. Distributed Video pre-processor is responsible for performing basic video processing algorithms such as video conversion, video enhancement, video restoration, video encoding and decoding, video compression, video segmentation, background subtraction, and so forth. While Distributed Feature Extractor is in charge of extracting the predominant features from the videos in a distributed manner. The component of Distributed Feature Extractor includes color feature extractor, dynamic and frame based texture feature extractor, dynamic and frame based shape feature extractor, motion feature extractor, object feature extraction and Key Frame extractor [10,16]. The pre-processing and Feature Extraction are employed based on the selection of higher-level services. The output of the Feature Extractor is represented either as Bag-of-Words [36] or Histogram [10,16] and stored in low-level result DS in the Big Data Curation Layer.

Figure 3 demonstrates the architectural overview to produce higher-level services using our proposed low-level distributed video processing APIs. Here, we integrated the popular image and video processing library JavaCV with Spark to support basic video processing operations, for example, frame extraction, frame conversion (RGB to gray-scale), and many more. On top of that, we built our distributed video processing and mining APIs, which plays an important role to provide any higher level services. Some sample APIs for distributed video processing are presented in Table 1. Our wrapper APIs permit us to achieve most video processing operations in parallel in a distributed environment, which reduces the processing time dramatically. Similar to our previous work [10,16], in this work, we have implemented several video processing algorithms on top of Spark that includes edge detection, video encoding, background subtraction, key-frame extraction and dynamic feature extraction. For distributed dynamic feature extraction, we have built APIs for Volume Local Binary Pattern (VLBP) [37], Volume Local Ternary Pattern (VLTP), Local Binary Pattern for three orthogonal planes (LBP-TOP) [37] and Directional Local Ternary Pattern for three orthogonal planes (DLTP-TOP) [16]. These dynamic feature extraction algorithms form a circularly symmetric neighbor set. For edge detection, we have implemented distributed Sobel operator, distributed Laplacian operator and distributed Canny operator [27]. Furthermore, we have developed distributed video encoding using MPEG and H264 [38]. We also deployed distributed key frame extractor [39] using Local Binary Pattern (LBP) [40] and Histogram of Oriented Gradient (HOG) [41] based features. The main advantages of these APIs are developer can produce any higher level services using these APIs without considering the insights of these APIs, which makes the developer life easy and simple. However, the developer can also build their own APIs and can integrate with the system.

3.4. Distributed Video Data Mining Layer

The Distributed Video Data Mining Layer (VDML) is responsible for producing the high-level semantic result from the features generated by the DVDPL. It provides two lower level services: Batched video data processing and Real-time video stream processing. In the batch processing, Video Classifier, Video Retrieval, Video Annotator are defined while in the real-time video stream processing Future Behavior Predictor, Video classifier, Object Tracker and Video Annotator are defined.

Video Classifier refers to classifying the videos based on pre-trained videos and Video Annotator refers to automatically assigning tags to the object or actions performed on the videos, while Video Retrieval refers to retrieving the similar looking videos from the Database. Lastly, Future Behavior Predictor is responsible for predicting the near future behavior or action by analyzing the previous few video frames. The parameter estimator is in charge of estimating the parameters. The result of DVDML is stored to High-Level result DS in the Big Data Curation Layer which works as an input to the Knowledge Curation Layer (KCL).

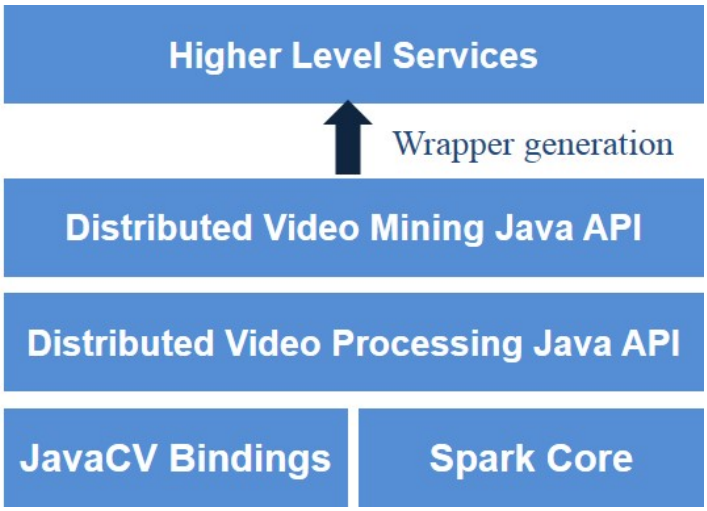

Figure 3. Architectural overview for wrapper API generation to produce the higher level services.

3.4.1. Video Data Mining Based on Spark MLlib

Spark MLlib [42] is a library developed on top of Apache Spark that provides access to a large number of machine learning algorithms. Spark MLlib comprises of fast and scalable executions of typical learning algorithms including classification, clustering, regression, collaborative filtering, and dimensionality reduction. It also supports numerous underlying statistics, linear algebra and optimization primitives including a generic gradient descent optimization algorithm. In our work, we have integrated the Spark MLlib to perform the distributed mining on the low-level feature data that are extracted by DVDPL.

3.4.2. Video Data Mining Based on Deep Learning Techniques

Recently, data-driven and computational intensive approaches like neural networks or deep learning (DL) have attracted lots of attention from the research community. This is because of their efficiency to extract hierarchical features from the raw data. Accordingly, they enable to discover significantly meaningful information from the vast amount of large-scale databases. Among the top deep learning techniques, Convolutional Neural Network (CNN) is the most popular one with particular focus on classifying and learning image data. By inheriting the success of deep learning, several tools or libraries (e.g., TensorFlowonSpark, Sparknet, DL4J, and BigDL) have been developed by integrating DL capabilities with big data frameworks like Apache Spark. However, most of the current tools have not been well-supported to video processing and Spark MLlib also lacks DL functions. Therefore, to complement our video data mining based on Spark Mllib and overcome existing limitations, we further incorporate DL libraries into our DVDML. Our DL library not only fits into Spark APIs but also provide a uniform set of APIs for programming of video understanding applications. In other words, this integration makes DL more accessible to scalable video processing. Within the scope of this paper, we develop our DL library based on BigDL [21] stack, which not only is compatible with most of popular DL framework (e.g., Tensorflow, Keras, and Caffe) but also provide the comprehensive support of DL technologies such as training and inference in the distributed setting.

Table 1. Sample APIs for distributed video data processing.

Category	APIs	Description
Background Subtraction	siat. dvdpl. BackgroundSubtractor. MOG (Video Data)	Subtract the background from the foreground of a video.
Edge Detection	siat. dvdpl. Edgedetector. SobelOperator (Video Data)	Detect the edge on each frame of the video using the Sobel operator.
	siat. dvdpl. Edgedetector. LaplacianOperator (Video Data)	Detect the edge on each frame of the video using the Laplacian operator.
	siat. dvdpl. Edgedetector. Canny(Video Data)	Detect the edge on each frame of the video using the Canny algorithm.
Video Encoding	siat. dvdpl. VideoEncoder. MPEG (Video Data)	Encode the video data for video data compression using MPEG algorithm.
	siat. dvdpl. VideoEncoder. H264 (Video Data)	Encode the video data for video data compression using H264 algorithm.
Key Frame Extractor	siat. dvdpl. KeyFrameExtractor. LBP (Video Data)	Extract the key frames from each video using frame based feature extractor Local Binary Pattern (LBP).
	siat. dvdpl. KeyFrameExtractor. HOG (Video Data)	Extract the key frames from each video using frame based feature extractor Histogram of Oriented Gradient (HOG).
Dynamic Feature Extractor	siat. dvdpl. DynamicFeatureExtractor. VLBP (Video Data)	Extract the dynamic texture feature from each video using Volume Local Binary Pattern (VLBP).
	siat. dvdpl. DynamicFeatureExtractor. VLTP (Video Data)	Extract the dynamic texture feature from each video using Volume Local Ternary Pattern (VLTP).
	siat. dvdpl. DynamicFeatureExtractor. ALMD (Video Data)	Extract the dynamic motion feature from each video using Adaptive Local Motion Descriptor (ALMD).
	siat. dvdpl. DynamicFeatureExtractor. LBPTOP (Video Data)	Extract the dynamic texture feature from each video using Local Binary Pattern from Three orthogonal planes (LBP-TOP).
	siat. dvdpl. DynamicFeatureExtractor. DLTPTOP (Video Data)	Extract the dynamic texture feature from each video using Directional Local Ternary Pattern from Three orthogonal planes (DLTP-TOP).
Deep Feature Extractor	siat. dvdpl. DeepFeatureExtractor. CNN (Video Data)	Extract the deep spatial feature from each video using Convolutional Neural Network (CNN).

3.5. Knowledge Curation Layer

There are two types of approaches for ontology development in video surveillance domain: developing ontology based on camera output, which is event detection using semantic technology and getting extracted features from computer vision algorithms and developing ontology based on those results [43,44]. In our framework, the knowledge curation layer generates ontology and creates knowledge based on the extracted higher level features which are obtained from DVDML. Since low-level features are inadequate for representing the video semantics that is why we are using the high-level features from videos for concept mapping but those rely on human knowledge and experience. Moreover, manual semantic concept tagging is time-consuming and very difficult, sometimes it is biased because of the human error and impossible for surveillance videos in the unconstraint environment. Therefore, collaborative semantic video annotation has been utilized in our framework where more than one user annotating the same resource and improve the quality of tagging. We have reused different ontologies like MPEG-7 [43], MediaOnt [45] and proposed some new terms for annotation. After this, the Resource Description Framework (RDF) triples are generated in accordance with our generated ontology. In the end, the higher level metadata is stored in an RDF triple store which is then made available for querying and analysis on video data. Another challenge that we have faced during development and immature in the literature is the concept mapping in constrained videos like medical or sports domain is easy. However, in our case (unconstrained videos), it is difficult, which we have addressed in our approach.

4. Experimental Analysis

4.1. Experimental Setup

For SIAT, we set up the distributed Hortonworks Data Platform (HDP 2.6.1.0) cluster consisting of one server and 9 agents. HDP is created by a big data software company, Hortonworks, and integrates different state of the art distributed computing and storage technologies like HDFS, YARN, Spark, Hive, HBase and many more. We tune the YARN CPU and memory resources according to our requirements. Furthermore, we used OpenJDK version 1.8 for coding purposes.

For networking purposes, we use the ProSafe GSM7328S fully managed switches delivering 24 and 48 ports of auto-sensing 1000 Mbps interfaces for high-density copper connectivity. The specification and configuration of each system and the distributed cloud architecture is shown in Figure 4. In this Figure 4, each node has four parameters i.e., i5 | 4 | 8 | 1000. These parameters show the processor model, number of cores, size of ram in GBs and size of hard-dist in GB respectively.

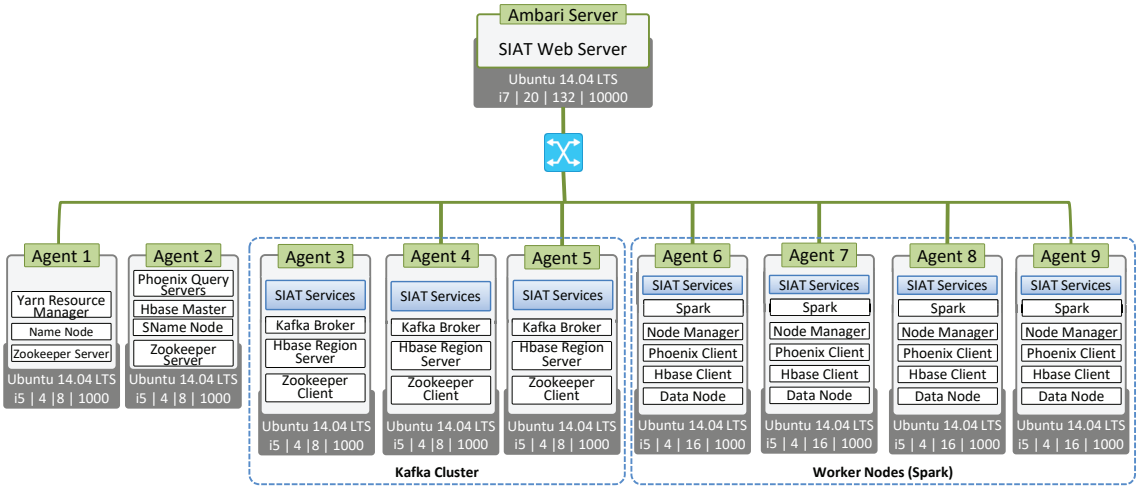

Figure 4. Distributed cloud environment for experimental analysis.

4.2. Application Scenarios

In order to evaluate the performance of the SIAT framework, we developed several application scenarios for both offline and online video data processing. Moreover, we have used scalability, Average Precision (AP) and fault tolerance as the evaluation metrics. Figure 5 illustrates the flow diagram for distributed offline and online video data processing. By coupling offline batch processing approach with real-time stream processing tools a version of symmetry can be accomplished. The presented scenario also represents the Lambda architecture. For the offline distributed video data processing, in the beginning, all the batch video data are stored in the HDFS, which are accessed through the Big Data Curation Layer for the respective selected service. Afterward, the video data are loaded into the Spark cluster. Subsequently, partitions of this video dataset are cached in each worker node which is represented by the Spark Resilient Distributed Dataset (RDD). RDD operations are performed in parallel to obtain a distributed result. In each partition, we perform distributed video pre-processing and feature extraction using defined low-level APIs from the Distributed Video Processing Layer. Finally, distributed mining algorithms from the DVDML are employed to obtain meaningful results, which includes video classification, video retrieval, video annotation and so forth. In contrast, for the online video data processing, we have adopted Apache Kafka and Spark Streaming. Apache Kafka is responsible for capturing video frames from multiple sources while Spark streaming is responsible for the distributed computing of video frames. Furthermore for online video processing, in the Spark streaming portion of distributed video processing, APIs are employed to perform the video pre-processing and dynamic feature extraction which plays a significant role in bringing out the outstanding information.

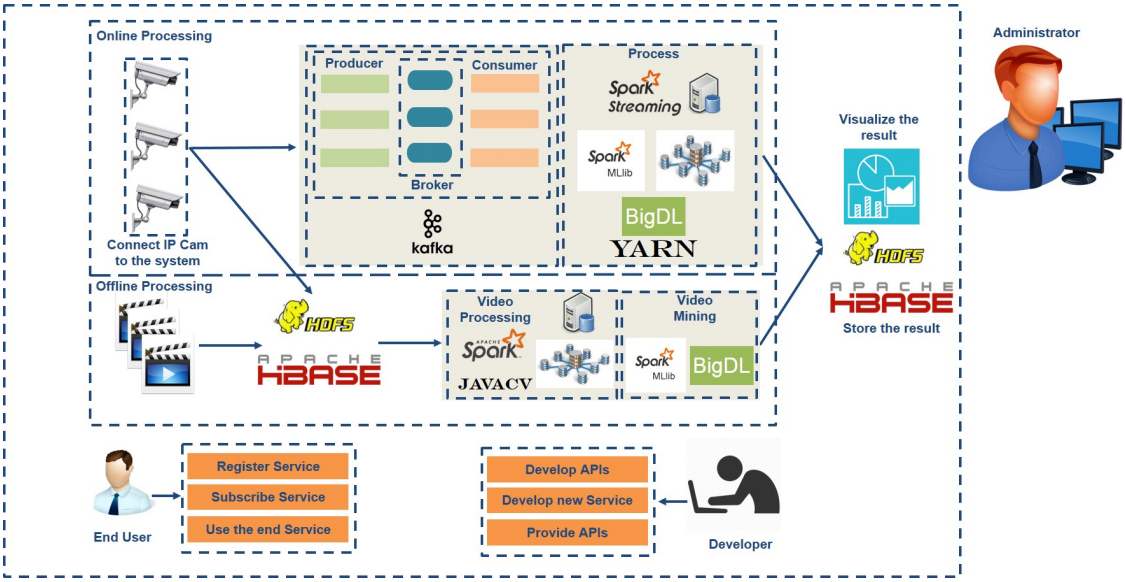

Figure 5. Flow diagram for distributed video processing.

4.2.1. Offline Service Scenario Applications

In this experiment, we have implemented dynamic feature extraction, edge detection, video encoding, and key frame extraction for batch video data in a distributed manner to measure the performance of our framework. Furthermore, we also developed human action recognition for large-scale video. In order to develop human action recognition in batch mode, we employed our dynamic feature extraction API along with Spark MLLib Random Forest API [42]. Similar to References [10,16], our distributed dynamic feature extraction API extracts the spatial and motion information from the video data which are represented as a histogram in the feature vector. Then these feature vectors are fed to the Spark MLLib Random Forest API to classify the human actions. In this work, we employed the KTH dataset [46] and the UCF50 dataset [47] to measure the scalability of

the proposed low-level distributed video processing APIs. Moreover, we also measured the Average Precision (AP) of human action recognition application for large scale video data. The KTH dataset consists of 600 videos including six human action classes: boxing, clapping, jogging, walking, running and waving, while the UCF-50 dataset is one of the biggest action datasets that consist of 6681 videos including 50 action categories.

Figure 6 demonstrates the scalability of low-level APIs for distributed video processing. In this experimental setup, we have implemented distributed dynamic feature extraction APIs for Volume Local Binary Pattern (VLBP) [37], Volume Local Ternary Pattern (VLTP), Local Binary Pattern for three orthogonal planes (LBP-TOP) [37] and Directional Local Ternary Pattern for three orthogonal planes (DLTP-TOP) [16]. For edge detection, we implemented distributed Sobel operator, distributed Laplacian operator and distributed canny operator [27]. Furthermore, we have developed distributed video encoding using MPEG and H264 [38]. We also deployed distributed key frame extractor [39] using Local Binary Pattern (LBP) [40] and Histogram of Oriented Gradient (HOG) [41] based features. From these experiments, we can see that with the increase of nodes our proposed distributed video processing APIs requires less time to bring out the outcomes, which proves the scalability of our framework. Furthermore on same experimental setup (using 4 worker nodes), we also compared our distributed video processing APIs running time with [36] for human action recognition on large scale video data. Figure 7 clearly demonstrates that our APIs shows better performance than the existing approach in terms of running time on batch data. In Reference [36], the authors employed trajectory based feature extraction, Gaussian Mixture Model and Fisher Vector Encoding algorithms for extracting and representing the features from videos, whereas our distributed video processing APIs only extracts the dynamic features and then these features are represented as histogram values.

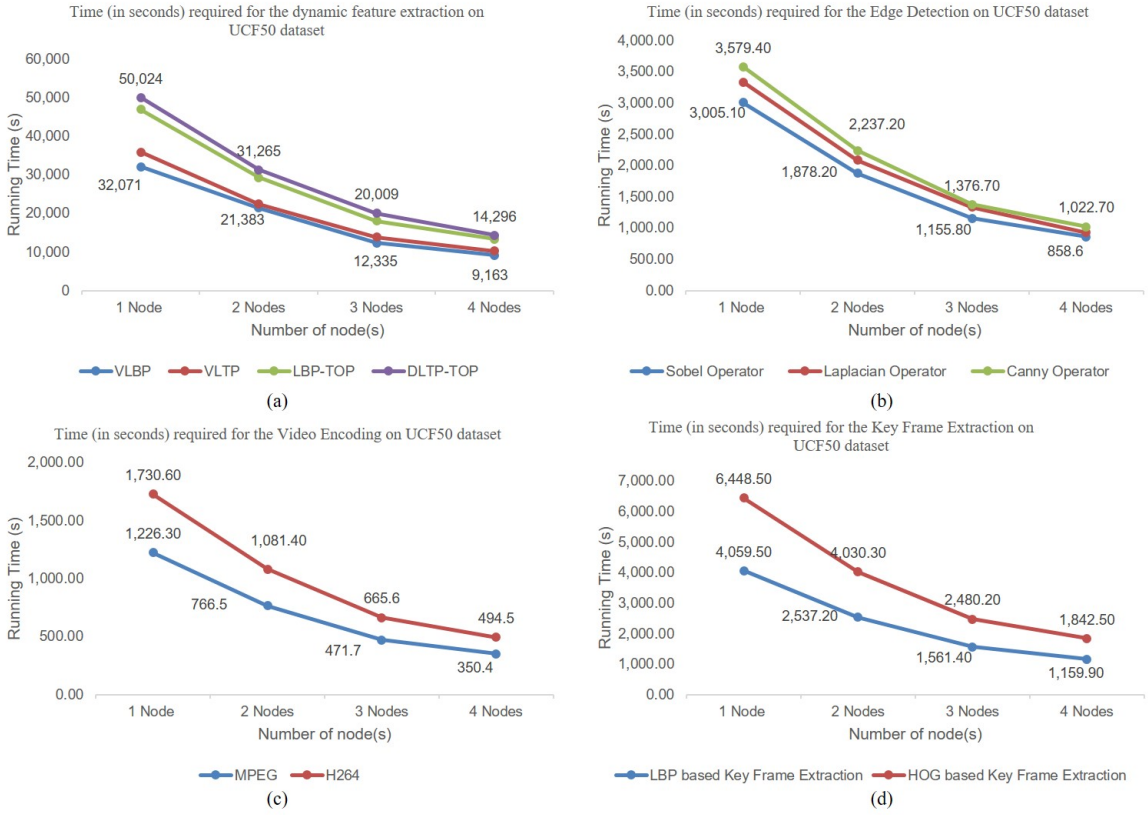

Figure 6. Scalability of low-level APIs for distributed video processing (**a**) Time (in seconds) required for the dynamic feature extraction on UCF50 dataset, (**b**) Time (in seconds) required for edge detection on UCF50 dataset, (**c**) Time (in seconds) required for video encoding on UCF50 dataset and (**d**) Time (in seconds) required for key frame extraction on UCF50 dataset.

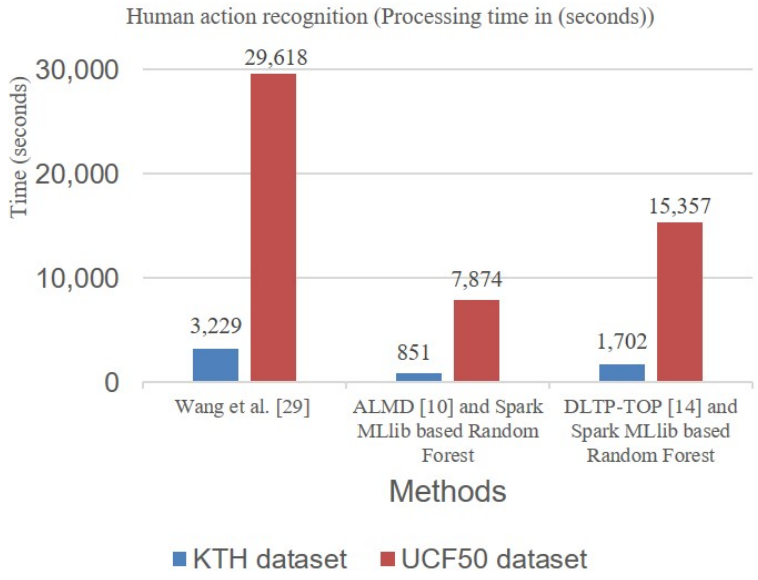

Figure 7. Comparison of proposed framework with existing work for human action recognition on large-scale video data.

Lastly, Figure 8 shows the Average Precision for human action recognition on UCF50 dataset [47] and KTH dataset [46] using distributed dynamic feature extraction approaches and Spark MLlib based Random Forest classifier. This experiment also shows the comparison with Reference [36] in terms of average precision. In Reference [36], the authors employed trajectory based feature extraction, GMM generation and feature vector encoding on top of Spark with gives 91.8% average precision on KTH dataset and 67.2% average precision on UCF50 dataset. Whereas our proposed APIs, ALMD and DLTP-TOP show 92.3% and 92.9% average precision on KTH dataset and 79.3% and 86.2% average precision on UCF50 dataset respectively. DLTP-TOP outperforms all other dynamic feature descriptors in terms of precision since the DLTP-TOP obtain more detailed discriminative features by encoding high-order derivative information.

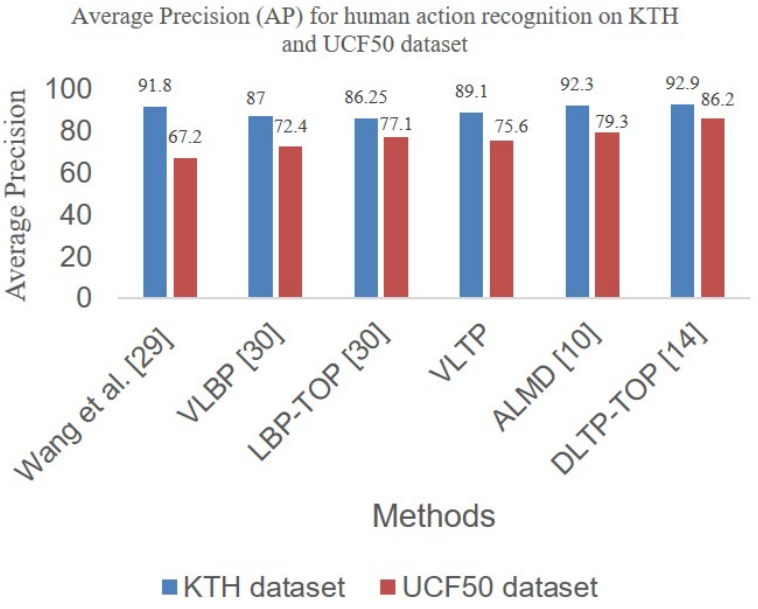

Figure 8. Average Precision (AP) for the human action recognition on KTH and UCF50 dataset.

Figure 9 further shows the scalability of our data mining layer based on deep learning API. Particularly, we implemented the deep feature extraction and classification under distributed fashion by using our DL library. Here, video datasets (i.e., Hollywood2 and UCF50) are loaded into RDD and we then apply the "flatMap" Spark operation to transform the video-based RDD into the frame-based RDD, which are fed into the pre-trained VGG16 model to extract the video features. We follow the workflow of Reference [48] for classification but our pipeline is performed on the distributed cluster rather than on a single machine. We run the deep feature extraction test and the classification test from 1 to 4 nodes. Both tests have shown the good scalability of our DL functions.

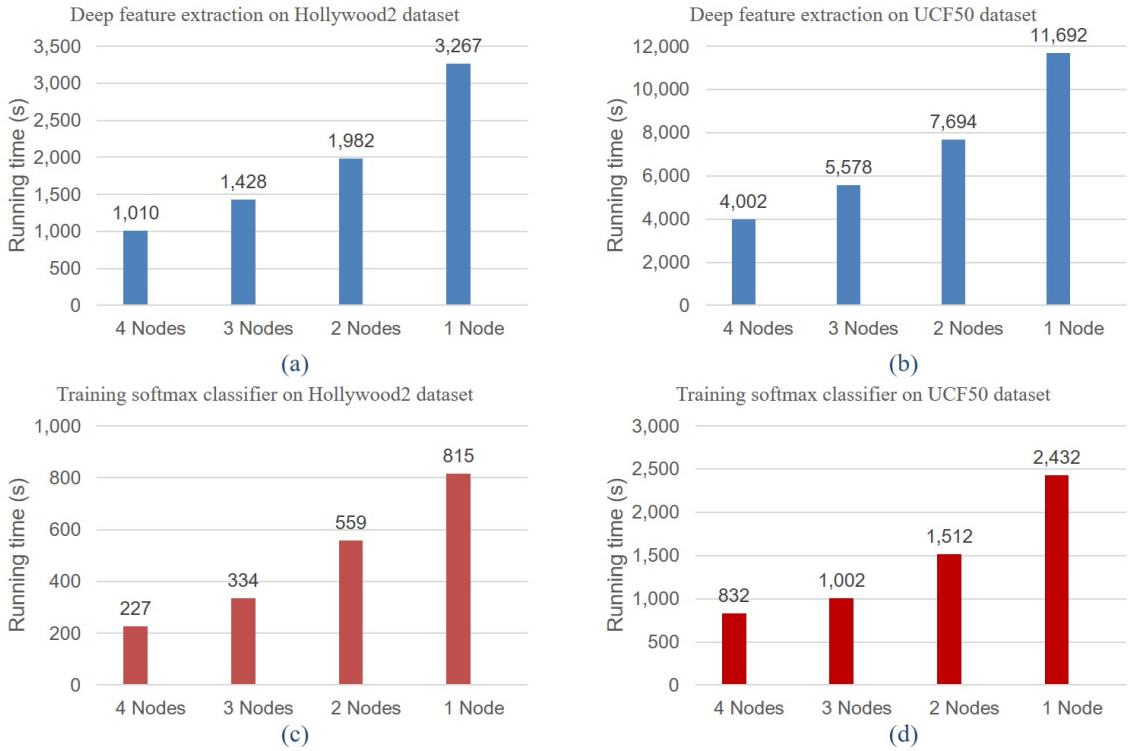

Figure 9. Scalability test of deep learning API for distributed feature extraction and training classifier. (**a**), Running time for deep feature extraction on Hollywood2 dataset; (**b**), Running time for deep feature extraction on UCF50 dataset; (**c**), Running time for training softmax classifier on Hollywood2 dataset; (**d**), Running time for training softmax classifier on UCF50 dataset.

4.2.2. Online Service Scenario Applications

To evaluate the performance real-time service in our framework, we implemented real-time face recognition and real-time action recognition. For the online video data processing, 10 IP cameras taking real-time video are employed for evaluation. These 10 cameras capture 250 frames (25 frames by each camera) per second. The frame size is within 1M to 1.5M bytes for each frame. Figure 10 shows the face recognition and action recognition processing time with a varying number of cameras and nodes, which proves the scalability of our framework for real-time services. In this experimental setup, we employed three nodes as Kafka broker while four nodes for video data processing using Spark Streaming and each of these nodes operates as consumers. For the frame acquisition experiment, if we employ one camera stream with one producer, one broker, and one consumer it takes 34.5 milliseconds while employing three camera streams with two producers, two brokers and two consumers take 61.6 milliseconds to obtain the frames. From this experiment, we can see that, for the frame acquisition if the increase the number of cameras and decrease the number of brokers then it requires more time for the frame acquisition while increasing the number of brokers reduce the frame acquisition time. Increasing the number of brokers helps to transmit more video frames since the number of

partition is also increased based on the topic partitions of data. In addition, because the consumer controls the volume of arrival video frames in the Kafka framework, it is possible to achieve efficient transfer by varying the number of consumers. Subsequently, the processing time for face detection and action recognition is also decreased by adding more nodes, since video data are processed in a distributed manner. Figure 10c,d shows real-time face recognition time in (ms) with varying number of cameras and nodes, and real-time action recognition time in (ms) with varying number of cameras and nodes respectively. Furthermore, we also compared the proposed framework face recognition time with video cloud platform [7]. In this experiment, for group 1 (3 node and 3 camera streams) our platform takes 34.6 milliseconds while video cloud platform [7] takes 132.45 milliseconds and similarly, for group 3 (4 node and 10 camera streams) our platform takes 138.4 milliseconds while video cloud platform [7] takes 423.8 milliseconds to recognize face on each frame. We perform our experiment on spark streaming while video cloud platform [7] employed Storm for real-time video processing. Due to the facility of in-memory computing and micro-batching on spark streaming our platform outperformed video cloud platform [7].

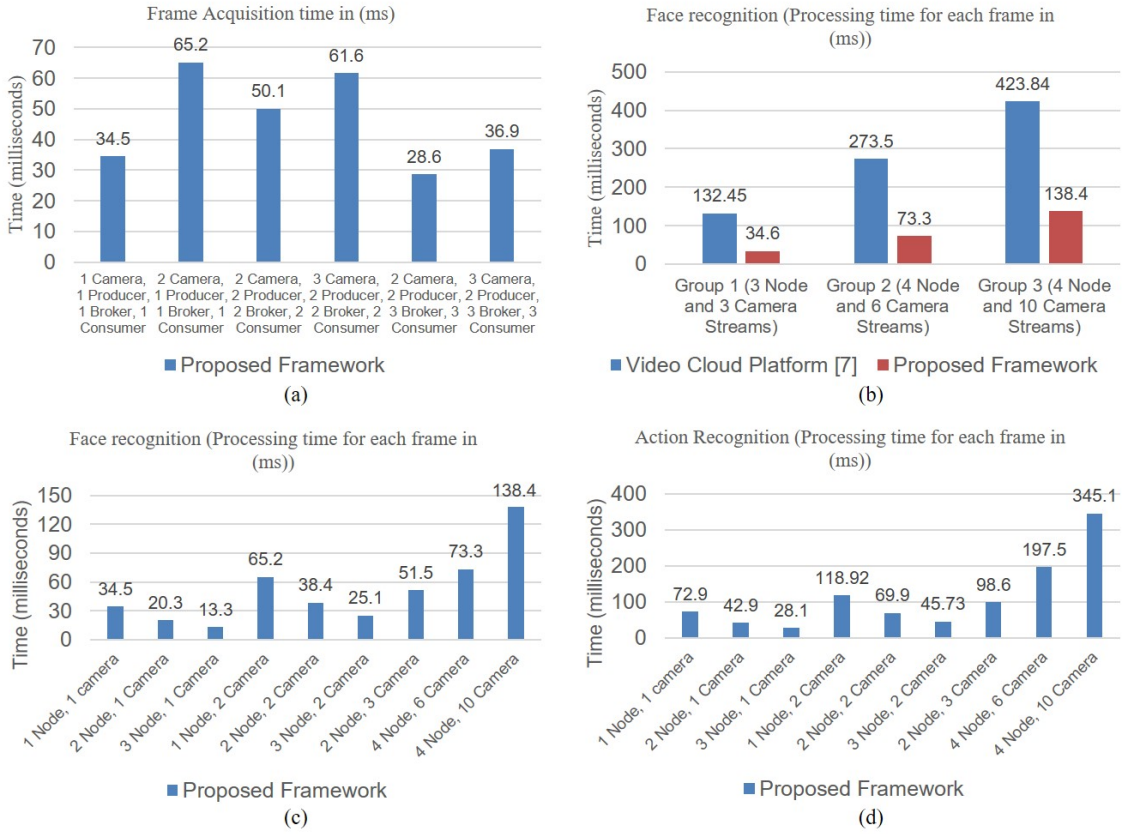

Figure 10. (**a**) Frame acquisition time in (ms) for online video processing, (**b**) comparison of proposed framework with existing framework for face detection, (**c**) Face recognition time in (ms) with varying number of cameras and nodes, and (**d**) Action recognition time in (ms) with varying number of cameras and nodes.

In the SIAT framework, we employed Apache Spark, which by default supports fault tolerance and we also used HDFS, in which a file is distributed into one or more blocks and these segments are saved in a set of Data Nodes with having more than one copy. In this work, we also measured the performance of our framework by killing nodes during face recognition. Figure 11 demonstrates the evaluation result. During the online processing, we have used four nodes for frame processing using spark streaming. Then we kill worker nodes and measure performance. In this experiment, when three nodes are running then one node is killed, while two nodes are running then two nodes are killed.

Eventually, all the experiments bring out the similar face recognition result for all the experimental scenarios (i.e., with or without killing nodes). Similarly, when 10 cameras are employed and two nodes are killed then it requires 308.2 ms for face recognition on each frame while if no nodes are killed then it requires 138.4 ms.

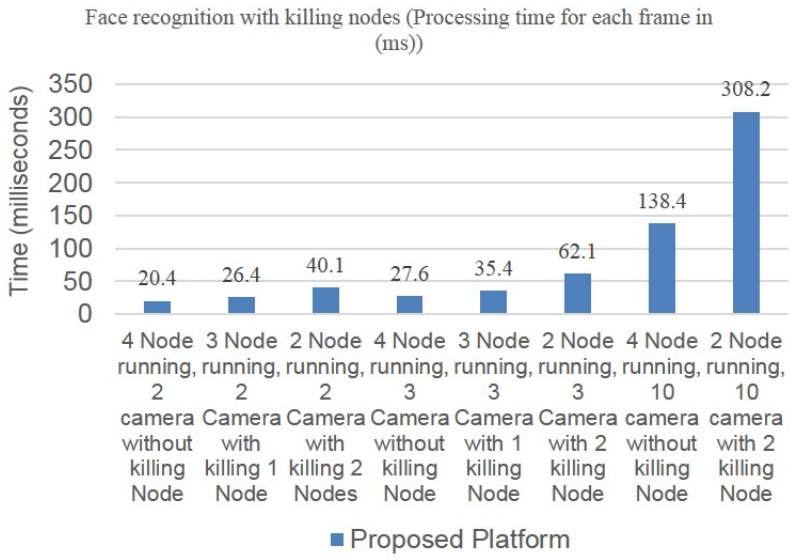

Figure 11. Performance evaluation by killing nodes during face recognition.

5. Conclusions

Online and offline big-data video analytics are always expensive in terms of transmission, storage, management and computation. In this paper, we proposed a robust, distributed, pluggable, layered and service-oriented cloud architecture with the intentions to extract the insight from a large scale video in almost real-time and/or offline. We then implement the proposed SIAT framework while exploiting state-of-the-art distributed streaming, data storage and processing technologies like Kafka, Hadoop, Hbase, Spark, Spark's MLLib and JavaCV. Moreover, we have introduced a distributed video processing library that provides video processing on top of Spark. Furthermore, for evaluation and experimental reasons, we develop real-time and offline video data mining services like Human Action recognition and Face recognition services respectively. The evaluation authenticates the scalability and fault tolerance of the proposed system and can be a candidate solution for large-scale video analytics.

The SIAT is an ongoing research project and demands further refinement. In the future, we will develop more real-world domain specific video analytics algorithms and further attention will be given to next-generation distributed deep learning and the back-propagation video analytics algorithms. Further investigation and video indexing are required in the data layer to make it more efficient. We will also perform experimental analysis for higher level services while considering the Knowledge Curation Layer. Besides, in the future, we will also consider the issues of a cloud-based system, for example, resource utilization, Security and Privacy, and so forth.

Author Contributions: Y.-K.L. provided guidance for improvement during the discussions; M.A.U. conceived the key idea, performed implementation for offline video data processing and was in charge of writing the manuscript; A.A. contributed to Big Data Curation Layer and also helped in writing the manuscript; M.S.I. worked on online video data processing; N.A.T. contributed by implementing the deep learning APIs and also evaluated the proposed idea.

Funding: This work was supported by Institute for Information and communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2016-0-00406, SIAT CCTV Cloud Platform).

Acknowledgments: This work was supported by Institute for Information and communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2016-0-00406, SIAT CCTV Cloud Platform).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. 37 Mind Blowing Youtube Facts, Figures and Statistics. Available online: https://merchdope.com/youtube-stats/ (accessed on 5 November 2018).
2. Video Streaming Now Makes Up 58% of Internet Usage Worldwide. Available online: http://digg.com/2018/streaming-video-worldwide (accessed on 5 November 2018).
3. IntelliVision Now Inside 4 Million Smart Cameras—Leader in AI-Based Video Analytics Software. Available online: https://www.intelli-vision.com/news/intellivision-now-inside-4-million-smart-cameras-leader-in-ai-based-video-analytics-software/ (accessed on 5 November 2018).
4. Video Analytics Market - Global Forecast to 2023. Available online: https://www.researchandmarkets.com/reports/4530884/videoanalytics-market-by-type-software-and (accessed on 5 November 2018).
5. Hossain , M.A. Framework for a Cloud-Based Multimedia Surveillance System. *Int. J. Distrib. Sens. Netw.* **2014**, *10*, 135257. [CrossRef]
6. Nazare, A.C., Jr.; Schwartz, W.R. A scalable and flexible framework for smart video surveillance. *Comput. Vis. Image Underst.* **2016**, *144*, 258–275. [CrossRef]
7. Zhang, W.; Xu, L.; Duan, P.; Gong, W.; Lu, Q.; Yang, S. A video cloud platform combing online and offline cloud computing technologies. *Pers. Ubiquitous Comput.* **2015**, *19*, 1099–1110. [CrossRef]
8. Shvachko, K.; Kuang, H.; Radia, S.; Chansler, R. The hadoop distributed file system. In Proceedings of the IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), Incline Village, NV, USA, 3–7 May 2010; pp. 1–10.
9. Zaharia, M.; Chowdhury, M.; Das, T.; Dave, A.; Ma, J.; McCauley, M.; Franklin, M.J.; Shenker, S.; Stoica, I. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In Proceedings of the Networked Systems Design and Implementation (NSDI'12), San Jose, CA, USA, 25–27 April 2012; p. 2-2.
10. Uddin, M.A.; Joolee, J.B.; Alam, A.; Lee, Y.-K. Human action recognition using adaptive local motion descriptor in spark. *IEEE Access* **2017**, *5*, 21157–21167. [CrossRef]
11. Kreps, J.; Narkhede, N.; Rao, J. Kafka: A distributed messaging system for log processing. In Proceedings of the NetDB, Athens, Greece, 12–16 June 2011; pp. 1–7.
12. Zaharia, M.; Das, T.; Li, H.; Hunter, T.; Shenker, S.; Stoica, I. Discretized Streams: Fault-Tolerant Streaming Computation at Scale. In Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles, Farminton, PA, USA, 3–6 November 2013; pp. 423–438.
13. Hamilton, M.; Raghunathan, S.; Annavajhala, A.; Kirsanov, D.; Leon, E.d.; Barzilay, E.; Matiach, I.; Davison, J.; Busch, M.; Oprescu, M.; et al. Flexible and scalable deep learning with MMLSpark. In Proceedings of the 4th International Conference on Predictive Applications and APIs, Boston, MA, USA, 24–25 October 2017; pp. 11–22.
14. Huang, Q.; Ang, P.; Knowles, P.; Nykiel, T.; Tverdokhlib, I.; Yajurvedi, A.; Dapolito, P., IV; Yan, X.; Bykov, M.; Liang, C.; et al. SVE: Distributed Video Processing at Facebook Scale. In Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, 28–31 October 2017; pp. 87–103.
15. Ichinose, A.; Takefusa, A.; Nakada, H.; Oguchi, M. A study of a video analysis framework using Kafka and spark streaming. In Proceedings of the IEEE International Conference on Big Data, Boston, MA, USA, 11–14 December 2017.
16. Uddin, M.A.; Akhond, M.R.; Lee, Y.-K. Dynamic scene recognition using spatiotemporal based DLTP on spark. *IEEE Access* **2018**, *6*, 66123–66133. [CrossRef]
17. Hu, C.; Xue, G.; Mei, L.; Qi, L.; Shao, J.; Shang, Y.; Wang, J. Building an intelligent video and image analysis evaluation platform for public security. In Proceedings of 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Lecce, Italy, 29 August–1 September 2017; pp. 1–6.
18. Chao, W.; Jun, X.M. Multi-agent based distributed video surveillance system over IP. In Proceedings of the International Symposium on Computer Science and Computational Technology, Shanghai, China, 20–22 December 2008; Volume 2, pp. 97–100.

19. Ostheimer, D.; Lemay, S.; Ghazal, M.; Mayisela, D.; Amer, A.; Dagba, P.F. A modular distributed video surveillance system over IP. In Proceedings of the Canadian Conference on Electrical and Computer Engineering, Ottawa, ON, Canada, 7–10 May 2006; pp. 518–521.
20. Zhang, H.; Yan, J.; Kou, Y. Efficient online surveillance video processing based on spark framework. In Proceedings of the International Conference on Big Data Computing and Communications, Shenyang, China, 29–31 July 2016; pp. 309–318.
21. Dai, J.; Wang, Y.; Qiu, X.; Ding, D.; Zhang, Y.; Wang, Y.; Jia, X.; Zhang, C.; Wan, Y.; Li, Z.; et al. BigDL: A Distributed Deep Learning Framework for Big Data. Available online: https://bigdl-project.github.io/master/whitepaper/ (accessed on 5 November 2018).
22. Lv, J.; Wu, B.; Yang, S.; Jia, B.; Qiu, P. Efficient large scale near-duplicate video detection base on spark. In Proceedings of the IEEE International Conference on Big Data, Washington, DC, USA, 5–8 December 2016.
23. Lv, J.; Wu, B.; Liu, C.; Gut, X. PF-Face: A Parallel Framework for Face Classification and Search from Massive Videos Based on Spark. In Proceedings of the IEEE Fourth International Conference on Multimedia Big Data (BigMM), Xi'an, China, 13–16 September 2018.
24. Huang, L.; Xu, W.; Liu, S.; Pandey, V.; Juri, N.R. Enabling Versatile Analysis of Large Scale Traffic Video Data with Deep Learning and HiveQL. In Proceedings of the IEEE International Conference on Big Data, Boston, MA, USA, 11–14 December 2017.
25. Yaseen, M.U.; Anjum, A.; Rana, O.; Hill, R. Cloud-based scalable object detection and classification in video streams. *Future Gener. Comput. Syst.* **2018**, *80*, 286–298. [CrossRef]
26. Iqbal, B.; Iqbal, W.; Khan, N.; Mahmood, A.; Erradi, A. Canny edge detection and Hough transform for high resolution video streams using Hadoop and Spark. *Clust. Comput.* **2019**, 1–12. [CrossRef]
27. Maini, R.; Aggarwal, H. Study and comparison of various image edge detection techniques. *Int. J. Image Process.* **2009**, *3*, 1–11.
28. Eagle Eye Networks. Available online: https://www.eagleeyenetworks.com/ (accessed on 5 November 2018).
29. Intelli-Vision. Available online: https://www.intellivision.com/ (accessed on 5 November 2018).
30. Google Vision API. Available online: https://cloud.google.com/video-intelligence/ (accessed on 5 November 2018).
31. IBM Intelligent Video Analytics. Available online: https://www.ibm.com/cloud/ (accessed on 5 November 2018).
32. Ejsmont, A. *Web Scalability for Startup Engineers*, 1st ed.; McGraw-Hill Education Group: Boston, MA, USA, 2015.
33. Snappy. Available online: https://google.github.io/snappy/ (accessed on 5 November 2018).
34. George, L. *HBase: The Definitive Guide: Random Access to Your Planet-Size Data*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2011.
35. Apache Phoenix. Available online: https://phoenix.apache.org/ (accessed on 5 November 2018).
36. Wang, H.; Zheng, X.; Xiao, B. Large-scale human action recognition with spark. In Proceedings of the IEEE 17th International Workshop on Multimedia Signal Processing (MMSP), Xiamen, China, 19–21 October 2015.
37. Zhao, G.; Pietikainen, M. Dynamic texture recognition using local binary patterns with an application to facial expressions. *IEEE Trans. Pattern Anal. Mach. Intell.* **2007**, *29*, 915–928. [CrossRef] [PubMed]
38. Chiang, T.; Lee, H.-J.; Pejhan, S.; Sodagar, I.; Zhang, Y.-Q. Demonstration of the mpeg-2, mpeg-4 and h.263 video coding standards. In Proceedings of the First Signal Processing Society Workshop on Multimedia Signal Processing, Princeton, NJ, USA, 23–25 June 1997.
39. Ejaz, N.; Tariq, T.B.; Baik, S.W. Adaptive key frame extraction for video summarization using an aggregation mechanism. *J. Vis. Commun. Image Represent.* **2012**, *23*, 1031–1040. [CrossRef]
40. Ojala, T.; Pietikainen, M.; Maenpaa, T. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. Pattern Anal. Mach. Intell.* **2002**, *24*, 971–987. [CrossRef]
41. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–25 June 2005.
42. Meng, X.; Bradley, J.; Yavuz, B.; Sparks, E.; Venkataraman, S.; Liu, D.; Freeman, J.; Tsai, D.B.; Amde, M.; Owen, S.; et al. MLlib: Machine learning in apache spark. *J. Mach. Learn. Res.* **2016**, *17*, 1235–1241.

43. Yang, N.C.; Chang, W.H.; Kuo, C.M.; Li, T.H. A fast mpeg-7 dominant color extraction with new similarity measure for image retrieval. *J. Vis. Commun. Image Represent.* **2008**, *19*, 92–105. [CrossRef]
44. Sah, M.; Direkoglu, C. Semantic annotation of surveillance videos for abnormal crowd behaviour search and analysis. In Proceedings of 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Lecce, Italy, 29 August–1 September 2017.
45. Mediaont. Available online: https://www.w3.org/TR/mediaont-10/ (accessed on 5 November 2018).
46. Schuldt, C.; Laptev, I.; Caputo, B. Recognizing human actions: A local svm approach. In Proceedings of the 17th International Conference on Pattern Recognition, ICPR 2004, Cambridge, UK, 26 August 2004; Volume 3, pp. 32–36.
47. Reddy, K.K.; Shah, M. Recognizing 50 human action categories of web videos. *Mach. Vis. Appl. J. (MVAP)* **2012**, *24*, 971–981. [CrossRef]
48. Simonyan, K.; Zisserman, A. Two-stream convolutional networks for action recognition in videos. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 568–576.