Pervasive and Mobile Computing 76 (2021) 101441

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

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

Multi-level feature driven storage management of surveillance videos

Min-Han Tsai [a], Nalini Venkatasubramanian [b], Cheng-Hsin Hsu [a,*]

[a] National Tsing Hua University, Taiwan
[b] University of California, Irvine, USA

A R T I C L E I N F O

Article history:
Received 18 January 2021
Received in revised form 18 May 2021
Accepted 24 June 2021
Available online 28 June 2021

Keywords:
Smart environments
Edge computing
Video analytics
Optimization
Internet-of-things

A B S T R A C T

Surveillance videos in smart environments have become commodities nowadays, which enable many novel applications, including various video analytics that turn videos into semantic results. In addition to live feeds, surveillance videos may be saved in a storage server for on-demand user-defined queries in the future. Different from on-demand video streaming servers, whose design objective is to maximize the user-perceived video quality, a surveillance video storage server has limited space and must retain as much information as possible while reserving sufficient space for incoming videos. In this article, we design, implement, optimize, and evaluate a multi-level feature driven storage server for diverse-scale smart environments, which can be buildings, campuses, communities, and cities. We focus on the design and implementation of the storage server and solve two key research problems in it, namely: (i) efficiently determining the information amount of incoming videos and (ii) intelligently deciding the qualities of videos to be kept. In particular, we first analyze the videos to derive approximate information amount without overloading our storage server. This is done by formally defining the information amount based on multi-level (semantic and visual) features of videos. We then leverage the information amounts to determine the optimal downsampling approach and target quality level of each video clip to save the storage space, while preserving as much information amount as possible. We rigorously formulate the above two research problems into mathematical optimization problems, and propose optimal, approximate, and efficient algorithms to solve them. Besides a suite of optimization algorithms, we also implement our proposed system on a smart campus testbed at NTHU, Taiwan, which consists of eight smart street lamps. The street lamps are equipped with a wide spectrum of sensors, network devices, analytics servers, and a storage server. We compare the performance of our proposed algorithms against the current practices using real surveillance videos from our smart campus testbed. Our efficient algorithms outperform the current practices in multiple dimensions, meaning we: (i) achieve a mere 7% approximation gap on captured information amount compared to the optimal solutions, (ii) save almost 3 times more clips after a week, (iii) achieve 58% less per-query error on average, (iv) always terminate in less than 100 ms, (v) do not consume excessive storage space, and (vi) scale well with larger storage spaces.

* Corresponding author.
E-mail address: chsu@cs.nthu.edu.tw (C.-H. Hsu).

https://doi.org/10.1016/j.pmcj.2021.101441
1574-1192/© 2021 Elsevier B.V. All rights reserved.

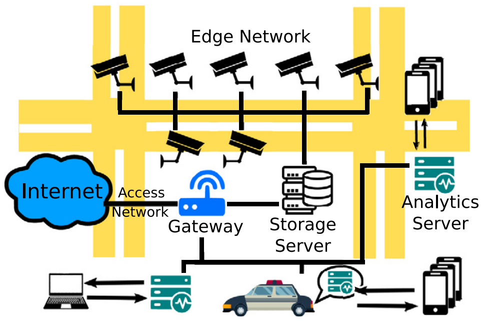

Fig. 1. A sample IoT edge network, consisting of cameras (and other sensors), a gateway, a storage server, and several analytics servers.

1. Introduction

Increasingly more surveillance cameras are being deployed in smart environments, such as cities, home, and workspaces, for enhanced security. Due to the broad applications of surveillance cameras and mature techniques like artificial intelligence, the global market of surveillance cameras is constantly growing. For instance, the surveillance camera market of the smart home sector will reach $9.7 billion by 2023, while the market share including other sectors is even larger. More and more cameras in smart environments offer novel and diverse analytics applications, such as object detection and tracking [1–3], face recognition [4,5], health monitoring [6,7], and traffic management [8–10]. Nowadays, the cameras upload surveillance video clips to data centers for storage and analytics [11–13]. Doing so, however, may lead to high operational cost and network congestion because each surveillance camera produces a traffic stream at several Mbps. On top of that many of these video clips are never queried by end users throughout their life-cycles. Therefore, a better way to manage the video clips is to store them on a *storage server* in an *edge* network, as illustrated in Fig. 1. The edge network interconnects multiple nearby Internet-of-Things (IoT) devices, including surveillance cameras, and connects to the Internet via a *gateway* through an *access network*. By not uploading all the video clips to the cloud, the traffic load on the access network is reduced. When users need to query/analyze the surveillance videos, they instruct nearby *analytics servers*, which in turn request for corresponding video clips from the storage server. These analytics servers could be: (i) stationary at base stations, serving close-by smartphones and laptops or (ii) mobile in police cars, fire trucks, and ambulances, serving end users in them.

Keeping surveillance video clips at a storage server, however, may quickly fill up its disk. For example, storing 1 Mbps video clips from 10 surveillance cameras for merely a week consumes 1.4 TB disk space, but surveillance video clips are typically archived for much longer than a week. When the storage space of the edge server is used up, we have to get rid of some video clips to make room for incoming ones. A naive way to do this is to delete the oldest video clips. Doing so, however, may lead to too much information loss, because video clips from different cameras at different times contain *different* amounts of information. In fact, video clips that contain some useful information are better *downsampled* (in temporal, spatial, fidelity, or other aspects) instead of being completely deleted. The downsampled video clips can still be queried/analyzed for useful analytics results in the future.

The resulting information amount from analytics queries depends on the end-user needs and the surveillance video content. Different video clips contain diverse amounts of information, which can be characterized by the *semantic* and *visual* features. The semantic features are high level, which directly reflect the user-intended queries, e.g., the business owners pay more attention to the number of pass-by pedestrians at the intersections, while the police department cares whether there are illegally parked or speeding cars. The visual features are low level, e.g., color distributions and dominated edges, which are more general across queries with heterogeneous analytics. Fig. 2 illustrates sample semantic and visual features. Among these features, the semantic ones better fit the user queries as long as the analytics are known. In contrast, the visual features are more general for unknown analytics in future user queries, but may not be exactly aligned with the actual analytics. Hence, we consider both semantic and visual features, striving to get the best out of them both.

In this article, we design, implement, and optimize a storage server for saving the surveillance video clips. *The goal is to retain video clips with the highest information amounts and selectively downsample the stored video clips to make room for future ones.* This is, however, no easy task for the following reasons:

- **Challenge #1:** Different video clips contain diverse information amounts, which depend on the dynamic query demands of video analytics from end users.
- **Challenge #2:** Different downsampling approaches lead to diverse amounts of information loss.
- **Challenge #3:** Quantifying the information amount requires executing video analytics and downsampling video clips requires video transcoding. Both video analytics and downsampling are computationally intensive and thus need to be carefully scheduled.

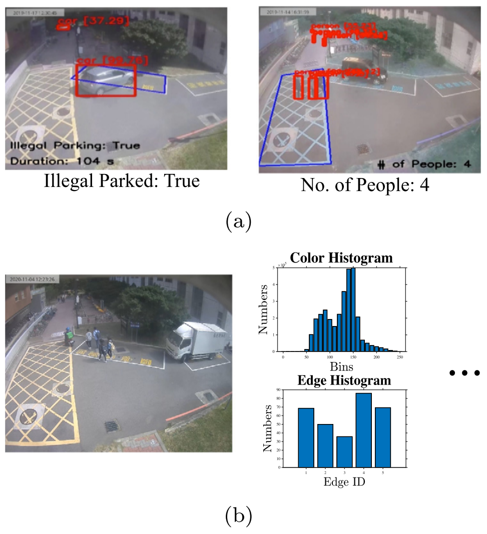

Fig. 2. Sample features: (a) semantic and (b) visual ones.

We address the above three challenges as follows. We first define the information amount to systematically guide the decisions made on our storage server. We then employ multiple downsampling approaches for free up the storage space. We study two key optimization problems of our proposed storage server. The first problem is selecting a sampling length of each video clip to analyze, in order to approximate the actual information amount without overloading the storage server. The second problem is choosing the downsampling approaches and quality levels for individual video clips to preserve as much information as possible, while making enough room for incoming video clips. For each of the optimization problems, we first give its optimal and approximation algorithms with analysis. For better efficiency and practicality, we also propose heuristic algorithms for both research problems. Our experiments reveal that our heuristic algorithms: (i) reach 58% less per-query error on average, (ii) save nearly 2.78 times more clips after a week, (iii) make decisions in real time (less than 100 ms), (iv) lead to only 7% less information amount gap than the optimal solutions, (v) control the used space within the allowed range, and (vi) scale to larger storage spaces.

2. Related work

Video storage server. Shao et al. [14] studied the correlation among the cameras at different locations to detect the abnormal behaviors, which was achieved by building a risk table. Each clip was determined to be deleted, partially deleted, or kept. Unfortunately, the strategy of making such decisions were not detailed in their paper. Usman et al. [15] proposed an intrusion-driven model, which encodes video clips with different encoding parameters. They assumed the video clips can only be downsampled once. Thomas et al. [16] turned activity, saliency, and collision levels into a cost function for their algorithm. The video clips were indexed with several keyframes to reduce the storage space, and end users could query video clips by comparing the similarity of frames using a neural network. Although they attempted to condense the video clips, the diverse levels of importance levels of different video clips were not considered in their solution. *These three studies [14–16] are quite different from our work because neither of them propose systematic approaches to: (i) quantify the information amount, nor (ii) decide the downsampling approaches and parameters.*

Video analytics. Video analytics applications were considered to be the *killer app* of edge computing [17] and attracted many researchers. For example, Satyanarayanan et al. [18] proposed a decentralized cloud computing paradigm, called *cloudlet*, leveraging edge servers. The videos from mobile devices were sent to cloudlets for analysis, while the analytics results were then sent to the cloud. Liu et al. [19] presented EdgeEye, which was an edge computing framework for real-time video analytics applications. Such applications could be offloaded by using their APIs. Chen et al. [20] presented a continuous, real-time object recognition system for mobile and wearable devices. They hid the network latency by caching video frames on mobile devices. The devices analyzed the cached frames using some hints sent by the server to localize the objects. Zhang et al. [21] presented Vigil, which was a real-time distributed wireless surveillance system leveraging edge computing. *The above mentioned papers [17–21] are orthogonal to our proposed storage server, as none of them considers the information amount.*

Video downsampling. Video downsampling is usually achieved by transcoding, which has been thoroughly studied in the literature. For example, Li et al. [22] proposed to transcode video clips in an on-demand manner to reduce the cloud resource consumption. Gao et al. [23] proposed a partial transcoding scheme for content management. They focused on the end-user viewing patterns and the operational cost on the content provider side. They formulated a cost minimization problem to find a balance between the storage cost and operational cost of real-time transcoding. Zakerinasab and Wang [24] proposed a distributed video transcoding scheme. The video segments were split into chunks and distributed to the cloud for parallel transcoding. Dutta et al. [25] proposed a scheme to transcode video clips in edge environments, in which the video clips were transcoded based on end-user expectations. Zhang et al. [26] offloaded the transcoding workload from mobile devices to the cloud. They designed an offloading policy for mobile devices and propose an online algorithm for the transcoding on the cloud to minimize the energy consumption while achieving low latency. Yoon et al. [27] moved the transcoding workload to the wireless edge, such as WiFi APs. They implemented a real-time video transcoder on a Raspberry Pi. *The above mentioned studies [22–27] are also orthogonal to our proposed storage server, as none of them accounts for the information loss.*

Video summarization. The concept of information amount in this article is partially inspired by previous work on video summarization. For example, Fu et al. [28] proposed to extract low- and high-level features from multi-view videos. They integrated the features with the Gaussian entropy fusion model to detect scene shots. After that, they constructed a spatial–temporal graph and clustered the scene shots by random walks. Their multi-objective optimization problem for summarization was based on the shot importance. Although their work considered high-level features like face recognition, they assumed boolean analytics results. Hence, their work could not differentiate the amounts of different high-level features. Our definition of information amount supports continuous weights to determine the diverse importance among analytics results. The weights can be adjusted based on the requirements of cameras at different places and orientations. Muhammad et al. [29] adopted memorability, entropy, and aesthetics into a score to select the keyframes for video summarization. These features were fused with weights; however, their solution depended on well-tuned parameters to cater end-user demands. Setting the weights manually was tedious, expensive, and error-prone. There exist other video summarization approaches [30,31] that removed redundancy in video clips to cut down the analysis time and storage space. Such decisions were mostly made purely based on the pixel values of the video clips without considering the high-level semantics that only exist in video analytics applications. *In contrast to video summarization work [28–31], we transform low-level features vectors to a different basis, which consists of principal components of the original feature vectors, so that the transformed features can better approximate the importance using our information amount concept.*

To the best of our knowledge, we are the first one to propose a storage server that determines the multiple quality levels of stored surveillance video clips coming from smart environments. We preserve the video clips that are more important to end users in better quality levels with longer achieved durations. By rigorously defining information amount based on multi-level features to quantify the importance, we make intelligent downsampling decisions. A preliminarily version of this article was published in our earlier paper [32].

3. Information amount of surveillance videos

We define *information amount* to quantify the importance of videos captured at different locations with diverse orientations. In information theory, entropy can be used to represent the information amount and feature vector complexity. Our definition of information amount combines the entropy of visual features (like color and edge histogram), and semantic features from the off-the-shelf analytics (like object detection). The core design rationale is that visual features will account for the information amount of video content even if we are unaware of actual analytics in the future. However, the amount of visual features may not directly capture the actual events that interest end users, which are rather complex. Such events are better captured by the semantic features, which are unfortunately computationally intensive to derive. Defining information amount purely based on semantic features is therefore too heavy for a resource-limited storage server. Hence, we leverage both visual and semantic features to define the information amount in the following.

Visual features. We observe that the viewports of surveillance videos are usually fixed, and thus the background is repeating and redundant across too many video frames. To reduce the duplicated computations, we pre-process the videos by removing the background and keeping the foreground objects. Particularly, we color the background of video frames in black, while keeping the color of foreground objects in the pre-processed videos. Next, we divide these pre-processed videos into several shots as follows. For each video frame, we compute the sum of the RGB values for each pixel. We then compute the difference of the RGB sums of individual pixels between two adjacent video frames. We consider a pixel is changed if its RGB sum difference is significantly different from its counterpart in the adjacent frame, i.e., the difference is beyond a tolerance. If the fraction of the changed pixels exceeds a threshold, we declare a new scene shot from the video frame. Moreover, to avoid noisy and trivial shots, we limit the minimal shot length. Notice that the abovementioned tolerance, threshold, and minimum length are all user-specified parameters, which are empirically determined to be 12, 95%, and 2 seconds respectively throughout this article, if not otherwise specified. Our background removal approach is inspired by Godbehere et al. [33]. Our shot detection algorithm is inspired by the black frame filter [34] with some augmentations for surveillance videos. This is because the difference of RGB sums between adjacent frames in the surveillance videos tend to be much smaller than generic, story-telling videos.

With pre-processed videos, we next define the visual features. We consider C new coming video clips to the storage server. For the shot S_c of clip c, we calculate its entropy using the set of features $F_1, F_2, \ldots, F_m$, where m is the considered number of features. Besides color and edge histogram features [28], we also adopt convolution and time-series features in this work.[1] We denote Shannon entropy as $E(\cdot)$. By transforming each feature into a one-column vector, we get the

[1] Our definition is general and works for other visual features, which could be global or local ones.

Table 1
Performance comparison of dimensionality reduction methods.

	PCA	Isomap	GRP	SRP	t-SNE	MDS
Residual variance	7.30×10^{-4}	7.78×10^{-4}	1.50×10^{-3}	7.06×10^{-2}	4.83×10^{-1}	1.45×10^{-2}
Trustworthiness	0.999925	0.999901	0.999688	0.987477	0.963822	0.959152
Continuity	0.999973	0.999955	0.999686	0.987920	0.971213	0.915735
Running time (s)	1.14×10^{-2}	3.55×10^{-1}	5.99×10^{-3}	8.71×10^{-3}	61.02	1.41

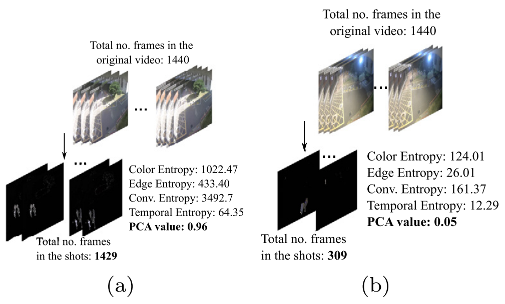

Fig. 3. Sample visual importance scores based on the PCA results from real video clips: (a) video with longer shots and higher entropy in daytime and (b) video with shorter shots and lower entropy at midnight.

entropy $E(F_i)$ of the feature F_i, where $i = 1, 2, \dots, m$. To quantify the importance of visual features, we determine a single score for each shot. One naive approach is to define a weighted function; however, it is difficult to specify appropriate weights without knowing the end-user demands in the future. Hence, a more systematic way is needed to determine the importance of individual visual features. To achieve that, we have compared six different Dimensionality Reduction (DR) methods: Principal Component Analysis (PCA) [35], Isometric mapping (Isomap) [36], t-distributed Stochastic Neighbor Embedding (t-SNE) [37], Multi-Dimensional Scaling (MDS) [38], Gaussian Random Projection (GRP), and Sparse Random Projection (SRP) [39] to reduce the feature dimensions from m to 1. We employ three widely-used and algorithm-independent quality metrics for evaluations: residual variance [36,40] (distance matrix based), trustworthiness, and continuity [39] (co-ranking matrix based). For distance matrix based metrics (residual variance), smaller is better; for co-ranking matrix based metrics (trustworthiness and continuity), larger is better. We note that all three metrics are in real numbers between 0 and 1. We capitalize an open-source Python library, pyDRMetrics [40], to implement the above DR methods and performance metrics. We run the experiments on a workstation with an Intel i7 CPU at 2.10 GHz, and also record the running time of different DR methods. We summarize the results in Table 1. We find that PCA outperforms other methods w.r.t all quality metrics: achieving smaller residual variance, higher trustworthiness, and higher continuity. Besides, all DR methods, except t-SNE and MDS, terminate in real-time. Hence, we chose PCA as our DR method in the rest of this article.

In PCA, the principle components are the eigenvectors of the covariance matrix of the visual features. We pick the eigenvector of the largest eigenvalue as the first component. The importance scores of these features are the projection on the transformed space. We feed m entropy values into PCA and generate a final score of visual features. The resulting visual importance score represents the shot and feature characteristics. Specifically, we write the visual importance of shot S_C as:

$$I_v(S_c) = P_1([E(F_1), E(F_2), \dots, E(F_m)]), \tag{1}$$

where $P_1(\cdot)$ is the transform function of PCA that considers the first principle component. In general, the larger $I_v(S)$ indicates more visual information embedded in video shot S. In Fig. 3, we show sample visual feature importance scores based on PCA results of actual videos captured in our smart campus testbed. Fig. 3(a) shows a daytime video that contains more frames in each shot. Its entropy across all features is also higher. In contrast, Fig. 3(b) reveals that the midnight video from the same camera has fewer frames in each shot, where each shot's entropy is also lower. Hence, the overall visual feature importance score of video in Fig. 3(a) is much higher than that of video in Fig. 3(b).

Semantic features. We decide to only extract semantic features from video shot S if $I_v(S) > \delta_v$, where δ_v is the threshold for entropy of visual features. This is to avoid wasting resources on trivial shots per our observations on some pilot tests. We assume A_c video analytics are needed for video clip c ($c \in [1, C]$). That is, each analytics a ($a \in [1, A_c]$) analyzes surveillance video clip c to detect certain *events*. We let x_a be the output of analytics a, where x_a can be either a discrete or a continuous value. Examples of discrete outputs include illegal parking (boolean) and queue length at a bus stop (integer); examples of continuous outputs include flood monitoring (depth) and fog detection (visibility). For analytics a with discrete outputs, we let n_a be the *normal* output. That is, if $x_a = n_a$, analytics a detects *no* event. For analytics a with continuous outputs, we define a *tolerance* level δ_a and consider no detected event if $|x_a - n_a| \le \delta_a$. Because the

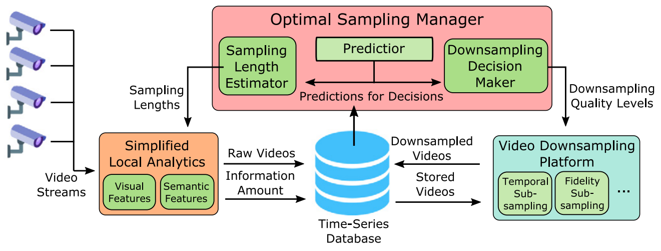

Fig. 4. System overview of our storage server in an edge network.

outputs of different analytics have diverse scales, x_a needs to be normalized. We let $\tilde{x}_a$ be the *maximal* absolute value[2] of the output from analytics a, and define the normalized information amount as:

$$e_{S_c,a} = \begin{cases} 0 & |x_a - n_a| \le \delta_a; \\ |x_a/\tilde{x}_a| & \text{otherwise}, \end{cases} \tag{2}$$

where δ_a is set to zero for analytics with discrete outputs. Generally, a larger $e_{S_c,a}$ value indicates that more semantic information is detected by analytics a in shot S_c.

The outputs of individual video analytics depend on the inputs, which are the surveillance video clips. We let $\mathbf{W}$ be a $C \times A$ matrix, where $\mathbf{W}_{c,a}$ ($c \in [1, C]$ and $a \in [1, A_c]$) represents the weight of analytics a on clip c. $\mathbf{W}_{c,a}$ is configurable by the system administrators for prioritizing different analytics and clips. With the symbols defined above, we formally write the semantic importance score of shot S_c in clip c as:

$$I_e(S_c) = \sum_{a \in A_c} \mathbf{W}_{c,a} \cdot e_{S_c,a} \Big/ \sum_{a \in A_c} \mathbf{W}_{c,a}, \tag{3}$$

where the summations iterate through all analytics and clips, before being normalized with weights $\mathbf{W}_{c,a}$.

To fuse the visual and semantic importance scores, we carry out min–max normalization where the range is derived from all samples throughout evaluations. That is, we use $\hat{I}_v(\cdot)$ and $\hat{I}_e(\cdot)$ to denote the normalized importance score, respectively. Last, we use f_c to denote the total number of frames among all shots in a clip c ($c \in [1, C]$). We write the total information amount as:

$$I(c, f_c) = \sum_{S_c \in c} \hat{I}_v(S_c) + \sum_{\substack{\forall S'_c \in c, \\ I_v(S'_c) > \delta_v}} \hat{I}_e(S'_c), \tag{4}$$

where $I(c, f_c)$ is the quantified information amount of clip c, which drives the designs of our optimization algorithms developed in the following sections.

4. Design of storage management

Fig. 4 gives an overview on the design of our storage server in an edge network, which is detailed in this section.

4.1. Workflow

The surveillance cameras continuously stream coded videos to the storage server. The video streams are initially saved on the storage server at the maximal quality, i.e., with the full information amount. The stored video clips can be requested by *external analytics servers* due to demands from end users. Our storage server provides requested videos to the *analytics servers* to serve, which runs video analytics for end users. The design and implementation of analytics servers are out of the scope of this article. We add *simplified local analytics* to the storage server, so as to extract visual and semantic features. We extract semantic features by applying video analytics that may be deployed on the external analytics servers. We employ a *sampling length estimator* to select the sampling length for each incoming video, in order to approximate the information amount without incurring excessively long running time. The videos in the full quality and information amount are sent to a time-series database for storage. We also add a *video downsampling platform* to the storage server, which supports multiple downsampling approaches. When the storage space is close to full, we downsample stored videos to free up some space for incoming video clips. We employ a *downsampling decision maker*, which carefully considers the trade-off among the remaining information amount, freed storage space, and complexity of different downsampling approaches when making the decisions. Furthermore, we employ a *predictor* to keep track of available storage and computation resources

[2] Without loss of generality, we assume $\tilde{x}_a \neq 0$. Otherwise, analytics a is not worthy of being executed.

for timely executions of information amount analysis and video clip downsampling. The predictions serve as inputs of the algorithms in sampling length estimator and downsampling decision maker. This is to ensure that enough storage space is freed for new video clips before their arrivals. In summary, the three key components of our storage server are: the sampling length estimator, the downsampling decision maker, and the predictor. These three components are executed in parallel. We collectively refer to them as the *optimal sampling manager*.

4.2. Components

We summarize the components of Fig. 4 in the following:

- *Sampling length estimator*. It hosts an algorithm to decide the sampling length for each incoming video clip. The sampling lengths are used to approximate the information amount of individual video clips in a more efficient manner to address Challenge #1 in Section 1.
- *Downsampling decision maker*. It hosts an algorithm to determine the video quality level of each video clip that is about to be downsampled. By doing so, we control information loss due to video clip downsampling to address Challenge #2.
- *Predictor*. It supports online predictions of resource consumption and information amount, which are needed by the above two components to ensure on time completion of the video analytics and downsampling approaches. This is to address Challenge #3.
- *Simplified local analytics*. It hosts multiple video analytics, which are also run by external analytics servers. The analytics are also needed for semantic feature extraction.
- *Video downsampling platform*. It hosts multiple downsampling approaches, which free up storage space.
- *Configuration manager*. It is an interface for system administrators to control and monitor the storage server. For example, system administrators may add/remove video analytics and downsampling approaches, or visualize the video clips in the time-series database.

Among these, the sampling length estimator and downsampling decision maker are the most crucial components. Their performance directly affects the analyzing and downsampling time. Furthermore, the storage space efficiency also relies on the results given by the downsampling decision maker. In Sections 5 and 6, we introduce the sampling length estimator and downsampling decision maker in detail.

5. Sampling Length Estimator: SLE

In this section, we develop our algorithms for Sampling Length Estimator (SLE). The algorithms are responsible for estimating the information amounts of the incoming and stored video clips.

5.1. Notations

Building upon the notations defined in earlier sections, we formally write the information amount of all video clips ($\mathbf{C}$) with all frames ($\mathbf{F}$) in all shots and all video analytics ($\mathbf{A}$) as:

$$H(\mathbf{C},\mathbf{F},\mathbf{A}) = \sum_{c=1}^{\mathbf{C}} I(c, f_c), \tag{5}$$

where $I(c, f_c)$ denotes the information amount across frames of all shots of clip c without any sampling. For a given $\mathbf{C}$, $\mathbf{F}$, and $\mathbf{A}$, computing $H(\mathbf{C},\mathbf{F},\mathbf{A})$ using Eq. (5) is extremely time consuming as it *dictates* executing all analytics on too many frames. The storage server, unfortunately, is unlikely to be able to perform analysis in real-time, and an *approximation* of $H(\mathbf{C},\mathbf{F},\mathbf{A})$ through sampling for a shorter execution time is needed. The sampled frames, however, need to be carefully chosen, e.g., selecting a few *consecutive* frames may lead to biased information amount due to the temporal locality across these frames. To cope with this issue, we define sampling length l as follows. We select a frame from every l consecutive ones from each shot. For simplicity, the same l is applied to all shots of every video clip, because events with similar characteristics tend to have temporal locality. Additionally, to control the computational complexity, we carefully choose the lengths $(L_{c,a_1}, L_{c,a_2}, \dots)$ from a discrete set $\mathbf{L}_0 = \{0, 1, l_1, l_2, \dots, l_m\}$, where 0 and 1 represent skipping and analyzing all frames, respectively. We write all sampling lengths into $C \times A$ matrix $\mathbf{L}$, where $\mathbf{L}_{c,a}$ denotes the sampling length l of analytics a for shots in clip c. Note that $\mathbf{L}_{c,a} = 0$ implies that analytics a is not applied on clip c at all. We aggregately write $L_c = (L_{c,a_1}, L_{c,a_2}, \dots)$ for presentation. Sampling lengths allow us to trade off the computational complexity and information amount accuracy. The approximate information amount is therefore written as:

$$H'(\mathbf{L}) = \sum_{c=1}^{|\mathbf{C}|} I(c, L_c), \forall L_c \in \mathbf{L}_0. \tag{6}$$

Computing $H'(\mathbf{L})$ with Eq. (6) is less computationally intensive compared to computing $H(\mathbf{C}, \mathbf{F}, \mathbf{A})$ with Eq. (5). The challenge is to make $H'(\mathbf{L})$ as close to $H(\mathbf{C}, \mathbf{F}, \mathbf{A})$ as possible by carefully selecting the best $\mathbf{L}$. *This is the job of our sampling length estimator.*

Next, we define a few helper functions. While applying sampling length $l > 1$ reduces the execution time of analytics, doing so may lead to reduced information amount. We let $d(\cdot)$ be the *degradation factor* of a sampling length, which represents the fraction of sampled information amount from the unsampled information amount. Intuitively, larger l leads to shorter execution time but lower degradation factor. The degradation factor of applying analytics a on video clip c is denoted as $d(c, a, L_{c,a})$. Give the degradation factor, the information amount due to semantic features is written as:

$$I'_e(S_c) = \frac{\sum_{a \in A_c} \mathbf{W}_{c,a} \cdot \hat{e}(c, f_c, a) \cdot \frac{|S_c|}{|\sum_{S_c \in c} S_c|} \cdot d(c, a, L_{c,a})}{\sum_{a \in A_c} \mathbf{W}_{c,a}}, \tag{7}$$

where $\hat{e}(c, f_c, a)$ denotes the prediction of the information amount from unsampled videos ($l = 1$). We apply min–max normalization on $I'_e(S_c)$ to get $\hat{I}'_e(S_c)$.

Extracting visual features is much lighter-weight, and thus we decide not to perform sampling when computing the information amount due to visual features. That is, we directly use the score of visual features in Eq. (4) in the SLE. By adding up $\hat{I}_v(S_c)$ and $\hat{I}'_e(S_c)$, our approximate information amount $I(c, L_c)$ (after sampling) is:

$$I(c, L_c) = \sum_{S_c \in c} \hat{I}_v(S_c) + \sum_{\substack{\forall S'_c \in c, \\ I_v(S'_c) > \delta_v}} \hat{I}'_e(S_c). \tag{8}$$

5.2. Problem formulation

Owing to the limited resources, we invest more computing power on the clips and analytics with higher information amounts to have more accurate estimations on them. In SLE, the goal is to make approximate information amount $H'(\mathbf{L})$ as close to that of the full-quality video clips $H(\mathbf{C}, \mathbf{F}, \mathbf{A})$ as possible. In addition, the execution of analytics needs to be done within a time constraint δ_i. Our preliminary tests reveal that visual feature extraction can be done relatively fast, and thus we focus on the execution time due to semantic analytics. For better mathematical tractability, we measure the execution time of shot detection, background subtraction, and visual feature extraction in our pilot tests. We then sum these times up and use a constant to represent it. Besides, we let function $t(c, a)$ be the execution time per frame when executing analytics a on video clip c.

Lemma 5.1 (*Hardness*)**.** *Our sampling length estimation problem is NP-Hard.*

Proof. This can be shown by a reduction from Multiple Choice Knapsack Problem (MCKP) [41]. Given an instance of MCKP with n mutually disjoint classes and capacity k. The item j in class i has profit $p_{i,j}$ and weight $w_{i,j}$. By mapping: (i) n classes to c clips, (ii) capacity k to time limitation δ_i, (iii) item j in class i to the sampling lengths of analytics ($L_{c,a_1}, L_{c,a_2,\ldots}$), (iv) profit $p_{i,j}$ to information amount computed with Eq. (8), and (v) weight $w_{i,j}$ to $t(c, a) \cdot |L_{c,a}|$, we reduce MCKP to our problem in polynomial time. $\square$

We next write our sampling length estimator problem as:

$$\begin{aligned} &\min_{\mathbf{L}} \left(H(\mathbf{C}, \mathbf{F}, \mathbf{A}) - H'(\mathbf{L})\right) = \max_{\mathbf{L}} \left(H'(\mathbf{L})\right) \\ &\quad s.t. \quad \sum_{\forall c \in \mathbf{C}} \sum_{\forall a \in \mathbf{A}} (t(c, a) \cdot |L_{c,a}|) < \delta_i. \end{aligned} \tag{9}$$

We assume the information amount after sampling is no larger than the full-quality video clips. By solving the formulation in Eq. (9), we maximize the approximated information amount $H'(\mathbf{L})$.

5.3. Optimal Estimation (OE) algorithm

We propose an optimal estimation algorithm based on dynamic programming. Let $z(c, \delta)$ be the maximal information amount when considering the first c clips under the time constraint δ. The state of this recursion is written as:

$$z(c, \delta) = \max \left(z(c-1, \delta - \sum_{\forall l_{c,a} \in l_j} t(c, a) \cdot l_{c,a}) + I(c, l_j) \right), \forall l_j \in \mathbf{L}_0, \tag{10}$$

where $z(0, \delta) = 0$ ($\forall \delta < \delta_i$) and $z(c, \delta) = -\infty$ if $\delta \leq 0$. We iterate from the tightest time constraint and the first clip. The optimal solution z^* is found when $z^* = z(|\mathbf{C}|, \delta_i)$. With dynamic programming, we store the computed results of increasingly more states to avoid redundant calculations. The next lemma analyzes the complexity of the OE algorithm, which is straightforward.

Lemma 5.2 (*Complexity*)**.** *Our OE algorithm has a space complexity of $\mathcal{O}(\delta_i \cdot |\mathbf{C}|^2)$. For the time complexity, each iteration computes Eq. (10) for $\mathcal{O}(|\mathbf{L}_0|)$ times. Thus, its total time complexity is $\mathcal{O}(|\mathbf{L}_0| \cdot \delta_i \cdot |\mathbf{C}|)$.*

5.4. Approximated Estimation (AE) algorithm

Algorithm 1 Approximate Estimation (AE) Algorithm for the SLE Problem

```
Inputs: Clips $\mathbf{C}$, Deadline $\delta_i$, Approximate Sampling Lengths $\mathbf{L}_0$, and Predictor $\hat{e}(\cdot)$
Output: Approximate Sampling Matrix $\mathbf{L}_x$.
1: Let $B_l = \max_{\forall c \in \mathbf{C},\ l_j \in \mathbf{L}_0} (I(c, l_j))$, $B_u = |\mathbf{C}| \cdot B_l$, and $\epsilon = 0.6$
2: $x = B_u \,/\, 2$
3: $\mathbf{J} = \emptyset$
4: for $c \in \mathbf{C}$ do
5:     $\{\|I\|\} = \frac{I(c,\ l_j)}{\sum_{\forall a \in A_c}(t(c,a) \cdot l_{c,a})}$, $\forall l_k \in \mathbf{L}_0 \ \cap\ \|I\| > \frac{0.8x}{\delta_i}$
6:     $l_k = \arg\max_{l_j}(\|I\|)$
7:     $\mathbf{J} = \mathbf{J} \cup \{(c, l_k)\}$
8: $\|\mathbf{J}\| = \sum_{\forall (c,l_k) \in \mathbf{J}} I(c, l_k)$
9: if $\|\mathbf{J}\| \le 0.8x$ then
10:     $B_u = x(1+\epsilon) = 0.8B_u$
11: else
12:     $B_l = x(1-\epsilon) = 0.2B_u$
13: if $B_u \,/\, B_L \le 5$ then
14:     Construct $\mathbf{L}_x$ by $\mathbf{J}$
15:     return $\mathbf{L}_x$
16: else
17:     $x = B_u \,/\, 2$ and go to line 3
```

OE has a pseudo-polynomial running time. We next propose a polynomial-time Approximation Estimation (AE) algorithm. The AE algorithm is inspired by a binary-search based algorithm [42], which is based on a *branching algorithm*. Given an $x > 0$ and an $\epsilon > 0$, the *branching algorithm* determines whether the optimal solution $z^* < x(1+\epsilon)$ or $z^* > x(1-\epsilon)$ exists. Algorithm 1 gives the pseudocode of the AE algorithm. We initialize x, upper bound B_u, and lower bound B_l in lines 1 and 2. We apply the default $\epsilon = 0.6$ [42], making the $z^* / z^0 \le 5$, where z^0 is the approximate solution. In lines 3–12, for each clip, we first add the array of lengths with maximal information, yet still meeting the constraint $0.8x \,/\, \delta_i$ into $\mathbf{J}$. Then we calculate the summation of information amount of the selected arrays in $\mathbf{J}$. If the summation is less than $0.8x$, the algorithm results in $z^* > x(1-\epsilon)$; otherwise, we have $z^* < x(1+\epsilon)$ [42]. Therefore, we update B_u and B_l accordingly. Last, in lines 13–17, we check the ratio between B_u and B_l to decide whether we keep iterating or return the approximation sampling length matrix L_x derived from $\mathbf{J}$. We note that Gens and Levener [42] is only one of the approximation algorithms for MCKP. There are other approximation algorithms [41,43], which can also serve as a starting points of developing similar approximation algorithms. We leave that as our future work. Our next lemma reports the complexity and performance bound of the AE algorithm; readers interested in its proof are referred to Gens and Levener [42].

Lemma 5.3 (*Complexity and Bound*)**.** *The AE algorithm gives the worst error bound of $\frac{4}{5}$ with respect to the optimal solution. Its running time is $\mathcal{O}(|\mathbf{C} \parallel \mathbf{L}_0| log|\mathbf{C}|)$, which is in polynomial time.*

5.5. Efficient Estimation (EE) algorithm

We propose a heuristic algorithm, called Efficient Estimation (EE) algorithm. This algorithm is based on an intuition: *the execution time and accuracy of information amount are both reduced once the sampling length is increased*. Algorithm 2 gives the pseudocode of the EE algorithm. In line 1, we initialize the sampling lengths $\mathbf{L}_c$ of all video clips and analytics to be 1 in $\mathbf{L}_0$. After doing so, if the total analytics execution time still fits into deadline δ_i, we return that $\mathbf{L}_e$, because it gives the most accurate information amount without any sampling. The while-loop starting from line 2 checks if the total execution time exceeds the deadline. If yes, we choose pair (c, a) that: (i) contains the least information amount and (ii) has a sampling length that can still be increased in line 3. We then increase the sampling length of (c, a) in line 7. When the total execution time reaches δ_i, we return sampling matrix $\mathbf{L}_e$ in line 8. The complexities of the proposed heuristic are given below. The proof is omitted as it is straightforward.

Lemma 5.4 (*Complexity*)**.** *The EE algorithm has a time complexity of $\mathcal{O}(\delta_i)$ and a space complexity of $\mathcal{O}(|\mathbf{C} \parallel \mathbf{A}|)$. Both are in polynomial.*

Algorithm 2 Efficient Estimation (EE) Algorithm for the SLE Problem

```
Inputs: Clips $\mathbf{C}$, Deadline $\delta_i$, Sampling Lengths $\mathbf{L}_0$, and Predictor $\hat{e}(\cdot)$
Output: Efficient Sampling Matrix $\mathbf{L}_e$.
1: Let $\mathbf{L}_c = 1, \forall c \in \mathbf{C}$
2: while $\sum_{\forall c \in \mathbf{C}} \sum_{\forall a \in \mathbf{A}} (t(c,a) \cdot |L_{c,a}|) > \delta_i$ do
3:     Find $(c,a)$ = $\arg\min_{(c,a)} \left( \mathbf{W}_{c,a} \cdot \hat{e}(c, f_c, a) \cdot \frac{|S_c|}{|\sum_{S_c \in c} S_c|} \cdot d(c,a,L_{c,a}) \cdot \frac{1}{t(c,a) \cdot |L_{c,a}|} \right), \ \forall \ \mathbf{L}_c \ \neq 0$
4:     if then$L_{c,a} = \max(\mathbf{L}_0)$
5:         $\mathbf{L}_c = 0$
6:     else
7:         Let $L_{c,a}$ of $\mathbf{L}_c$ be the next larger length in $\mathbf{L}_0$
8: Construct $L_e$ from selected sampling lengths
9: return $\mathbf{L}_e$
```

6. Downsampling Decision Maker: DDM

In this section, we develop our algorithms for the Downsampling Decision Maker (DDM). The algorithms are responsible for determining the quality levels of video clips to be preserved.

6.1. Notations

For concrete discussions, we introduce temporal and fidelity downsampling approaches, while spatial or other approaches can be readily adopted by our storage server. The first approach is *temporal* subsampling, which keeps the first frames of recurring non-overlapping time windows. For instance, temporal subsampling with a *frame-skip* of 4 means keeping 1 out of every 4 frames (i.e., frames $4k + 1 \ \forall k \in \mathbf{N}$). Any future video analytics on *deleted* video frame f are approximated with the immediately preceding frame that is kept, which is $4\lfloor \frac{f-1}{4} \rfloor + 1$ in this example. Secondly, the *fidelity* downsampling approach essentially transcodes the video clips with a lower bitrate. The resulting video clips have the same number of video frames, although their analytics outputs may be different from the ones from the original video clips.

We let $\mathbf{P}_0$ be all possible downsampling quality levels, where each approach specifies a frame rate and a bitrate. Downsampling decision maker generates the quality levels for all video clips which are stored on the storage server. Thus, we let $\mathbf{P}$ be a 1-dimensional downsampling decision matrix with a size of $|\mathbf{C}|$. $\mathbf{P}_c$ indicates the selected downsampling approach (e.g., 12 frame-per-second, or fps, at 500 kbps) of clip c. $\mathbf{P}_c = -1$ indicates storing the original quality of c on the storage server, while $\mathbf{P}_c = 0$ indicates deleting clip c from the storage server. The approximate information amount after taking downsampling decision $\mathbf{P}$ is:

$$H'(\mathbf{P}) = \sum_{c=1}^{|\mathbf{C}|} I(c, P_c). \tag{11}$$

Here, we overload the notation $I(\cdot)$: with a parameter of quality level P_c, $I(\cdot)$ denotes the summation of the information amount of *all frames* in clip c after downsampling with P_c. We assume that each video clip can be downsampled more than once with different decisions. The challenge is to maximize $H'(\mathbf{P})$ by carefully selecting the best $\mathbf{P}$. *This is the job of our downsampling decision maker.*

For the semantic features, we let $d'(c, a, P_c)$ be the degradation factor of information amount when applying analytic a on clip c at the quality level P_c. Note that different downsampling approaches affect the semantics feature extractions differently, and the information amount reduces as the quality gets lower. The symbol $\hat{I}''_e(S'_c)$ stands for the importance score of semantic features of shot S_c of clip c. We write the degraded semantic importance $I''_e(S_c)$ after downsampling as:

$$I''_e(S_c) = \sum_{a \in A_c} \mathbf{W}_{c,a} \cdot e_{c,a} \cdot d'(c, a, P_c) \Big/ \sum_{a \in A_c} \mathbf{W}_{c,a}, \tag{12}$$

where $e_{c,a}$ is the analytics results from simplified local analytics, which are conducted with the lengths decided by SLE. As for the visual features, we directly apply the score of visual features calculated in SLE. We approximate the information amount $I(c, P_c)$ after downsampling as follows:

$$I(c, P_c) = \sum_{S_c \in c} \hat{I}_v(S_c) + \sum_{\substack{\forall S'_c \in c, \\ I_v(S'_c) > \delta_v}} \hat{I}''_e(S'_c) \tag{13}$$

6.2. Problem formulation

Given that the storage space is limited, we aim to downsample the video clips with less per-unit-size information amount to free up enough storage space. The goal of DDM is to maximize the total information amount of all stored clips after downsampling $H'(\mathbf{P})$. The execution time and storage space are the two constraints of our problem. First, the downsampling time of clips is stateful, which means that the time is related to the quality levels before and after. Therefore, we write P'_c and P_c to denote the quality levels before and after downsampling. Both P'_c and P_c are from the pre-selected set $\mathbf{P}_0$. We let $t(c, P'_c, P_c)$ be the downsampling time of video clip c. All the video downsampling tasks must be done by a deadline δ_d. Second, we set watermarks to determine when to trigger downsampling. The DDM problem is solved once the used space of the storage server reaches the high watermark $O_{v'}$. Once the downsampling is triggered, the target used storage space is a low watermark O_v. $O_{v'}$, O_v, and δ_d are user-configurable parameters. Last, we denote the resulting used storage of clip c in quality P_c as $\hat{o}_{c,P_c}$

Lemma 6.1 (*Hardness*)**.** *Our downsampling decision making problem is NP-Hard.*

Proof. We show the reduction from the multidimensional Knapsack Problem (d-KP) [41]. The set of items is partitioned into n mutually disjoint classes and only one item is selected from every class. The weighted sums of items of feasible solutions must meet the constraints in k dimensions. We consider a 2-MCKP problem here. The item j in class i has profit $p_{i,j}$ with weights $w_{i,j,1}$ and $w_{i,j,2}$ in the two dimensions, respectively. By mapping: (i) class n to video clip c, (ii) δ_d to the first dimension constraint and O_v to the second one, (iii) item set in each class to pre-selected quality set P_0, (iv) profits $p_{i,j}$ of items to the information amount derived from Eqs. (12) and (13), and (v) weights $w_{i,j,1}$ to the downsampling time, $w_{i,j,2}$ to the resulting used space, we get a corresponding DDM problem in polynomial time. This yields the hardness proof. □

We write our downsampling decision maker problem in the following:

$$\begin{aligned} &\max_{\mathbf{P}} \left(H'(\mathbf{P})\right) = \max\left(\sum_{c=1}^{|\mathbf{C}|} I(c, P_c)\right) \\ s.t. \quad &\sum_{\forall c \in \mathbf{C}} t(c, P'_c, P_c) < \delta_d, \textit{ and} \sum_{\forall c \in \mathbf{C}} \hat{o}_{c,P_c} < O_v. \end{aligned} \tag{14}$$

Note that each video clip is allowed to be downsampled more than once. In addition, downsampling is a lossy procedure, which means that the drop of information amount is non-reversible. By solving Eq. (14), we generate the best $\mathbf{P}$ to preserve the most space-effective video clips.

6.3. Optimal Decision (OD) algorithm

We first enumerate the clips from the strictest space and time constraints. We then employ dynamic programming for the optimal decisions. We write $z'(c, o, \delta)$ as the maximal preserved information amount after conducting downsampling on the first c clips under the space and time constraints o and δ, with the selected downsampling qualities. The state update is given as:

$$z'(c, o, \delta) = \max\left(z'(c-1, o - \hat{o}_{c,p_j}, \delta - t(c, P'_c, p_j)) + I(c, p_j)\right) \forall p_j \in \mathbf{P}_0, \tag{15}$$

where $z'(0, o, \delta) = 0$ ($\forall \delta < \delta_d \wedge o < O_v$) and $z'(c, o, \delta) = -\infty$ if $\delta \leq 0 \wedge o \leq 0$. We reach the optimal solution when $z' = z'(|\mathbf{C}|, O_v, \delta_d)$. The next lemma analyzes the complexity of the OD algorithm.

Lemma 6.2 (*Complexity*)**.** *In our OD algorithm, each of the iteration runs Eq. (15) for $\mathcal{O}(|\mathbf{P}_0|)$ times. This leads to $\mathcal{O}(|\mathbf{C}| \cdot O_v \cdot \delta_d \cdot |\mathbf{P}_0|)$ operations in total. The selected quality of each clip needs to be saved while updating the states, so the space complexity is $\mathcal{O}(|\mathbf{C}|^2 \cdot O_v \cdot \delta_d)$.*

6.4. Approximation Decision (AD) algorithm

The OD algorithm has pseudo-polynomial complexities, and thus we propose an Approximation Decision (AD) algorithm in Algorithm 3. Starting from the branching algorithm [42], we first initialize the upper and lower bounds by the maximal information amount among all clips. We initialize an x by d, which is the number of constraints, as the criteria to add the pairs of clips and qualities to $\mathbf{J}$ in lines 2–7. We set the complexity coefficient $\hat{t}$ as 2. Each pair of clips has the maximal information amount per time and per space, respectively. Lines 9–12 check whether the information amount of the selected pair set $\|\mathbf{J}\|$ exceeds $\frac{1}{2d}x$, which triggers the updates of upper and lower bounds for the next round of the binary search. Once the approximation solution falls in the range in line 13, we construct the approximate downsampling decision matrix $\mathbf{P}_x$ from $\mathbf{J}$. Our algorithm is inspired by a binary search algorithm [44] for MMCKP [41]. We note that other approximation algorithms for MMCKP [45] may be leveraged for developing better approximation algorithms for our DDM problem. We leave that as a potential future task.

Algorithm 3 Approximate Decision (AD) Algorithm for the DDM Problem

Inputs: Information Amount I, Clips $\mathbf{C}$, Deadline δ_d, Approximate Downsampling Decision Matrix $\mathbf{P}_x$, Positive Integer $\hat{t}$.
Output: Approximate Downsampling Decision Matrix $\mathbf{P}_x$.

```
1: Let $B_l = \max_{\forall c \in \mathbf{C},\ p_k \in \mathbf{P}_0}(I(c, p_k))$, $B_{l_0} = B_l$, $B_u = |\mathbf{C}| \cdot B_l$, and $d = 2$
2: $x = \frac{d}{1+2d} B_u + B_l$
3: $\mathbf{J} = \emptyset$
4: for $c \in \mathbf{C}$ do
5:     $\{\|I\|\} = \frac{I(c,\ P_c)}{t(c, P_c', P_c)/\delta_d} + \frac{I(c,\ P_c)}{\hat{o}_{c,P_c}/O_v},\ \forall\ p_i \in \mathbf{P}_0\ \cap\ \|I\| > \frac{x}{d}$
6:     $p_k = \arg\max_{p_j}(\|I\|)$
7:     $\mathbf{J} = \mathbf{J} \cup \{(c, p_k)\}$
8: $\|\mathbf{J}\| = \sum_{\forall (c, p_k) \in \mathbf{J}} I(c, p_k)$
9: if $\|\mathbf{J}\| \leq \frac{1}{2d} x$ then
10:     $B_u = (1 + \frac{1}{2d}) B_u$
11: else
12:     $B_l = \frac{1}{2d} B_l$
13: if $B_u - (1+2d)B_l \leq (\frac{1}{2})^{\hat{t}} B_{l_0}$ then
14:     Construct $\mathbf{P}_x$ by $\mathbf{J}$
15:     return $\mathbf{P}_x$
16: else
17:     $x = \frac{1}{1+2d} B_u + dB_l$ and go to line 3
```

Lemma 6.3 (*Complexity and Bound*)**.** *In AD algorithm, each round of branching algorithm takes* $\mathcal{O}(|\mathbf{C}| \cdot |\mathbf{P}_0|)$, *where* $|\mathbf{P}_0|$ *can be seen as a constant because we preselected* $\mathbf{P}_0$. *There are at most* $\mathcal{O}(\hat{t} + log(|\mathbf{C}| - 2d))$ *iterations in the binary search algorithm. Thus, the total time complexity is* $\mathcal{O}(|\mathbf{C}|(\hat{t} + log(|\mathbf{C}| - 2d)))$, *which is polynomial time. An improved binary search algorithm finds the approximate solution with the ratio of* $1 + 2d + (\frac{1}{2})^{\hat{t}}$, *where d and* $\hat{t}$ *equals 2 in our case.*

Algorithm 4 Efficient Decision (ED) Algorithm for the DDM Problem

Inputs: Information Amount I, Weight $\mathbf{W}$, Deadline δ_d, Watermark O_v, Selected Sampling Length Matrix L, and Predictor $\hat{e}(\cdot)$.
Output: Efficient Downsampling Matrix P_e.

```
1: Let $\mathbf{P}_c = -1, \forall c \in C;\ S = \sum_{\forall c \in C} o_c;\ T = 0;$
2: while $S > O_v$ or $T > \delta_d$ do
3:     $c = \arg\min_{\forall c \in C, \mathbf{P}_c \neq 0} \left(I(c, L_c) / \hat{o}_{c,\mathbf{P}_c}\right)$
4:     $\hat{p} = \arg\min_{\forall \hat{o}_{c,\mathbf{P}_c} \geq \hat{o}_{c,\hat{p}}} (\hat{o}_{c,\mathbf{P}_c} - \hat{o}_{c,\hat{p}})$
5:     $S = S - \hat{o}_{c,\mathbf{P}_c} + \hat{o}_{c,\hat{p}};$
6:     $T = T - \hat{t}(c, \mathbf{P}_c) + \hat{t}(c, \hat{p});$
7:     $\mathbf{P}_c = \hat{p}$
8: Construct $P_e$ from the selected $\mathbf{P}_c$
9: return $\mathbf{P}_e$
```

6.5. *Efficient Decision (ED) algorithm*

We propose an heuristic algorithm, called Efficient Decision (ED) for the DDM problem. The algorithm is based on the following intuition: *the video clip with the smallest per-unit-size information amount should be sacrified first, while the degree of its downsampling approach should be kept as small as possible.* Algorithm 4 gives the pseudocode of the ED algorithm. Line 1 initializes the downsampling decisions of all video clips to be, i.e., keeping them as-is. Then we let S and T be the used storage space and total execution time, respectively. The while-loop starting from line 2 iterates as long as: (i) the used storage space is higher than the low watermark or (ii) the total execution time exceeds the deadline. The numerator in line 3 accounts for the information amount of clip c, and the denominator accounts for the overall size. That is, in line 3, we select clip c with the smallest per-unit-size information amount to be downsampled. In line 4, we choose the quality $\hat{p}$ which has the smallest difference of sampled size to gradually reduce the storage space usage to the low watermark O_v. In lines 5–7, we update $\mathbf{P}$, S, and T. We note that line 4 may end up with $\hat{p} = 0$, which indicates that clip c is going to be deleted. As long the low watermark and deadline constraints are met, we return the decision matrix $\mathbf{P}_e$ derived from the selected P_c in line 8. The complexities of the proposed heuristic are given below. The proof is trivial and omitted.

Lemma 6.4 (*Complexity*)**.** *The time complexity of the GD algorithm has a time complexity of* $\mathcal{O}(S - O_v + \delta_d)$ *and a space complexity of* $\mathcal{O}(C)$.

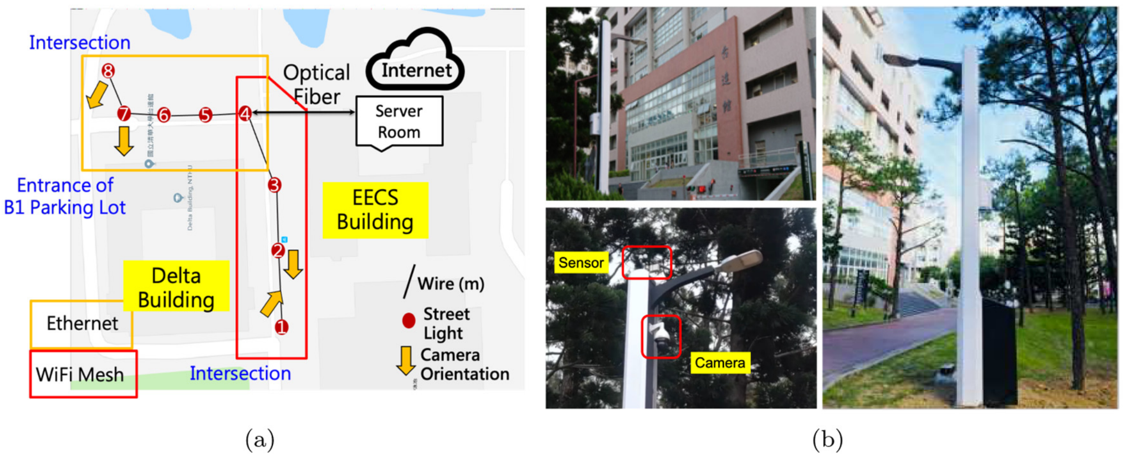

Fig. 5. The smart street lamp testbed at NTHU, Taiwan: (a) testbed topology and (b) sample photos.

Table 2
Sample prediction table.

Index	Sample values
Analytics	{People counting, Illegal parking, ...}
Quality level	{(12 fps, 500 kbps), (6 fps, 10 kbps), ...}
Day-of-the-week	{Weekday, Weekend}
Time-of-the-day	{0, 1, ..., 23}

7. Implementations

In this section, we first present a real implementation of our proposed storage server and a smart campus deployment. This is followed by one of the possible predictor design and its implementation.

7.1. Testbed implementation

We have implemented the proposed storage server on our smart street lamp testbed at NTHU, Taiwan, which consists of eight street lamps, as shown in Fig. 5. The street lamps are equipped with several sensors, including air-quality, temperature, humidity, wind speed, and motion sensors. Four of the street lamps come with IP cameras: three fixed bullet cameras and one PTZ (Pan-Tilt-Zoom) camera. The street lamps are interconnected by a mixture of Gigabit Ethernet and WiFi mesh networks. Some of the street lamps hold analytics servers, which can be compact PCs, like Intel NUCs, or single-board computers, like Raspberry Pis. Fig. 2(a) reveals two sample analytics analyzing the surveillance video clips.

Our implementation is summarized in Fig. 6, which complies with the design in Fig. 4. Most implemented components belong to the storage server, which is written in Python. We leverage: (i) InfluxDB [46] to realize the time-series database, (ii) Darknet [47] to realize the simplified local analytics platform and analytics servers, and (iii) FFmpeg [48] to realize the video downsampling platform. We implement the optimal downsampling manager in a *modularized* manner, where different components communicate using the socket API. This allows us to readily replace various algorithms in our storage server for comparisons. While our storage server *can*: (i) receive surveillance video streams from the four IP cameras on our smart street lamps and (ii) serve queries through analytics servers from actual users, doing so results in an unpredictable *workload*, which may prevent us from fairly comparing different storage server designs. To cope with this limitation, we also implement: (i) a *virtual camera*, which emulates a camera in our testbed by replaying the surveillance videos and (ii) a *query generator*, which emulates the requests from analytic servers. The two components allow us to impose *exactly* the same workload on different storage server designs.

7.2. Predictor

The resource consumption of video analytics and downsampling approaches depends on: (i) analytics/downsampling types, (ii) analytics/downsampling parameters (i.e., **L** and **P**), and (iii) *context information*. Context information refers to external information that serves as *hints* for more accurate predictions. Examples of context information include the weather (fewer people taking buses in rainy days), the day-of-the-week (campus buses are busier on weekdays), the time-of-the-day (fewer pedestrians crossing an intersection in late evenings), and parking lot occupancy (illegal parking are more likely when the lots are more full). We let function $t(c, a)$ be the per-frame execution time when analyzing

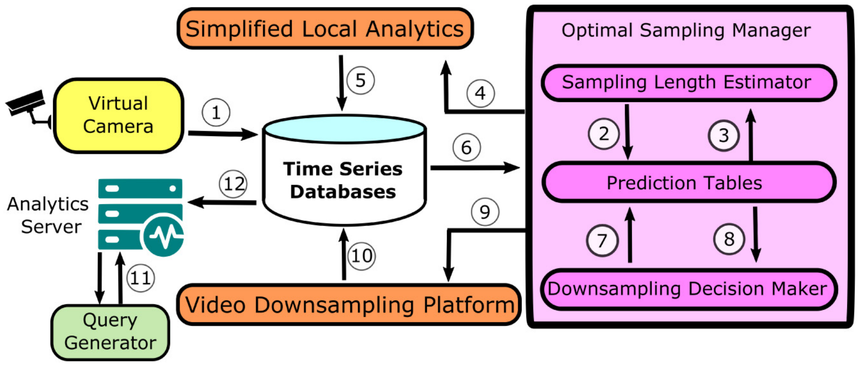

Fig. 6. Testbed consisting of our implemented storage server, an analytics server, a virtual camera, and a query generator. The circled numbers indicate the order of the operations.

sampling frames of video clip c using analytics a. The context information of c is stored in the same time-series database along with c itself. Similarly, we use $t(c, \mathbf{P}'_c, \mathbf{P}_c)$ to represent the downsampling time of approach d to downsample clips from quality $\mathbf{P}'_c$ to $\mathbf{P}_c$ on video clip c.

There are several ways of predicting the resource consumptions under different parameters and context. One possibility is to adopt *general* regression models with *arbitrarily chosen* parameters under *diverse* context. Doing so, however, requires too many samples to train the regression models, which is fairly expensive. We argue that the parameters need *not* be arbitrarily chosen because they are determined by the administrators of the storage server. In other words, as long as our predictions are accurate with a few *pre-selected* parameter values, the storage server will work as good as, if not better than, having general regression models. Hence, we propose to employ lookup tables indexed by analytics/downsampling decisions and context. The tables are built and updated with online samples to accommodate the environmental dynamics. Whenever there is a ground-truth sample coming from the video downsampling platform or analytics servers, we update the sliding window and the tables. When a prediction is needed, the values in the lookup tables are returned. When a cell is not populated (is empty), we use the value in the closest cell (in the sense of context) for prediction. Following the same design rationale of the resource prediction, we build an information amount table $\hat{e}(\cdot)$ indexed by analytics, downsampling decision, and context. We give a sample lookup table in Table 2. We estimate $\hat{e}_{c,f_c,a}$ and update $\hat{e}(\cdot)$ with the moving averages, which in turn allows us to derive $H'(\mathbf{L})$, and $H'(\mathbf{P})$ using Eqs. (6) and (11), respectively. Last, we note that there exists a large design space for more comprehensive predictors, which could be data-driven and machine learning based. Real deployments of our proposed storage server may be readily incorporated with these comprehensive predictors for better performance; however, we conduct our evaluations with the predictors based on lookup tables to be conservative.

8. Evaluations

In this section, we first introduce the evaluation metrics, baselines, and environment setup of our trace-driven simulation. Next, we evaluate the effectiveness, efficiency, practicality, and scalability of our proposed algorithms for the SLE and DDM problems.

8.1. Setup

For fair evaluations, we record the video clips from an IP camera facing a major intersection on our campus in November 2020.[3] More precisely, the video clips are encoded in HEVC codec [49] with a resolution of 2048 × 1536. Each video clip lasts for 1 h at 24 fps and 1 Mbps. We conduct the evaluations on a Linux server with an Nvidia GeForce GTX 1080Ti GPU running our storage server. We report the sample evaluation results from the second week (9th to 15th; Monday to Sunday) of November; while results from other weeks are similar. Specifically, we replay the video clips for seven days and generate random queries on the last day (Sunday) for analytics on the surveillance video clips captured in these seven days as follows. We first let λ represent the average number of requests per day; if not otherwise specified, we let $\lambda = 8$. The inter-arrival time of the query events is generated with an unit of *hour* following a Poisson process. For each query event, we randomly select an analytics using the uniform distribution. For statistically meaningful results, we repeat the simulations six times with different random seeds, and report the results from a sample run and across all six runs.

For comparisons, we have also implemented the following algorithms to mimic the current practices:

3 We have made the dataset available upon request after anonymization. Please see our webpage (/https://nmsl.cs.nthu.edu.tw/projects//) for the instructions of getting access to the dataset.

- **Equal-Fidelity (EF)** downsamples the old video clips to a pre-defined bitrate (500 kbps if not otherwise specified) while keeping the same frames when more storage space is needed. EF only downsamples each video clip once, i.e., a previously downsampled video clip is deleted when EF is invoked again.
- **Equal-Frame-Rate (EFR)** is similar to EF, except that it downsamples the old video clips to a pre-defined frame rate (6 fps if not otherwise specified). The bitrate is not changed.
- **First-In-First-Out (FIFO)** always deletes the oldest video clips to free disk space for incoming video clips.

We consider the following performance metrics:

- **Information amount**. We report the estimated information amount over time when the algorithms for the SLE problem are triggered. Besides, we evaluate the total information amount of our storage server as the time advances. Generally speaking, more preserved information amounts lead to the lower chances for end-user queries to fail in the future.
- **Used storage space**. More storage space is used when increasingly more surveillance video clips are stored at the storage server. Our storage server controls the used storage space based on the high/low watermarks. We evaluate the algorithms for the DDM problem by measuring the dynamics of the used space.
- **Number of the stored video clips**. More video clips saved on the storage server offer better chances for satisfying queries from end users. This is because, as long as a video clip is not completely deleted, we can still apply some analytics on it even though its quality level may be low.
- **Running time of the algorithms**. Processing streamed videos requires real-time decision making, or the whole storage server may get stuck. We compare the running time of the algorithms solving the SLE and DDM problems.
- **Per-query information amount error**. We refer to the information amount gap between the full and downsampled video clips as the per-query information amount error. It can be readily computed using Eqs. (8) and (13).

We compare the performance of different algorithms under the following parameters. We let the sampling lengths $\mathbf{L}_0 = \{1, 24, 48, 96, 144\}$. We empirically choose the following downsampling decisions: $\mathbf{P}_0 = \{(24, 1000), (24, 500), (12, 500), (12, 100), (6, 100), (6, 10), (1, 10)\}$, where the first element is the frame rate in fps and the second element is the bitrate in kbps. We note that some combinations of the frame rate and bitrate are not included, because FFmpeg fails to transcode the video clip to those combinations, e.g., 10 kbps is clearly too low for a 24 fps clip. Moreover, the high/low watermarks are set to be 100% and 50% of the storage space in our experiments. We let $\delta_i = \delta_d = 6$ hours because we aim to complete all the analytics and downsampling tasks before the next decision is made. Due to the space limit, we report the sample results with the following two parameters in our evaluations: (i) while our proposed storage server supports weights, we use the unit weight matrix. (ii) we let the threshold for the entropy of visual features $\delta_v = 0$, which means all the detected shots are included when deriving the semantic features of the video clips. We consider two sample analytics (illegal parking and people counting) throughout our simulations; while two *unknown* analytics (illegal parking at a different parking spot and car counting) are added in the later experiments to evaluate the efficiency of the included visual features. Due to our sample analytics aiming to detect people and cars existing or not, we set $n_a = \delta_a = 0$ as the normal output and tolerance level respectively. Furthermore, we vary the storage space size $O \in \{\mathbf{20}, 40, 80\}$ GB to understand how it affects the overall performance. For the sake of fairness and clarify, we adopt information amount estimated by OE when evaluating algorithms for a DDM problem. Similar simulation results with the same trends are observed, when the information amounts are estimated by EE or AE. Last, we note that some of our proposed algorithms employ discretized variables, which can be in different granularity levels. Finer granularity leads to more precise free space management at the cost of longer running time. To understand such a trade-off, we compare two granularity levels: MB and GB in the evaluations. We include 95% confidence intervals as error bars whenever applicable.

8.2. Results

Our EE algorithm efficiently solves the SLE problem Figs. 7(a) and 7(d) plot the estimated information amounts from sampled analytics on weekdays and the weekend, respectively. These two figures reveal that more information amounts are captured during daytime of weekdays, while the midnight on the weekend results in hardly any information amount. More importantly, our EE algorithm effectively estimates the sampling lengths for analyzing the videos. The resulting information amount of EE is less than that of OE by only 7% on weekdays and 9% on weekends at most. Besides, our EE beats AE by a large margin, especially at the peak hours. Figs. 7(b) and 7(e) report the running time of the analytics execution time, which is well controlled within the deadline (shown by the dashed lines) when applying the sampling lengths. The running time of three estimation algorithms are illustrated in Figs. 7(c) and 7(f).[4] Because the difference between OE and EE/AE is too large, so we plot the figures with log-scale Y-axes. We observe that our EE algorithm runs in real time on both weekdays and weekends. In summary, our proposed EE algorithm runs in real-time, yet achieves near optimal performance, better than the optimal OE. In contrast, although the approximation algorithm AE comes with a provable bound, it results in inferior performance compared to the EE algorithm. *Hence, we conclude that the EE algorithm efficiently solves the SLE problem.*

[4] We notice that AE does not fully utilize all the execution time before the deadline. However, after carefully comparing the information amount of AE compared to that without any sampling, we have found that the error bounds of AE ($H(C, F, A)/H'(L)$) are merely 3.17 and 3.93 on the weekdays and weekend, respectively. We note that these empirical error bounds are below the theoretical ones mentioned in Lemma 5.3.

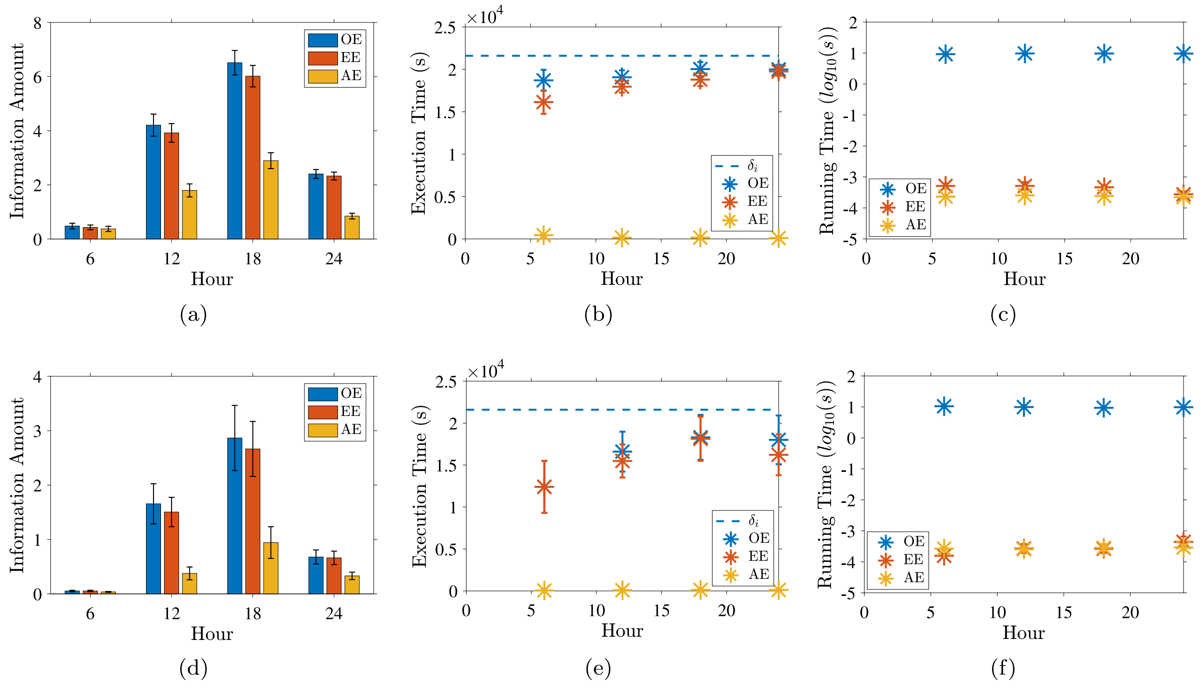

Fig. 7. Comparisons of different algorithms for the SLE problem: (a)–(c) on weekdays; (d)–(f) over the weekend. (a) and (d) are the information amount; (b) and (e) are the execution time of sampled analytics; (c) and (f) are the algorithms' running time.

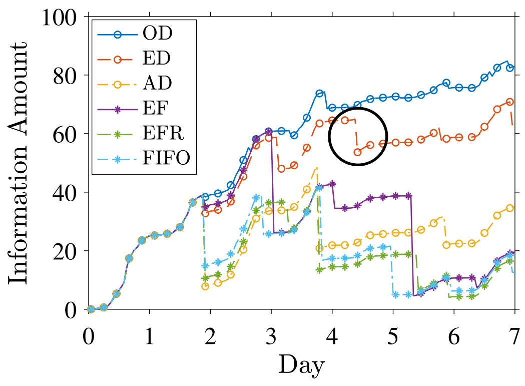

Fig. 8. Comparisons of downsampling decision algorithms: preserved information amount. Results from GB granularity are shown.

Table 3
Running time of downsampling decision algorithms with different granularity levels (Unit: s).

Algorithm	MB	GB
OD	N/A	3.32×10^2 ($\pm 1.95 \times 10^1$)
ED	8.85×10^{-2} ($\pm 1.26 \times 10^{-3}$)	1.25×10^{-2} ($\pm 2.72 \times 10^{-3}$)
AD	2.06×10^{-3} ($\pm 2.28 \times 10^{-4}$)	1.81×10^{-3} ($\pm 2.31 \times 10^{-4}$)
EF	1.30×10^{-3} ($\pm 2.58 \times 10^{-3}$)	1.00×10^{-3} ($\pm 2.64 \times 10^{-5}$)
EFR	8.12×10^{-4} ($\pm 9.02 \times 10^{-5}$)	9.13×10^{-4} ($\pm 1.05 \times 10^{-4}$)
FIFO	5.26×10^{-4} ($\pm 1.89 \times 10^{-5}$)	4.95×10^{-4} ($\pm 5.13 \times 10^{-6}$)

Our ED algorithm preserves more information amount. We report the preserved information amount on our storage server over a week. Fig. 8 gives the results from GB granularity level. We make three observations on this figure. First, the first downsampling decisions of all algorithms are made right before day 2 where EF, EFR, and FIFO suffer from a large drop on the information amount, while OD, ED, EF lead to little loss. Further, only OD and ED effectively preserve information amount in the remaining days. Second, the growth of information amount from day 1 to 5 (weekdays) is higher than that of the last two days (weekends). Third, on day 7, the overall information amount saved by our ED algorithm significantly

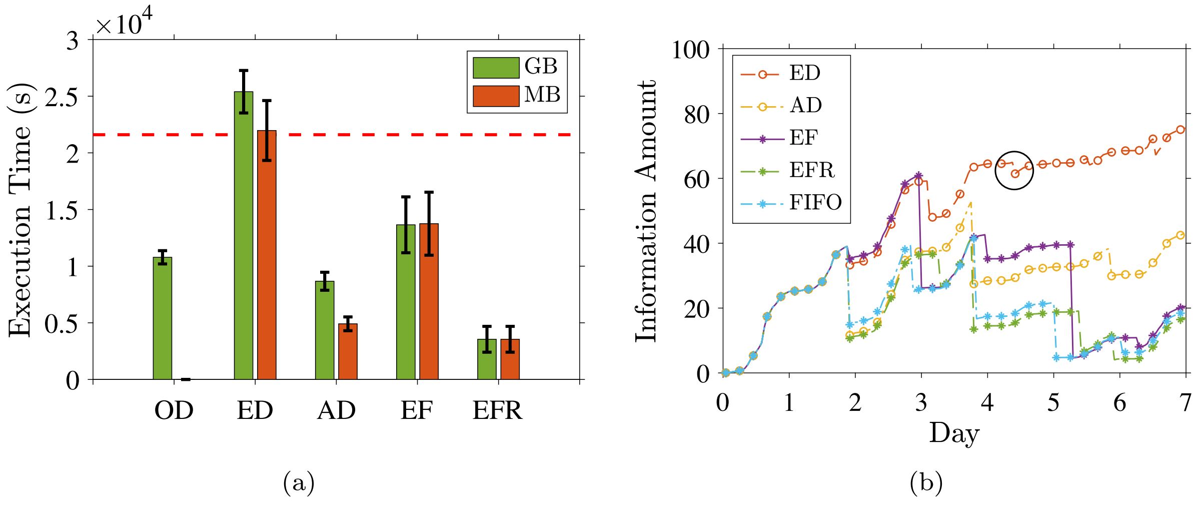

Fig. 9. Comparisons of downsampling decision algorithms: (a) execution time of the video sampling tasks and (b) preserved information amount under MB granularity.

outperforms all other alternatives. For example, EE outperforms AD by 44% and EF by 69%. *In summary, the ED algorithm efficiently addresses the DDM problem and preserves more information amount over time.*

Our ED algorithm runs in real time with fine granularity level. Table 3 reports the average running time of the algorithms for the DDM problem with 95% confidence intervals. The table shows that ED runs in real-time, always less than 89 ms. It is negligible compared to the execution time of the downsampling tasks, which is in the order of minutes if not hours. OD has an extremely large execution time, even with a coarse granularity level of GB. In fact, we ran the OD algorithm for a week without getting the optimal solution[5]; hence, we put Not-Applicable (N/A) in that table cell. Besides, because the size of a video clip is about 400 MB in our testbed, taking GB as the granularity level leads to high errors when making the downsampling decisions. For instance, Fig. 9(a) shows that with GB granularity level, ED results in excessive downsampling time, actually exceeding the downsampling deadline marked as the dashed line. A closer look indicates that the video codec cannot complete all downsampling tasks because of the prediction error on execution time due to coarse (GB) granularity. A similar issue also occurs on the prediction of information amount. We observe that ED does not always lead to the best performance compared to OD in various aspects. When the computing resource is very scarce, AD may be more suitable than ED, because the downsampling time of AD is only 34% of ED as shown in Fig. 9(a). Hence, system administrators could adopt different algorithms depending on their available computing resources. Fig. 9(b) gives the total information amount from different algorithms for the DDM problem. Compared to Fig. 8, we note that ED does not suffer from a sudden drop of the total information amount on day 4 as highlighted by the circles. Another observation we can make out of Fig. 9(b) is that ED significantly outperforms AD, even though the latter algorithm comes with a provable bound. *In summary, we find that our proposed ED algorithm runs in real-time and performs well with fine (MB) granularity level.* In contrast, the optimal OD algorithm fails to scale to fine granularity level, and thus is less practical. Hence, we do not report results from OD in the rest of the article. In addition, results from MB granularity level are given, if not otherwise specified.

Our storage server is effective on used space management yet results in low per-query errors. Fig. 10(a) plots sample results of used space from days 1 to 2 and Fig. 10(c) plots those from days 5 to 6. Results from other days are similar. In Fig. 10(a), the five lines overlap with one another until about 21:00 on day 1. This is because the used storage space has not reached the high watermark. These two figures show that: (i) the algorithms for the DDM problem are invoked once the used storage space reaches the high watermark and (ii) the used storage space drops below the low watermark as designed. Moreover, our ED algorithm manages the used space quite well, as it stops right at the low watermark. Figs. 10(b) and 10(d) report the number of video clips saved on the storage server. Except for FIFO and EFR, all other algorithms result in an increasing number of saved video clips over the week. We observe that our ED deletes the least clips on both weekdays and weekends. For example, on day 5, ED removes 48% fewer clips than EF; ED saves 2.78 times more videos clips than FIFO on the last day. In Figs. 11(a)–11(c), we report the per-query information amount error. On weekdays, we first observe that EF, EFR, and FIFO lead to fairly large errors on almost all queries. The reason is that the queries are generated on day 7; at that time, the earlier video clips are already removed by these baseline algorithms. Our ED performs much better: its information amount error is 58% lower than that of FIFO on average. We notice that in the weekend, FIFO shows the smallest error. This is no surprise as the queries are on the most recent video

[5] We estimate at least 38 days are needed to get the optimal decision.

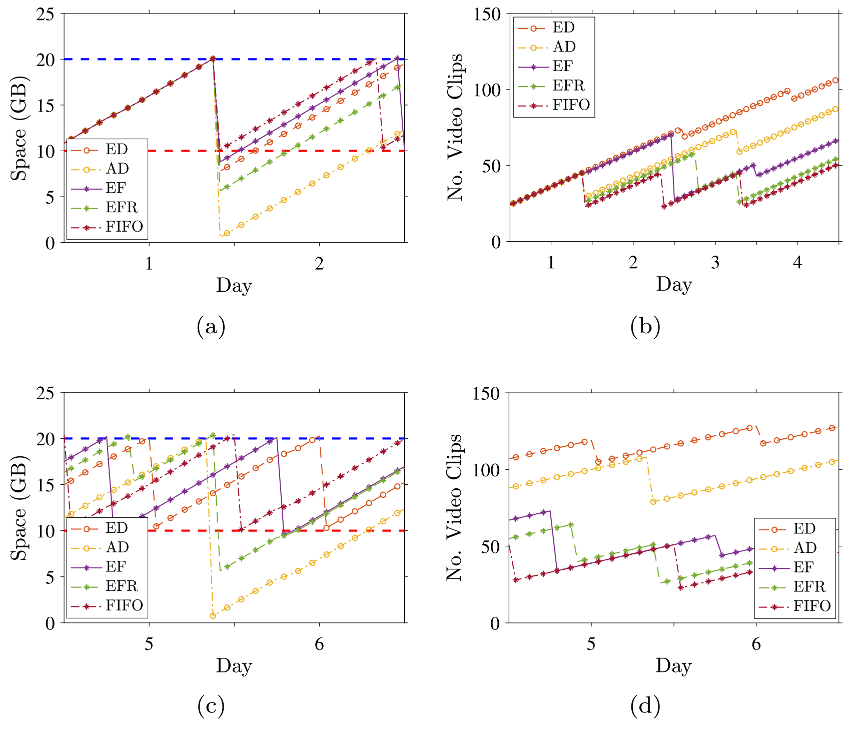

Fig. 10. Comparison of different algorithm for the DDM problem: (a)–(c) on weekdays; (b)–(d) over the weekend. (a) and (b) are the total used space of stored video clips; (b) and (d) are the number of stored video clips.

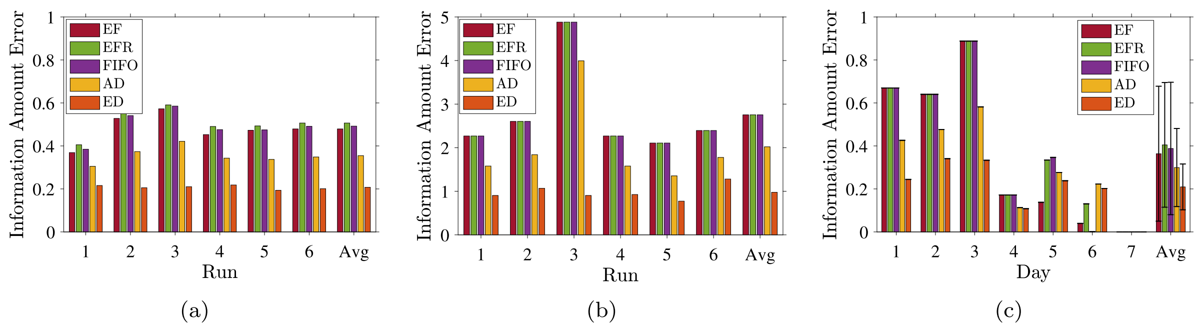

Fig. 11. Information amount error for queries: (a) average and (b) maximum errors across seven days in different runs; (c) average error of each day in seven days, sample results from run 1 are shown. In (c), note that all algorithms (except OD, which is not included) lead to zero error on day 7.

Table 4
Information amount error with and without visual features.

	Weekday	Weekend
With	9.77×10^{-2} ($\pm 1.14 \times 10^{-2}$)	2.60×10^{-2} ($\pm 5.85 \times 10^{-3}$)
Without	1.40×10^{-1} ($\pm 1.85 \times 10^{-2}$)	4.78×10^{-2} ($\pm 1.03 \times 10^{-2}$)

clips. Even in the worst case (day 6), our ED still achieves a small information amount error of 0.2, while other baseline algorithms could lead to almost 0.9 on some weekdays (like day 3). Last, we also note that our AD still outperforms the baseline algorithms (EF, EFR, and FIFO), although it has inferior performance to ED as shown in Fig. 11. A deeper look into Figs. 10(a), 10(b), and 11(c) reveals that EF, EFR, FIFO and AD suffer from two shortcomings. First, they tend to free too much used space, especially AD and EFR, and thus delete too many video clips. Second, even for those kept video clips, their quality is too low for analytics algorithms to make sense out of them. Hence, EF, EFR, FIFO, and AD do not perform well in terms of per-query error. *Taken all together, our ED not only well manages the used space and number of clips, but also successfully retains the information on both weekdays and weekends compared to other alternatives.*

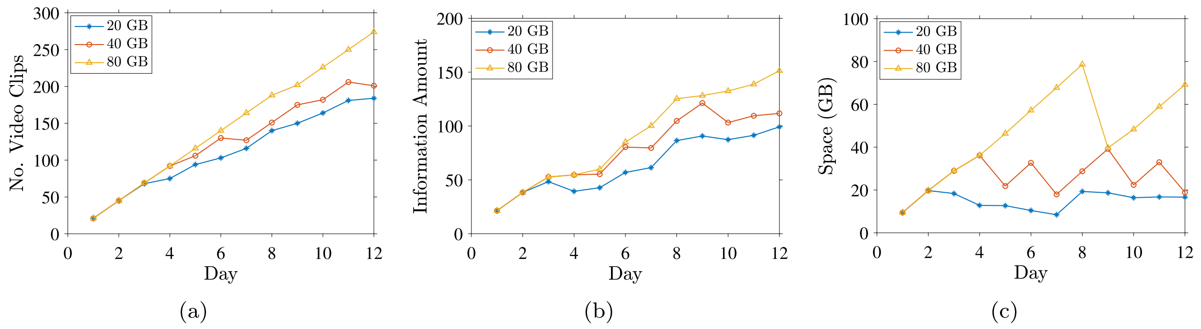

Fig. 12. Our ED algorithm scales well with larger storage spaces: (a) number of stored video clips, (b) total information amount, and (c) total used space.

Our storage server successfully supports unknown analytics. We define the information amount based on both semantic and visual features. The latter features are meant to cover the unknown analytics in the future. To understand their effectiveness, we next employ two unknown analytics in the queries on day 7. We then compute their errors of information amount with and without considering the visual features in our proposed ED algorithms. We give the results in Table 4. This table reveals that introducing (low-level) visual features reduces the information amount errors: 30% on weekdays and 46% on weekends, respectively. The reason is that visual features are more general across queries with heterogeneous analytics. *By considering visual features in the information amount, our storage server is capable of retaining important videos even if there are new analytics coming from end users.* Table 4 also shows that the errors on weekends are slightly lower than those on weekdays. This is as expected, because the quires are made closer to weekends (than weekdays).

Our ED algorithm scales well with larger storage spaces. Last, we consider different storage size: between 20 to 80 GB. To exercise the larger disks, we consider a longer 12-day trace between 9th and 20th of November. We report the per-day performance of our ED algorithm in Fig. 12. We observe that ED successfully capitalizes additional storage spaces: it saves more video clips and total information amount when the storage space is increased to 80 GB. In addition, the used space is well bounded between the high/low watermarks. *In summary, our ED algorithm scales well with larger storage spaces.*

9. Conclusion

In this article, we detailed the design, implementations, optimization, and evaluations of a multi-level feature driven storage server for surveillance videos gathered from smart environments. The design goal of the storage server is to retain as much information amount as possible under the constraints of storage space and computational power. We achieved the design goal in the following steps. We first carefully defined the information amount by extracting semantic and visual features from shots in the surveillance video clips. We proposed to use lookup tables for predicting resource consumption and information amount due to different analytics and downsampling approaches. Such predictions are capitalized to solve two key research problems: (i) sampling length estimation, which determines the sampling length to capture the information amount without overloading the storage server and (ii) downsampling decision maker, which selects a downsampling decision matrix to retain the most information amount without consuming excessive resources (both computation time and storage). We conducted extensive trace-driven simulations to compare the performance of our proposed algorithms in the system against the current practices. We found that our efficient algorithms (EE and ED) run faster (than the optimal algorithms OE and OD) and achieve much better performance (than the approximation algorithms AE and AD). The evaluation results demonstrate that our efficient algorithms:

1. achieve a mere $\sim 7\%$ captured information amount gap compared to the optimal solutions;
2. boost the number of saved video clips by up to 2.78 times;
3. reduce per-query error by $\sim 58\%$ on average;
4. terminate in 88 ms at most;
5. well-manage the used space between the high/low watermarks;
6. generate smaller errors on unknown analytics;
7. scale well with larger storage space.

Our proposed storage server can be extended in several directions. First, a cluster of geographically distributed storage servers could be jointly managed for even better performance. Second, quantifying the satisfaction levels of end-user queries is non-trivial. The information amount computed by our storage server may not directly reflect the user experience. Incorporating the concept of Quality-of-Experience [50] may be worth investigating. Third, more comprehensive predictors, such as those built upon Reinforcement-Learning (RL) [51] may be adopted; while the implications of more accurate predictions on the performance storage server can be quantified. Fourth, a wider array of analytics applications for diverse scenarios, such as infrastructure monitoring and smart agriculture, may be considered, where the information overlaps among these analytics applications can be investigated and potentially leveraged in the storage server design. Last, sensors other than surveillance cameras may also leverage our proposed solution, where audio and time-series data can be efficiently stored on edge servers.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was partially supported by the Ministry of Science and Technology of Taiwan under the grant: #107-2221-E 007-091-MY3 and by a NOVATEK Fellowship.

References

[1] N. Funde, P. Paranjape, K. Ram, P. Magde, M. Dhabu, Object detection and tracking approaches for video surveillance over camera network, in: Proc. of IEEE International Conference on Advanced Computing & Communication Systems (ICACCS'19), Coimbatore, India, 2019, pp. 1171–1176.
[2] G. Chandan, A. Jain, H. Jain, et al., Real time object detection and tracking using Deep Learning and OpenCV, in: Proc. of IEEE International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 2018, pp. 1305–1308.
[3] R. Xu, S. Nikouei, Y. Chen, A. Polunchenko, S. Song, C. Deng, T. Faughnan, Real-time human objects tracking for smart surveillance at the edge, in: Proc. of IEEE International Conference on Communications (ICC'18), Kansas City, MO, 2018, pp. 1–6.
[4] P. Li, L. Prieto, D. Mery, P. Flynn, On low-resolution face recognition in the wild: Comparisons and new techniques, Trans. Inf. Forensics Secur. 14 (8) (2019) 2000–2012.
[5] P. Rasti, T. Uiboupin, S. Escalera, G. Anbarjafari, Convolutional neural network super resolution for face recognition in surveillance monitoring, in: Proc. of Springer International Conference on Articulated Motion and Deformable Objects, Springer, Palma de Mallorca, Spain, 2016, pp. 175–184.
[6] A. Prati, C. Shan, K. Wang, Sensors, vision and networks: From video surveillance to activity recognition and health monitoring, IOS Press J. Ambient Intell. Smart Environ. 11 (1) (2019) 5–22.
[7] J. Seo, S. Han, S. Lee, H. Kim, Computer vision techniques for construction safety and health monitoring, Elsevier Adv. Eng. Inform. 29 (2) (2015) 239–251.
[8] F. Mehboob, M. Abbas, Intelligent video surveillance, in: A. Neves (Ed.), Video Surveillance-Based Intelligent Traffic Management in Smart Cities, IntechOpen, 2018, pp. 1–9.
[9] S. Javaid, A. Sufian, S. Pervaiz, M. Tanveer, Smart traffic management system using Internet of Things, in: Proc. of IEEE International Conference on Advanced Communication Technology (ICACT'18), Chuncheon, South Korea, 2018, pp. 393–398.
[10] M. Saifuzzaman, N. Moon, F. Nur, IoT based street lighting and traffic management system, in: Proc. of IEEE Region 10 Humanitarian Technology Conference (R10-HTC'17), Dhaka, Bangladesh, 2017, pp. 121–124.
[11] S. Dey, A. Chakraborty, S. Naskar, P. Misra, Smart city surveillance: Leveraging benefits of cloud data stores, in: Proc. of IEEE Conference on Local Computer Networks Workshops (LCNW'12), Clearwater, Florida, 2012, pp. 868–876.
[12] D. Rodríguez-Silva, L. Adkinson-Orellana, F. Gonz'lez-Castano, I. Armino-Franco, D. Gonz'lez-Martinez, Video surveillance based on cloud storage, in: Proc. of IEEE International Conference on Cloud Computing (CLOUD'12), Honolulu, Hawaii, 2012, pp. 991–992.
[13] S. Hossain, M. Hassan, M. Al, A. Alghamdi, Resource allocation for service composition in cloud-based video surveillance platform, in: Proc. of IEEE International Conference on Multimedia and Expo Workshops (ICMEW'12), Melbourne, Australia, 2012, pp. 408–412.
[14] Z. Shao, J. Cai, Z. Wang, Smart monitoring cameras driven intelligent processing to big surveillance video data, IEEE Trans. Big Data 4 (1) (2017) 105–116.
[15] A. Usman, R. Usman, S. Shin, An intrusion oriented heuristic for efficient resource management in end-to-end wireless video surveillance systems, in: Proc. of IEEE Annual Consumer Communications & Networking Conference (CCNC'18), Las Vegas, Nevada, 2018, pp. 1–6.
[16] S. Thomas, S. Gupta, K. Subramanian, Smart surveillance based on video summarization, in: Proc. of IEEE Region 10 Symposium (TENSYMP'17), Cochin, India, 2017, pp. 1–5.
[17] G. Ananthanarayanan, K. Chintalapudi, M. Philipose, L. Ravindranath, S. Sinha, Real-time video analytics: The killer app for edge computing, IEEE Comput. 50 (10) (2017) 58–67.
[18] M. Satyanarayanan, P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, W. Hu, B. Amos, Edge analytics in the internet of things, IEEE Pervasive Comput. 14 (2) (2015) 24–31.
[19] P. Liu, B. Qi, S. Banerjee, EdgeEye: An edge service framework for real-time intelligent video analytics, in: Proc. of International Workshop on Edge Systems, Analytics and Networking (EdgeSys'18), Munich, Germany, 2018, pp. 1–6.
[20] Y.-H. Chen, L. Ravindranath, S. Deng, P. Bahl, H. Balakrishnan, Glimpse: Continuous, real-time object recognition on mobile devices, in: Proc. of the ACM Conference on Embedded Networked Sensor Systems (SenSys'15), Seoul, South Korea, 2015.
[21] T. Zhang, A. Chowdhery, P. Bahl, K. Jamieson, S. Banerjee, The design and implementation of a wireless video surveillance system, in: Proc. of ACM Annual International Conference on Mobile Computing and Networking (MobiCom'15), Paris, France, 2015, pp. 426–238.
[22] X. Li, M. Salehi, M. Bayoumi, High performance on-demand video transcoding using cloud services, in: Proc. of IEEE International Symposium on Cluster, Cloud and Grid Computing (CCGrid'16), Cartagena, Colombia, 2016, pp. 600–603.
[23] G. Gao, W. Zhang, Y. Wen, Z. Wang, W. Zhu, Y. Tan, Cost optimal video transcoding in media cloud: Insights from user viewing pattern, in: Proc. of IEEE Intl. Conf. on Multimedia and Expo (ICME'14), Chengdu, China, 2014.
[24] M. Zakerinasab, M. Wang, Dependency-aware distributed video transcoding in the cloud, in: Proc. of IEEE Conference on Local Computer Networks (LCN'15), Clearwater Beach, FL, 2015.
[25] S. Dutta, T. Taleb, P. Frangoudis, A. Ksentini, On-the-fly QoE-aware transcoding in the mobile edge, in: Proc. of IEEE Global Communications Conference (GLOBECOM'16), Washington, DC, 2016.

[26] W. Zhang, Y. Wen, H. Chen, Toward transcoding as a service: energy-efficient offloading policy for green mobile cloud, IEEE Netw. 28 (6) (2014) 67–73.
[27] J. Yoon, P. Liu, S. Banerjee, Low-cost video transcoding at the wireless edge, in: Proc. of IEEE Symposium on Edge Computing (SEC'16), Washington, DC, 2016.
[28] Y. Fu, Y. Guo, Y. Zhu, F. Liu, C. Song, Z.-H. Zhou, Multi-view video summarization, IEEE Trans. Multimed. 12 (7) (2010) 717–729.
[29] K. Muhammad, T. Hussain, M. Tanveer, G. Sannino, V. Albuquerque, Cost-effective video summarization using deep CNN with hierarchical weighted fusion for IoT surveillance networks, IEEE Internet Things J. 7 (5) (2019) 4455–4463.
[30] R. Panda, A. Roy-Chowdhury, Multi-view surveillance video summarization via joint embedding and sparse optimization, IEEE Trans. Multimed. 19 (9) (2017) 2010–2021.
[31] A. Murugan, K. Devi, A. Sivaranjani, P. Srinivasan, A study on various methods used for video summarization and moving object detection for video surveillance applications, Springer Multimed. Tools Appl. 77 (18) (2018) 23:273–23:290.
[32] M. Tsai, N. Venkatasubramanian, C. Hsu, Analytics-aware storage of surveillance videos: Implementation and optimization, in: Proc. of IEEE International Conference on Smart Computing (SMARTCOMP'20), Bologna, Italy, 2020, pp. 25–32.
[33] A. Godbehere, A. Matsukawa, K. Goldberg, Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation, in: Proc. of IEEE American Control Conference (ACC'12), IEEE, Montreal, Canada, 2012, pp. 4305–4312.
[34] FFmpeg Developers, FFmpeg documentation blackframe, 2020, URL: https://ffmpeg.org/ffmpeg-filters.html#blackframe.
[35] S. Wold, K. Esbensen, P. Geladi, Principal component analysis, Elsevier Chemom. Intell. Lab. Syst. 2 (1–3) (1987) 37–52.
[36] J. Tenenbaum, V. De, C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) (2000) 2319–2323.
[37] L. Van Maaten, G. Hinton, Visualizing data using t-SNE, JMLR J. Mach. Learn. Res. 9 (11) (2008) 2579–2605.
[38] M.A. Cox, T.F. Cox, Multidimensional scaling, in: Handbook of Data Visualization, Springer, 2008, pp. 315–347.
[39] E. Bingham, H. Mannila, Random projection in dimensionality reduction: applications to image and text data, in: Proc. of ACM International Conference on Knowledge Discovery and Data Mining (KDD'01), San Francisco, California, 2001, pp. 245–250.
[40] Y. Zhang, Q. Shang, G. Zhang, PyDRMetrics-A Python toolkit for dimensionality reduction quality assessment, Elsevier Heliyon 7 (2) (2021) e06199.
[41] H. Kellerer, U. Pferschy, D. Pisinger, Multidimensional knapsack problems, in: Knapsack Problems, Springer, 2004, pp. 235–283.
[42] G. Gens, E. Levner, An approximate binary search algorithm for the multiple-choice knapsack problem, Elsevier Inf. Process. Lett. 67 (5) (1998) 261–265.
[43] E. Lawler, Fast approximation algorithms for knapsack problems, INFORMS Math. Oper. Res. 4 (4) (1979) 339–356.
[44] C. He, J. Leung, K. Lee, M. Pinedo, An improved binary search algorithm for the multiple-choice knapsack problem, EDP Sci. RAIRO-Oper. Res. 50 (4–5) (2016) 995–1001.
[45] B. Patt-Shamir, D. Rawitz, Vector bin packing with multiple-choice, Elsevier Discrete Appl. Math. 160 (10–11) (2012) 1591–1600.
[46] InfluxDB, InfluxDB official website, 2019, URL: https://www.influxdata.com/.
[47] Alexey, YOLO-v3 and YOLO-v2 for Windows and Linux, 2019, URL: https://github.com/AlexeyAB/darknet.
[48] FFmpeg Developers, FFmpeg documentation, 2019, URL: http://ffmpeg.org/.
[49] G. Sullivan, J. Ohm, W. Han, T. Wiegand, Overview of the high efficiency video coding (HEVC) standard, IEEE Trans. Circuits Syst. Video Technol. 22 (12) (2012) 1649–1668.
[50] S. Möller, A. Raake, Quality of Experience: Advanced Concepts, Applications and Methods, Springer, 2014.
[51] R. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT press, 2018.