applied sciences

MDPI

Article

Static Video Summarization Using Video Coding Features with Frame-level Temporal Sub-Sampling and Deep Learning

Obada Issa[1,*] and Tamer Shanableh[2]

[1] Department of Computer Science and Engineering, American University of Sharjah, UAE, b00071518@aus.edu
[2] Department of Computer Science and Engineering, American University of Sharjah, UAE, tshanableh@aus.edu
* Correspondence: b00071518@aus.edu

Abstract: There is an abundance of digital video content due to the cloud's phenomenal growth and security footage, it is therefore essential to summarize these videos in data centers. This paper offers innovative approaches to the problem of key-frame extraction for the purpose of video summarization. Our approach includes feature variables extracted from the bit streams of coded videos, followed by optional stepwise regression for dimensionality reduction. Once the features are extracted and reduced in dimensionality, we apply innovate frame-level temporal sub-sampling techniques followed by training and testing using deep learning architectures. The frame-level temporal sub-sampling techniques are based on cosine similarity and PCA projections of feature vectors. We create three different learning architectures by utilizing LSTM networks, 1D-CNN networks, and Random Forests. The four most popular video summarization datasets, namely, TVSum, SumMe, OVP, and VSUMM are used to evaluate the accuracy of the proposed solutions. This includes the Precision, Recall, F-score measures, and computational time. It is shown that the proposed solutions when trained and tested on all subjective user summaries, achieved F-scores of 0.79, 0.74, 0.88, and 0.81, respectively, for the aforementioned datasets, showing clear improvements over prior studies.

Keywords: Video Summarization, Video Coding, Temporal Subsampling, Convolution Neural Networks, Long-Short Term Memory

1. Introduction

There is a surge in the amount of digital videos around the world due to the growth of the Internet and surveillance footage. Databases must be used to summarize these videos, which is where video summarization comes in handy. Video summarization is the process of creating a meaningful summary of the original video to make it easier to retrieve videos, identify anomalies, and facilitates activity tracking [1]. Video summarization is also important for several reasons, such as allowing users to quickly navigate through large amounts of video content, reducing storage space in archives, and has many practical applications in a variety of fields. Video summarization techniques can be categorized into two groups [2]. The first involves choosing sections from the original video, while the second, which is the most popular, involves choosing key frames from the original video. Therefore, this work focuses on video summarization by automatically selecting key-frames from a video.

Video summarization takes a lot of computational power, thus more effective methods are always encouraged. The summarizing process can be lengthy, and computing resources are wasted on redundant or similar frames if every frame in a video is reviewed for selection. Space reduction should also be utilized for any group of features to speed up the process and guarantee that only important features are considered [3]. This work aims to address these two issues.

Citation: To be added by editorial staff during production.

Academic Editor: Firstname Lastname

Received: date
Revised: date
Accepted: date
Published: date

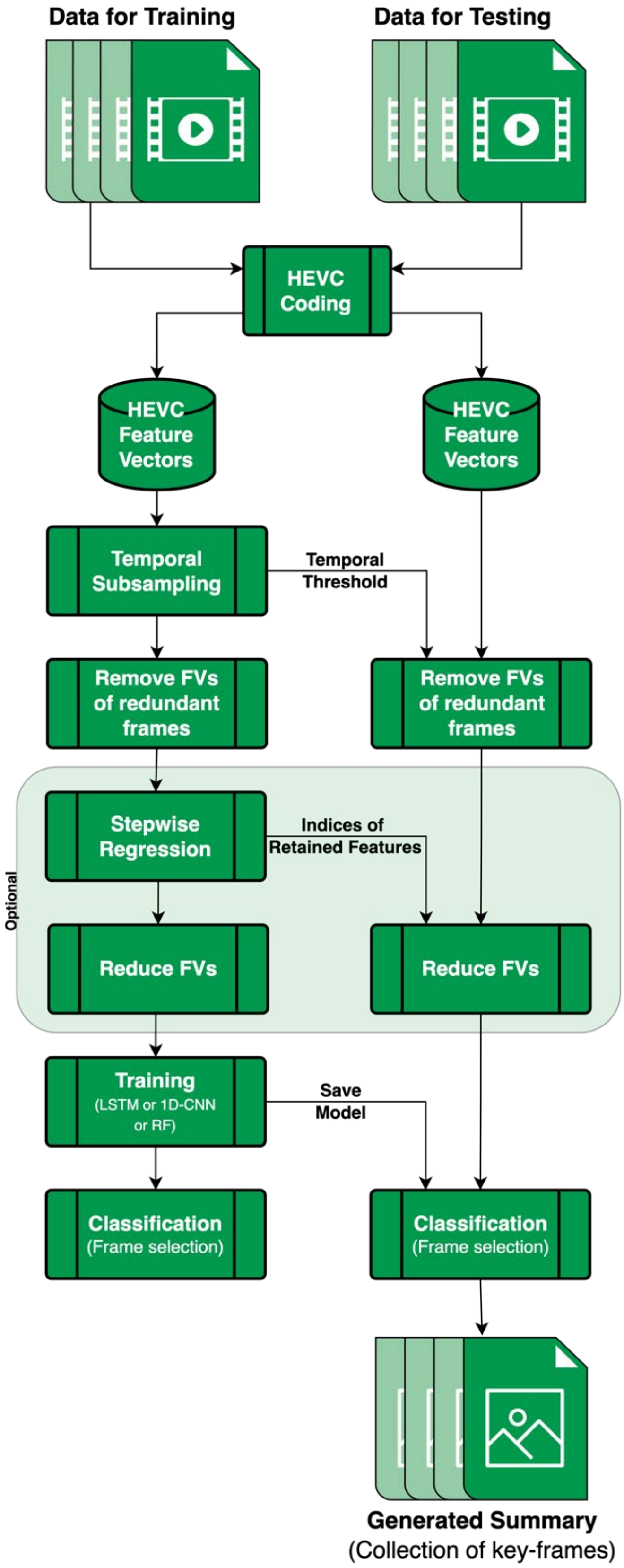

Figure 1. General overview of the system architecture. Feature extraction is done through HEVC coding. Temporal subsampling using HEVC features, PCA, or Cosine similarity. Reduction of the feature space is optional with Stepwise regression. Training is done with LSTM networks, 1D-CNNs or Random Forests.

In recent years, deep learning has become more common for generation tasks in image and video processing. To reach the desired results, a variety of tools and techniques can be employed alone or together. Most notably, Random Forests (RF) [4], Convolution Neural Networks (CNN) [5], and Long Short-Term Memory (LSTM) [6].

While video compression community belongs to the electrical engineering discipline, the deep learning community belongs to the computer and data science disciplines. The deep learning community frequently struggles with inadequate understanding of video compression due to the division in these research fields. With the growth of the High Efficiency Video Codec (HEVC) video standard [7], HEVC information in the video bitstream is often ignored and underutilized in the deep learning field. This work aims at leveraging the useful information encapsulated by HEVC coding in the video bitstream. HEVC bitstream information in the form of features was proven useful in several applications such as static video summarization [8], encoding speedup and video transcoding [9], data embedding [10], detection of double and triple compression [11], and saliency detection [12].

This work also presents novel methods for temporal subsampling of frames based on HEVC features, Principle Component Analysis (PCA), and Cosine similarity. In addition, this paper presents the use of stepwise regression (SW) for reducing the dimensionality of the feature space. A general overview of the system architecture is shown in Figure 1. The main contributions can be summarized as follows:

- The introduction of two new architectures for video summarization based on HEVC features using LSTM networks and 1D-CNNs
- The introduction of two new subsampling methods based on cosine similarity and projections of HEVC feature vectors.
- Complete experimental results with the four most commonly used datasets in video summarization, namely, TVSum, SumMe, OVP, and VSUMM. The use of all fours datasets in one research paper is rarely used in the literature, if any. From our observations and experimental results, it is rarely the case that a reported video summarization solution works well on all four datasets. Therefore, most paper opt to use a subset of these four datasets.
- Detailed discussion about the suitability of different methodologies used in digital video summarization including accuracy and computational time.

Video summarization has been the subject of substantial research over the past two decades. The efforts made to handle the challenge of video summarization are outlined in this section, with a focus on deep learning-based approaches.

Researchers in [13] take advantage of spatio-temporal learning with 3D-CNNs, LSTMs, and Recurrent Neural Networks to detect soccer video highlights. A GAN-based framework was presented by [14] with an attention-aware Ptr-Net generator and a 3D-CNN discriminator. HEVC intra-frame coding was leveraged by [15] through merging weighted luminance and chrominance values with texture-based feature against a threshold to group frames into a video summary. A stacked memory network (SMN) with LSTM layers was presented in [16] that models long dependencies among frames to lower redundancy in the final summaries. A framework presented in [17] focuses on cost-sensitive learning by having a spatial stream that represents the appearance of frames, and a temporal stream that uses motion vectors to represent the temporal information of a video.

Researchers in [18] built an unsupervised GAN with an attention mechanism to detect meaningful parts of a video. In [19], motion information between frames is leveraged, where spatio-temporal information is extracted and inter-frame motion is generated from it, and a self-attention model selects key-frames for the summary. Multi-video summarization was explored by [20] by applying target-appearance-based shot segmentation along with feature extraction from frames, these features are passed to a bidirectional LSTM to generate probabilities to form a summary. An attentive encoder–decoder network was presented by the authors in [21], where they have a bidirectional LSTM as the encoder to extract contextual information between frames, and then two attention-based LSTM networks as the decoder, which uses additive and multiplicative objective functions. An encoder-decoder CNN structure was developed by [22] by having a diagnostic view plane detection network as the encoder, followed by a decoder that feeds feature into a bidirectional LSTM to analyze features of preceding and future frames. The final reinforcement learning network selects key-frames for the summary. Video summarization was achieved by [23] on the Internet of Things (IoT) domain by developing a CNN for shot segmentation and image memorability, with using aesthetic- and entropy-based features to ensure summary variation. The work by [24] uses motion information and clustering validity index to segment shots and select key-frames by estimating their forward and backward motion.

A self-attention binary neural tree (SABT-Net) model is presented in [25], where they use GoogleNet for feature extraction along with shot encoding, branch routing, self-attention, and score prediction modules to achieve video summarization. Authors in [26] used a sparse autoencoder to combine feature vectors derived from multiple pre-trained CNNs into a reduced space with a Random Forests classifier to form video summaries. A TTH-RNN was presented in [27] and comprises a tensor-train embedding layer with a hierarchical LSTM to capture forward and backward temporal intra-shot dependencies and encodes inter-shot dependencies to establish the importance of each frame and form the final summary. The researchers in [28] offers CLIP-It, a framework for dealing with query-focused video summarization by having a multimodal transformer that correlates frames with user-written queries.

The research by [29] proposes a deep hierarchical LSTM with attention for Video summarization (DHAVS) in response to the LSTMs' inability to handle longer video sequences. They use a 3D-CNN to extract spatio-temporal features and an attention-based hierarchical LSTM module to capture the temporal correlations between video frames. Since most summarizing techniques analyze the visual components of the video and ignore audio elements, [30] provide a method that uses both the visual and audio information. Structural similarity index is used to determine similarity among frames and Mel-frequency cepstral coefficient for feature extraction from audio signals.

The work by [31] uses GANs to extract representative parts of the videos as features through reconstruction loss followed by knowledge distillation using a basic network for key-frame selection. The authors in [32] use a bidirectional LSTM that takes advantage of the underlying hierarchical structure of video sequences and learns temporal representations via intra-block and inter-block attention. They then partition shots and calculate shot-level importance scores to rank the frames that go into the final video summary.

2. Methodology

2.1. Data preprocessing

The original videos were converted to YUV frames before encoding them using a HEVC/H.265 video coder. We modified the coder to produce low-level features which are discussed in this section. The HEVC codec is used to compress the videos, hence, rich feature sets can be extracted from based on the quadratic recursive splitting of the coding units (CUs) in HEVC. An overview of the process of acquiring the HEVC feature set is shown in Figure 2.

CUs in HEVC can vary in depth from 0, which is typically equivalent to a maximum block size of 64x64 pixels, to 3, which is equivalent to a block of 16x16 pixels. CUs are then split to prediction units (PUs) of size 4x4 to 32x32, which are then further split into transform units (TUs) of size 4x4 to 32x32. Figure 3 illustrates the partitioning scheme followed in HEVC coding. We base our feature vectors on the partitioning and prediction information found in the output bit streams.

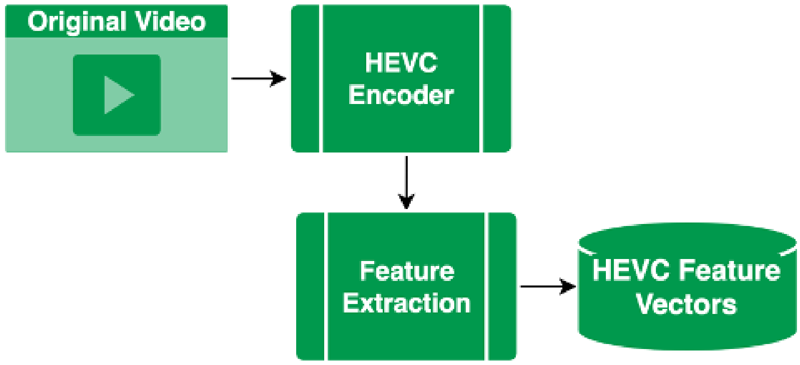

Figure 2: MPEG to HEVC video conversion process to extract HEVC features.

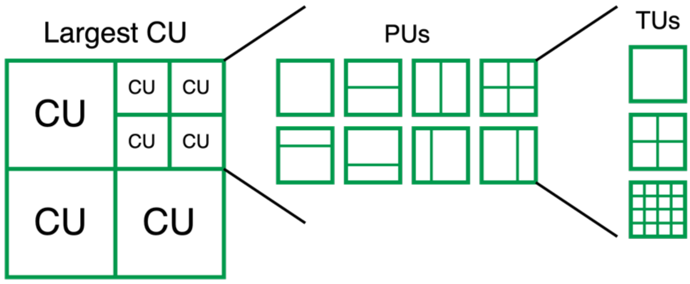

Figure 3: Coding Unites partitioning in HEVC coding.

For video summarization, we presented a set of 64 feature variables. The variables are chosen to quantify the spatiotemporal activity of the video frames. Table 1 has a list of the variables. Feature variables in Table 1-A are averaged per frame and the rest in Table 1-B are not. The tables use the abbreviations MVD for motion vector difference, SAD for sum of absolute differences, and CU for coding unit.

Table 1: HEVC features extracted per frame from a custom HEVC decoder [8]. (A) Feature variables that are averaged per frame. (B) Feature variables that are not averaged per frame.

Feature number	Feature description
1	Number of CU parts
2	MVD bits per CU
3	CU bits excluding MVD bits
4	Percentage of intra CU parts
5	Percentage of skipped CU parts
6	Number of CUs with depth 0 (i.e., 64x64)
7	Number of parts with depth 1 (i.e., 32x32)
8	Number of CUs with depth 2 (i.e., 16x16)
9	Number of parts with depth 3 (i.e., 8x8)
10	Row-wise SAD of the CU prediction error
11	Column-wise SAD of the CU prediction error
12	Ratio of gradients (i.e., feature 10 divided by feature 11) per CU
13	Total distortion per CU as computed by the HEVC encoder

(A)

Feature number	Feature description
14 to 22	Standard deviation of feature IDs 1-9 per frame
23	Max CU depth per frame

24	For CUs with depth > 0, $\log_2(\lvert sum\ of\ MVD \rvert)$
25	For CUs with depth = 0, $\log_2(\lvert sum\ of\ MVD \rvert)$
26 to 29	Standard deviation of feature IDs 23-25 per frame
30	Per frame: Summation of variance of the x and y components of all MVs
31 to 47	Histogram of x-component of all MVs per frame (using 16 pins)
48 to 64	Histogram of y-component of all MVs per frame (using 16 pins)

(B)

These feature are chosen as they capture the spatio-temporal activities of the video frames, they also rely on motion estimation and compensation with previous video frames hence preserving the temporal dependencies.

2.2. Temporal Subsampling

Temporal subsampling of frames is necessary to reduce the amount of video data that needs to be fed into our proposed models. This is commonly practiced in video summarization as many frames contain redundant content in the temporal sense. In this work, temporal subsampling is done through one of the following proposed methods:

2.2.1. HEVC-based Temporal Subsampling

We use the sum of the HEVC features as an indication of the temporal activity of individual video frames. This can be achieved by summing up all of the HEVC feature values to create a temporal activity index. The lower the index, the lower the temporal activity, which indicates that the underlying frame is potentially redundant and can be safely deleted. We carried out comprehensive experiments and we found that the summation of HEVC feature variables are lower for redundant frames. Conceptually this is a valid conclusion as the HEVC feature variables mainly reply on motion estimation and compensation, thus capture the temporal activity of the video frames. Lower summations pertain to redundant frames and vice versa.

In general, the temporal activity index of each frame is compared with a threshold to determine whether or not it will be deleted.

The calculation of the threshold is based on the train dataset in each of the 5 splits in each run. The average values of each and every feature listed in Table 1 are calculated per train split using video frames with ground truth zero (i.e., video frames that are not included in the video summary). This results in 64 average values that are summed to generate “sum of averages”. Likewise, the standard deviation of each and every feature listed in Table 1 are calculated per train split using video frames with ground truth zero. This results in 64 standard deviation values that are summed to generate “sum of standard deviations”. Lastly, the threshold is computed as: “sum of averages” + “sum of standard deviations”. This process is illustrated in Figure 4. To vary the percentage of deleted frames, we add a multiplier to the calculated threshold which has a range of 0 to 1. In this work, using empirical testing, we set the multiplier to 0.3. Consequently, a video frame is retained if its sum of features is greater than the calculated threshold and vice versa.

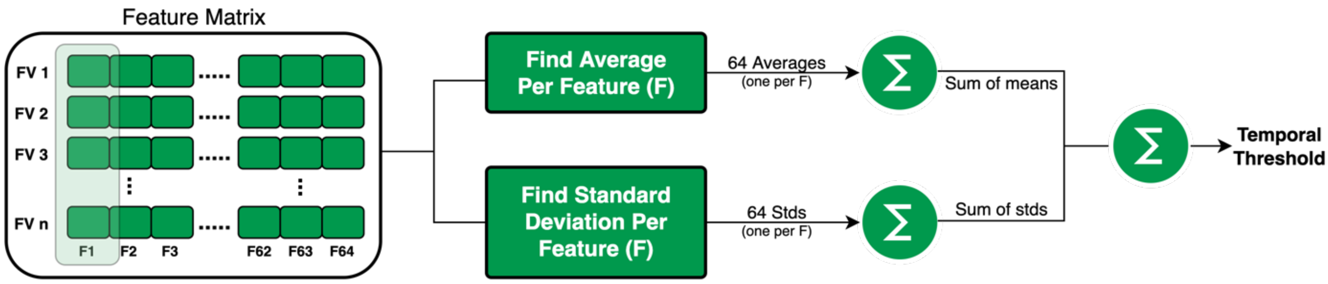

Figure 4: Calculation of temporal activity threshold for temporal sub-sampling with HEVC features.

2.2.2. PCA-based Temporal Subsampling

Principal Component Analysis (PCA) is a well-known dimensionality reduction method [33]. In our proposed setup, we use PCA to project each of the feature vectors into a scalar value. Consequently, the consecutive differences of projected values are computed and stored in list *D*. After that, for each difference element d in *D*, we check it against a threshold and decide whether or not to retain the underlying video frame. The threshold is based on statistics gathered from the projected feature vector values as detailed in Algorithm 1.

The theory here is that lower differences between principle components belonging to feature vectors of frames mean higher similarity between them, which indicates redundancy and allows us to remove one of the frames. For example, for the following frames: $[fr_1, fr_2, fr_3, fr_4, \ldots, fr_n]$ and their feature vectors: $[v_1, v_2, v_3, v_4, \ldots, v_n]$.The first principle component would look like: $[p_1, p_2, p_3, p_4, \ldots, p_n]$ and the consecutive differences would be: $[d_1, d_2, d_3, d_4, \ldots, d_{n-1}]$ with d_1 and d_2 being the difference between $p_1 - p_2$ and $p_2 - p_3$, respectively. In Algorithm 1, with the calculation of the TH, mean and std are the mean and standard deviation of all values in *D*, respectively. If d_1 is less than the composite thresholding value, then fr_1 gets marked for elimination. This continues until all feature vectors are covered.

This proposed algorithm relies on projecting feature vectors into scalars. The temporal activity threshold is computed based on the means and standard deviations of the differences of these scalar values pertaining to consecutive feature vectors of a video sequence, hence the use of the first PCA component only.

Algorithm 1: PCA-based temporal subsampling of frames.

```
Input:
FVs_train[]: Feature matrix of train data set
FVs_test[]: Feature matrix of test data set
k: Predetermined multiplier
Output:
IDX_DEL[]: Frame indices to delete

// Calculate temporal TH
[Projected_FVs, first_PC] = Project FVs_train using PCA into scalar values
D = Consecutive differences of Projected_FVs
mean = Mean of all values in D
std = Standard deviation of all values in D
TH = mean + (k × std)

// Perform temporal sub-sampling
for each FV in FVs_test do
    p = Project FV using first_PC
    if p ≤ TH
        Append index of FV to IDX-DEL[]
    end
end
```

2.2.3. Cosine-based Temporal Subsampling

Cosine similarity [34] is a metric that assesses how similar two vectors are to one another. It represents the cosine of the angle formed by two vectors. Cosine similarity is formally defined as the division between the dot product of vectors and the product of the Euclidean magnitude of each vector. The range of the cosine similarity value is from 0 to 1, with 1 denoting the highest similarity and 0 denoting the lowest. The following is the equation for the cosine similarity score between two feature vectors fi and fj:

$$similarity = cos(\theta) = \frac{f_i \cdot f_j}{\parallel f_i \parallel \parallel f_j \parallel} \tag{1}$$

In our setup, we apply cosine similarity between each feature vector and its successor, and then store the similarity score and index of the first feature vector in a tuple list S. After gathering all the similarity scores, the tuple list S is sorted ascendingly. All the feature vectors denoted by scores in the upper 90% (i.e., the scores closer to 1) in the tuple list S are marked for elimination. This subsampling process is detailed in Algorithm 2.

Algorithm 2: Cosine-based temporal subsampling of frames.

```
Input:
FVs[]: Feature matrix of train and test data sets
Output:
IDX_DEL[]: Frame indices to delete

Scores{}: Empty tuple to hold cosine scores
for i = 0...count_of(FVs)-1 do
    C = Cosine score between FVs(i,:) and FVs(i+1,:)
    Append [C, i] to Scores {}
end
Sort Scores{} ascendingly (based on C values)
IDX_DEL[] = Indices (i) of upper 90th percentile in Scores{}
```

The concept here is that higher similarity scores between feature vectors implies higher similarity between them, which indicates redundancy and allows the algorithm, to eliminate one of the frames. For example, if we have the following frames: $[fr_1, fr_2, fr_3, fr_4, \dots, fr_n]$ and their feature vectors: $[v_1, v_2, v_3, v_4, \dots, v_n]$. The cosine similarity scores between them are: $[c_1, c_2, c_3, c_4, \dots, c_{n-1}]$ with c_1 and c_2 being the score between $v_1 - v_2$ and $v_2 - v_3$, respectively. If c_1 is in the upper 90% of the similarity indices then fr_1, which is represented by v_1, is marked for elimination, and this continues until all feature vectors are covered.

2.3 Reducing the Feature Space

A supervised feature-selection approach known as stepwise regression is used to automatically select the most relevant predictor variables used to predict response variables [35]. The authors in [36] first suggested using stepwise regression in video-based intelligent systems. Since then, it has been effectively employed with many vision-based applications, as documented in several works, including [12], [37], and [38].

In this study, we use stepwise regression to reduce the dimensionality of our feature vectors, where features are treated as predictors and the class labels are treated as response variables. This is to assess the suitability of the selected features and consequently reducing dimensionality of the feature vectors if needed. Stepwise regression is only used with training data because it is a supervised approach. Later, the test data's dimensionality is reduced by using the indices of the retained feature variables of the training set, as illustrated in Figure 5.

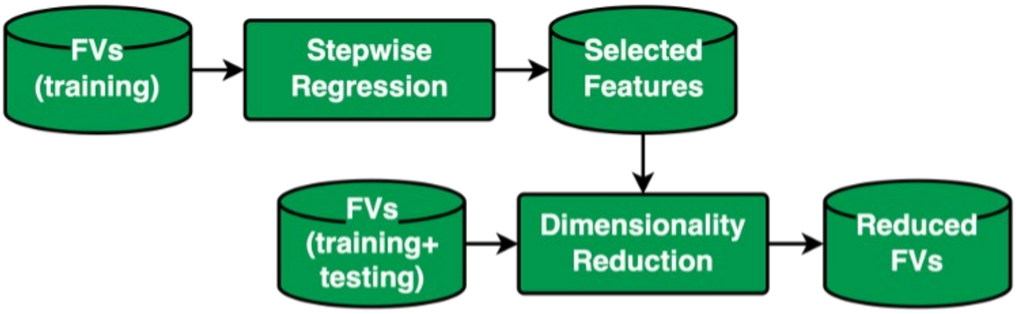

Figure 5: General overview of feature space reduction with stepwise regression.

For completeness, a summary of the stepwise regression algorithm is as follows; for a feature set of $x_1, x_2, \ldots, x_k$, F_{in} is the F-random feature for the feature to be added to the reduced feature space and F_{out} is the feature to be dropped from the reduced feature space. The following are the steps for stepwise regression:

1. Create single-set models from all features:

$$h(x) = \theta_0 + \theta_1 x_1 \tag{2}$$

where $h(x)$ is the hypothesis that the added features are important for classification. x_1 was one of the features that yielded the highest F-score. f_1 is the statistic of x_1 and is given by the following formula:

$$f_1 = \frac{SS_R(\theta_2|\theta_1\theta_0)}{MS_E(x_2, x_1)} \tag{3}$$

where MS_E is the mean square error and SS_R is the regression sum square error.

2. Repeat step 1 for all feature variables. For every new $h(x)$ produced, it is examined in combination with the existing $h(x)$ if they produce a higher hypothesis than the older $h(x)$ alone. We add x_2 if its f_2 is greater than F_{in} and obtain the following:

$$f_2 = \frac{SS_R(\theta_1|\theta_2\theta_0)}{MS_E(x_1, x_2)} \tag{4}$$

$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \tag{5}$$

After adding x_2, x_1 is checked for removal by comparing f_1 to the new F_{out}. If f_1 is lesser, then x_1 is dropped.

3. The algorithm continues until there are no features to add or drop, with the final hypothesis looking similar to the following:

$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots \tag{6}$$

2.4 Video Summarization Architectures

2.4.1. LSTM-based Architecture

Recurrent neural networks (RNN) [39] are a type of neural network used with sequential or time series data. They differ from standard neural networks, which assume that inputs and outputs are independent, in that they remember information from earlier inputs and use it to impact the current input and output. A major drawback of RNN networks is that they are susceptible to the vanishing gradient problem [40]. The gradient of the loss function approaches zero as the network's number of layers with activation functions increases, making the network more challenging to train. Due to the vanishing gradient problem RNNs are not able to remember long-term dependencies. Long Short-Term Memory Network (LSTM) is an advanced RNN network that allows information to persist

[6]. It is capable of handling the vanishing gradient problem with a chain structure that contains memory blocks called cells.

These cells can forget information that is no longer useful before passing it to the next cell. The output of the cell is taken as input to the other. This chain structure is what allows the LSTM to only retain the useful information without suffering from the vanishing gradient problem. The LSTM network can remember the information between different frames of the video while only retaining the important information.

The LSTM architecture used in this work is a 4-layer LSTM network with 50 nodes in each layer. The proposed LSTM architecture is shown in Figure 6 (left).

2.4.2. 1D-CNN-based Architecture

Convolutional neural networks (CNN), as opposed to conventional artificial neural networks, can combine feature extraction and classification into a single learning body, averting the need for fixed and manually constructed features. In a typical 2-dimensional CNN, the kernel can slide along two dimensions of the data [5]. A kernel is a matrix of weights that extracts key information by multiplying them by the input. Contrary, in 1-dimensional CNN (1D-CNN), the kernel slides along one dimension of the data where the convolution operation is applied, significantly reducing the computational complexity.

1D-CNNs are usually used with sequential data due to their simplicity and effectiveness, which is why the architecture used in this work is a single-layer 1D-CNN with 256 filters of size 5. Figure 6 (right) shows the proposed 1D-CNN architecture.

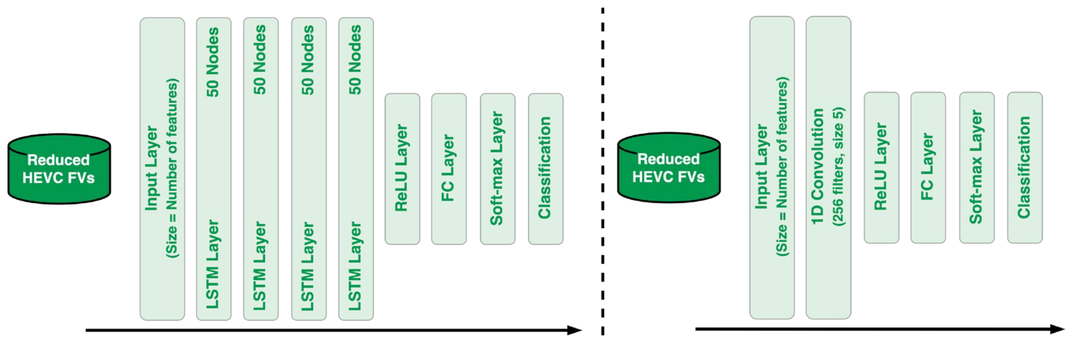

Figure 6: Proposed LSTM network architecture (Left), and 1D-CNN network architecture (Right).

2.4.3. Random Forests-based Architecture

Random Forests is a supervised learning approach. An ensemble of decision trees or a “forest” are usually trained using the "bagging" method. The fundamental concept of the bagging method is that the final output is improved by combining several learning models [4]. Random Forests increases the model's randomness while creating the decision trees. When splitting a node, it looks for the strongest feature among a random group of features rather than the best feature from the entire set. There is significant variety as a result, which usually results in a better overall model. By using random thresholds for each feature, Random Forests make trees even more random, as opposed to searching for the best thresholds (like in conventional decision trees). We employ a threshold of 0.9 in our implementation in order to keep features with values over the threshold. When none of the features are higher than the threshold, then all of them are used. For training across the chosen features, we specify that the forest produces 128 trees. Then, in order to quantify the findings, we compute a few performance measures using the predicted labels that we had previously saved.

3. Experimental Results

This section may be divided by subheadings. It should provide a concise and precise description of the experimental results, their interpretation, as well as the experimental conclusions that can be drawn.

3.1 Datasets

In this work, the proposed solutions are evaluated using four popular datasets for video summarization; namely, TVSum [41], SumMe [42], OVP [43], and VSUMM [43]. The TVSum dataset contains 50 videos of various genres such as news, documentaries, and vlogs at 30 fps. The SumMe dataset contains 25 videos at 30 fps. The OVP (Open Video Project) dataset has 50 videos from Open Video Project at 30 fps among several genres and have a duration of 1-4 minutes. The VSUMM dataset contains 50 videos from YouTube at 30 fps, across several genres as well and have a duration of 1-10 minutes.

In these datasets, a video summary is generated manually by a number of users and stored in a matrix referred to as "user summaries" which is used as the ground truth. Some existing research papers clearly state that they compare their automatically generated summaries against each of the user summaries and report the average F-score. While other research papers loosely mention that their automatically generated summaries are compared against the ground truth without further details.

Since this work is concerned with static video summarization or key-frame extraction, we train and test the datasets on the disjunction (inclusive OR) of all user summaries. In our published datasets we refer to these vectors as "user_summary_inclusive_OR" which we added to the files of the datasets and made publically available.

The use of all four datasets in one research paper is, to the best to our knowledge, rarely done in the reporting of experimental results in the literature. From our observation and experimental results, it is rarely the case that a reported video summarization solution works well on all four datasets. Therefore, most papers opt to use a subset of these four datasets.

3.2 Evaluation criteria

We use quantitative metrics similar to the criteria used in other works for fair comparison. We define the following metrics using the temporal overlap between the predicted summary A and the ground truth summary B:

$$Precision\ (P)\ = \frac{overlap(\boldsymbol{A},\boldsymbol{B})}{length(\boldsymbol{A})} \tag{7}$$

$$Recall\ (R) = \frac{overlap(\boldsymbol{A},\boldsymbol{B})}{length(\boldsymbol{B})} \tag{8}$$

$$F - measure\ (F) = \frac{2P \times R}{P + R} \times 100 \tag{9}$$

To put these metrics into words: Precision (P) is the percentage of true positive predictions over all positive predictions, Recall (R) is the percentage of true positive predictions over the ground truth, and the F-score (F) is the harmonic mean between them.

3.3 Experimental Setup

Before presenting the results, we describe the general setup that is common to all three proposed architectures. After the video coding feature vectors are generated and the temporal sub-sampling algorithm is applied, the feature vectors are split into a 20%-80% fashion for testing and training, respectively. We apply cross-validation with 5 folds (K=5), where, in every fold, the new testing set shifts by 20% and the older testing set is added back to the training set. The results are then averaged over 5 folds. The training setup is illustrated in Figure 7.

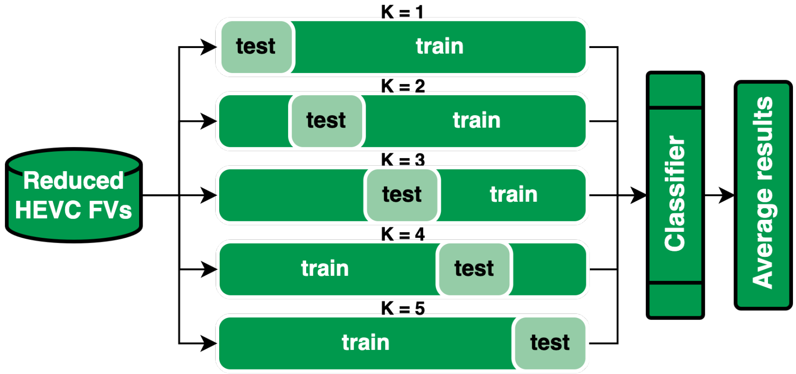

Figure 7: General overview of learning architecture with averaged results over 5 folds of cross-validation

We use HEVC features derived from the custom re-encoder mentioned in Section III. We test our setups with and without dimensionality reduction of the feature space. In addition, the proposed temporal subsampling of video frames methods using HEVC features, PCA projections, and cosine similarity are all tested with the following three learning architectures: 1D- CNNs, LSTM networks and Random Forests. For each of the 4 datasets, the top performing model from each learning architecture is shortlisted and compared against benchmark methods in the literature. First, the results are reported for every dataset, then the best models are compared with the literature, followed by a thorough discussion of the results.

The metrics used for comparison are Precision (P), Recall (R) and F-score. The experiments were conducted on a PC with a 9th gen Intel i9, 32 GB of RAM and NVIDIA RTX 2070 GPU.

3.4 Results

3.4.1. TVSum dataset

The best results across all learning architectures in Table 2-A do not use stepwise regression (denoted as SW in the tables), while the second-best results do use it for dimensionality reduction of the feature space. HEVC-based temporal subsampling achieves the highest results on the TVSum dataset, regardless of using stepwise regression or not. The highest overall scores appear with using the LSTM network.

Table 2: Proposed solutions: F-scores of the 2 best performing models using the 3 proposed learning architectures, with and without reduction of the feature space, and across the 3 proposed temporal subsampling methods on the TVSum (A), SumMe (B), OVP (C), and VSUMM (D) datasets.

Architecture	Reduction	Temporal subsampling	F-score	Time (K=5)	Time (K=1)
1D-CNN	Stepwise	HEVC-based	0.728	20.36	4.07
1D-CNN	None	HEVC-based	0.737	48.86	9.77
RF	Stepwise	HEVC-based	0.737	22.13	4.43
RF	None	HEVC-based	0.740	49.74	9.95
LSTM	Stepwise	HEVC-based	0.775	81.43	16.29
LSTM	None	HEVC-based	**0.785**	195.42	39.08

(A)

Architecture	Reduction	Temporal subsampling	F-score	Time (K=5)	Time (K=1)
1D-CNN	None	PCA-based	0.610	12.38	2.48
1D-CNN	None	HEVC-based	0.644	16.29	3.26
LSTM	None	PCA-based	0.646	49.51	9.90
LSTM	None	HEVC-based	0.676	65.14	13.03
RF	None	PCA-based	0.720	13.22	2.64
RF	None	HEVC-based	**0.737**	17.04	3.41

(B)

Architecture	Reduction	Temporal subsampling	F-score	Time (K=5)	Time (K=1)
1D-CNN	Stepwise	Cosine-based	0.827	6.17	1.23
1D-CNN	None	HEVC-based	0.840	22.94	4.59
RF	Stepwise	Cosine-based	0.852	6.86	1.37
RF	None	HEVC-based	0.864	24.70	4.94
LSTM	Stepwise	Cosine-based	0.866	25.52	5.10
LSTM	None	HEVC-based	**0.879**	91.75	18.35

(C)

Architecture	Reduction	Temporal subsampling	F-score	Time (K=5)	Time (K=1)
1D-CNN	Stepwise	HEVC-based	0.728	13.86	2.8
1D-CNN	None	HEVC-based	0.744	33.26	6.7
LSTM	Stepwise	HEVC-based	0.753	55.43	11.1
LSTM	None	HEVC-based	0.770	133.03	26.6
RF	Stepwise	HEVC-based	0.799	14.67	2.9
RF	None	HEVC-based	**0.808**	35.21	7.0

(D)

3.4.2. SumMe dataset

Regardless of the temporal subsampling method used, all results in on the SumMe dataset in Table 2-B are without stepwise regression. Across all learning architectures, the best model uses HEVC-based temporal subsampling and the second-best model uses PCA-based temporal subsampling. The highest overall scores appear with using the Random Forests architecture.

3.4.3. OVP dataset

Across all three learning architectures in Table 2-C, the best results on the OVP dataset come from using HEVC-based temporal subsampling and without applying stepwise regression. The second-best model across all learning architectures, however, uses stepwise regression for dimensionality reduction and cosine similarity for temporal subsampling. The highest overall score appears with using the LSTM network.

3.4.4. VSUMM dataset

In Table 2-D for the VSUMM dataset, all the results across all three learning architectures use HEVC-based temporal subsampling. The first across all learning architectures is without stepwise regression, while the second is with stepwise regression. The highest overall scores are with using Random Forests.

3.4.5. All datasets versus benchmarks

Again, as mentioned above, we carried out the training and testing using all user summaries combined into one label vector. In existing work, different papers use different approaches for training and testing with some of them loosely using the term ground truth without further details. Nonetheless, for completeness, in this section we provide comparisons against existing work which carries out training and testing using different approaches but with the same datasets.

Table 3: F-scores of our best performing models from the 3 proposed learning architectures against benchmark models in the literature on the TVSum (A), SumMe (B), OVP (C) and VSUMM (D) datasets. Sorted ascendingly from top to bottom.

Method	F-score
RR-STG [44]	0.637
PGL-SUM [45]	0.654
SMN [46]	0.675
Ours (1D-CNN)	0.737
Ours (RF)	0.740
Ours (LSTM)	**0.785**

(A)

Method	F-score
MC-VSA [47]	0.534
re-seq2seq [48]	0.556
MAVS [16]	0.583
Ours (1D-CNN)	0.644
Ours (LSTM)	0.676
Ours (RF)	**0.737**

(B)

Method	F-score
VRHDPS [49]	0.630
VSUMM [50]	0.680
VISCOM [51]	0.720
Ours (1D-CNN)	0.840
Ours (RF)	0.869
Ours (LSTM)	**0.879**

(C)

Method	F-score
VSUMM [50]	0.670
VISCOM [51]	0.670
VRHDPS [49]	0.680
Ours (1D-CNN)	0.744
Ours (LSTM)	0.770
Ours (RF)	**0.808**

(D)

Tables 3 (A-D) contain the F-scores of our best performing models from each learning architecture compared against state-of-the-art works in the literature on the SumMe, TVSum, OVP, VSUMM datasets. With the SumMe dataset in Table 3-A, our Random Forests model with no dimensionality reduction and with HEVC-based temporal subsampling surpasses the highest scores in the literature.

With the TVSum dataset in Table 3-B, our LSTM network model with no dimensionality reduction and with HEVC-based temporal subsampling also exceeds the highest scores in the literature. Our second and third best models with Random Forests and 1D-CNNs, without dimensionality reduction and with HEVC-based temporal subsampling of frames also exceeded benchmark scores.

With the OVP dataset in Table 3-C, our model with the LSTM network, without dimensionality reduction and with HEVC-based temporal subsampling of frames, surpasses the highest scores in the literature. Our second and third ranking models with the Random Forests and 1D-CNNs, without dimensionality reduction and with HEVC-based temporal subsampling also outperformed benchmark scores.

With the VSUMM dataset in Table 3-D, our model with Random Forests without using stepwise regression and with HEVC-based temporal subsampling tops the best scores in the literature. Our second and third best models with LSTM networks and 1D-CNNs, without stepwise regression and with HEVC-based temporal subsampling of frames also exceeded benchmark scores.

4. Discussion of Results

4.1 Reduction of feature space

One observation from the results in Tables 2 is that the highest score is constantly achieved without resorting to reducing the dimensionality of the HEVC feature set. Dimensionality reduction methods aim to retain the most representative features and discard the features that are deemed unnecessary, redundant, or non-representative of the original image information. The fact that retaining all and not some of the 64 HEVC features yields higher scores, means that all 64 HEVC features are excellent representatives, and none of them can be discarded.

This is also true for the second highest scores across all learning architectures in the SumMe dataset, but with PCA-based temporal subsampling of frames per training set. This means that for the SumMe dataset, the quality of the features used is more important or influential than the method used for temporal subsampling of frames due to the difficult nature of the videos it contains which were intended to be used with importance- and interestingness-based applications of video summarization [52]. According to the presented results, HEVC features successfully capture importance and interestingness information of video frames.

For TVSum, OVP and VSUMM, the second highest score across all three learning architectures is when HEVC features are reduced with stepwise regression, regardless of the temporal subsampling method used. The interesting finding with the TVSum and VSUMM datasets is that the F-score of the video summarization is negatively affected by less than 2% when HEVC-based temporal subsampling of frames is used, compared to the best scores. Even in the case of the second highest scores with the OVP dataset, where dimensionality reduction is applied and cosine similarity is used for temporal subsampling, the F-score decrease is less than 2% as well. This indicates that even when some of the HEVC features are removed, regardless of the method being used for temporal subsampling, the retained features are still highly representative of the frame content and contain close and comparable information compared to the full set of HEVC features.

In general, the use of Stepwise Regression did not generate the best results in any of our experiments. This can be justified by the fact that Stepwise Regression uses linear multivariate regression for variable selection. However, the problem at hand, which is mapping feature variables to key frames is clearly non-linear and hence the performance of such a variable selection approach.

The following are examples of using stepwise regression with the TVSum and SumMe datasets. For the TVsum dataset, we found that the most significant features pertain to the following IDs from Table 1: 4,5,8,9 and standard deviation of (2-4,6-8), 10-13, 23-25, 30 and 10 bins of MVx histogram and 12 bins of the MVy histogram. Whereas for the SumMe dataset we found that the most significant features pertain to the following IDs from Table 1: 3,5-7, standard deviation of 1 and 4, 23, standard deviation of 23, 30 and 7 bins of MVx histogram and 5 bins of the MVy histogram.

4.2 Learning architecture

Recall that the HEVC features are extracted from the HEVC video coding process. Such a process is based on motion estimation and compensation which is known to make use of previous video frames in the coding of the present video frame. As such, the resultant feature vector of a video frame inherently contains information from previous frames. This justifies the outstanding results obtained using the RF and 1D-CNN architectures, that unlike LSTM networks, which lack the ability to maintain information beyond the current frame.

On the other hand, in SumMe and VSUMM, RFs achieved higher scores compared to the other two learning architectures, implying less content or scene changes in the content of the videos within these datasets. When a video contains many scene changes, LSTMs excel; However, when there are not many changes, then RFs can keep up with and exceed LSTMs in terms of classification accuracy.

The datasets where LSTM networks performed better, (i.e., TVSum and OVP), indicated that the videos contained in them have more temporal variance or scene changes in their content compared to the other two datasets. This can be explained by the way LSTM networks work, where they can retain information about older frames or content through their long memory along with the recently preceding frames with the short memory.

4.3 Elapsed runtimes

LSTM networks are computationally expensive and require at least 4 times required by 1D-CNNs or Random Forests according to our experiments. When runtime is not a priority, LSTM networks are recommended. On the other hand, when runtime is a priority, Random Forests are the learning architecture of choice. 1D-CNNs still have place when runtime is of absolute significance and the accuracy of the summary is not highly prioritized or not intended to be relied on in a sensitive application. Recall that in this work we have used cross-validation with K=5 to generate the results, the results reported in the experimental are for both K=5 and K=1.

In conclusion, as the proposed feature set contains only 64 variables , the model generation and testing time is very fast in comparison to typical work where hundreds or thousands of CNN features are used.

5. Limitations and Future work

This work was designed for key-frame extraction, or static video summarization, but in the meantime, we do not know how it can be expanded or modified to work for dynamic video summarization, which is usually a computationally heavier task. For the learning architectures used, LSTM architectures can be a limiting factor due to expensive computation. That, however, can be remedied by using alternative architectures such as light-weight 1D-CNNs and Random Forests.

In HEVC-based temporal subsampling, we mentioned having a multiplier to vary the mount of deleted or eliminated frames that was arrived at through empirical testing. This multiplier can be potentially calculated dynamically or in an automated manner.

6. Conclusion

In this work we presented multiple proposals for generating summaries of the video content in the form of key-frames. The proposals are based on a precise and concise feature set generated from an HEVC video coder. We presented novel methods for temporal subsampling of frames using PCA projections and cosine similarity, along with the use of stepwise regression for the reduction of the feature space.

We also developed three learning architectures using LSTM networks, 1D-CNNs and Random Forests. The experimental results section presented extensive results using all four well-known datasets in the video summarization domain, namely, TVSum, SumMe, OVP, and VSUMM. The reported results surpass reviewed work in the literature in terms of F-score. The advantage against existing work is mainly attributed to our use of HEVC features that are based on video coding. Such coding is based on motion estimation and compensation, leading the final HEVC feature vectors to successfully capture temporal dependencies across frames. The reported results are not exclusive to high F-scores, but also reasonable runtimes. The feature vectors have a length of 64 features only, making them compact compared to traditional features from well-known pre-trained CNN networks that have lengths usually in the hundreds or thousands of features.

Author Contributions: Conceptualization, O.I. and T.S.; methodology, O.I. and T.S..; software, O.I. and T.S..; validation, O.I. and T.S.; formal analysis, O.I. and T.S.; investigation, O.I. and T.S.; resources, O.I. and T.S.; data curation, O.I. and T.S.; writing—original draft preparation, O.I. and T.S.; writing—review and editing, O.I. and T.S.; visualization, O.I. and T.S.; supervision, O.I. and T.S.; project administration, O.I. and T.S.; funding acquisition, O.I. and T.S. All authors have read and agreed to the published version of the manuscript.

Funding: The work in this paper is supported by the American University of Sharjah under research grant number FRG22-E-E44. The work in this paper was also supported, in part, by the Open Access Program from the American University of Sharjah. This paper represents the opinions of the author(s) and does not mean to represent the position or opinions of the American University of Sharjah.

Data Availability Statement: The datasets used are made publicly available via GitHub at: https://github.com/b00071518/HEVC-SVS

Acknowledgments: This paper represents the opinions of the author(s) and does not mean to represent the position or opinions of the American University of Sharjah.

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

References

1. Basavarajaiah M, Sharma P. Survey of Compressed Domain Video Summarization Techniques. *ACM Comput Surv*. 2020;52(6):1-29. doi:10.1145/3355398
2. Apostolidis E, Adamantidou E, Metsai AI, Mezaris V, Patras I. Video Summarization Using Deep Neural Networks: A Survey. *arXiv:210106072 [cs]*. Published online September 27, 2021. Accessed November 15, 2021. http://arxiv.org/abs/2101.06072
3. Van Der Maaten L, Postma E, Van den Herik J, others. Dimensionality reduction: a comparative study. *J Mach Learn Res*. 2009;10(66-71):13.
4. Breiman L. Random Forests. *Machine Learning*. 2001;45(1):5-32. doi:10.1023/A:1010933404324
5. Szegedy C, Liu W, Jia Y, et al. Going Deeper with Convolutions. *arXiv:14094842 [cs]*. Published online September 16, 2014. Accessed November 28, 2021. http://arxiv.org/abs/1409.4842
6. Hochreiter S, Schmidhuber J. Long Short-Term Memory. *Neural Computation*. 1997;9(8):1735-1780. doi:10.1162/neco.1997.9.8.1735
7. Sullivan GJ, Ohm JR, Han WJ, Wiegand T. Overview of the High Efficiency Video Coding (HEVC) Standard. *IEEE Trans Circuits Syst Video Technol*. 2012;22(12):1649-1668. doi:10.1109/TCSVT.2012.2221191
8. Issa O, Shanableh T. CNN and HEVC Video Coding Features for Static Video Summarization. *IEEE Access*. 2022;10:72080-72091. doi:10.1109/ACCESS.2022.3188638
9. Hassan M, Shanableh T. Predicting split decisions of coding units in HEVC video compression using machine learning techniques. *Multimed Tools Appl*. 2019;78(23):32735-32754. doi:10.1007/s11042-018-6882-8
10. Shanableh T. Altering split decisions of coding units for message embedding in HEVC. *Multimed Tools Appl*. 2018;77(7):8939-8953. doi:10.1007/s11042-017-4787-6
11. Youssef S, Shanableh T. Detecting Double and Triple Compression in HEVC Videos Using the Same Bit Rate. *SN COMPUT SCI*. 2021;2(5):406. doi:10.1007/s42979-021-00800-8
12. Shanableh T. Saliency detection in MPEG and HEVC video using intra-frame and inter-frame distances. *SIViP*. 2016;10(4):703-709. doi:10.1007/s11760-015-0798-9
13. Agyeman R, Muhammad R, Choi GS. Soccer Video Summarization Using Deep Learning. In: *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE; 2019:270-273. doi:10.1109/MIPR.2019.00055
14. Fu TJ, Tai SH, Chen HT. Attentive and Adversarial Learning for Video Summarization. In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE; 2019:1579-1587. doi:10.1109/WACV.2019.00173
15. Wang F, Liu F, Zhu S, Fu L, Liu Z, Wang Q. HEVC intra frame based compressed domain video summarization. In: *Proceedings of the International Conference on Artificial Intelligence, Information Processing and Cloud Computing - AIIPCC '19*. ACM Press; 2019:1-7. doi:10.1145/3371425.3371450
16. Wang J, Wang W, Wang Z, Wang L, Feng D, Tan T. Stacked Memory Network for Video Summarization. In: *Proceedings of the 27th ACM International Conference on Multimedia*. ACM; 2019:836-844. doi:10.1145/3343031.3350992
17. Zhong S hua, Wu J, Jiang J. Video summarization via spatio-temporal deep architecture. *Neurocomputing*. 2019;332:224-235. doi:10.1016/j.neucom.2018.12.040
18. Apostolidis E, Adamantidou E, Metsai AI, Mezaris V, Patras I. Unsupervised Video Summarization via Attention-Driven Adversarial Learning. In: Ro YM, Cheng WH, Kim J, et al., eds. *MultiMedia Modeling*. Vol 11961. Lecture Notes in Computer Science. Springer International Publishing; 2020:492-504. doi:10.1007/978-3-030-37731-1_40
19. Huang C, Wang H. A Novel Key-Frames Selection Framework for Comprehensive Video Summarization. *IEEE Trans Circuits Syst Video Technol*. 2020;30(2):577-589. doi:10.1109/TCSVT.2019.2890899
20. Hussain T, Muhammad K, Ullah A, Cao Z, Baik SW, de Albuquerque VHC. Cloud-Assisted Multiview Video Summarization Using CNN and Bidirectional LSTM. *IEEE Trans Ind Inf*. 2020;16(1):77-86. doi:10.1109/TII.2019.2929228
21. Ji Z, Xiong K, Pang Y, Li X. Video Summarization With Attention-Based Encoder–Decoder Networks. *IEEE Trans Circuits Syst Video Technol*. 2020;30(6):1709-1717. doi:10.1109/TCSVT.2019.2904996
22. Liu T, Meng Q, Vlontzos A, Tan J, Rueckert D, Kainz B. Ultrasound Video Summarization Using Deep Reinforcement Learning. In: Martel AL, Abolmaesumi P, Stoyanov D, et al., eds. *Medical Image Computing and Computer Assisted Intervention – MICCAI 2020*. Vol 12263. Lecture Notes in Computer Science. Springer International Publishing; 2020:483-492. doi:10.1007/978-3-030-59716-0_46
23. Muhammad K, Hussain T, Tanveer M, Sannino G, de Albuquerque VHC. Cost-Effective Video Summarization Using Deep CNN With Hierarchical Weighted Fusion for IoT Surveillance Networks. *IEEE Internet Things J*. 2020;7(5):4455-4463. doi:10.1109/JIOT.2019.2950469
24. Zhao Y, Guo Y, Sun R, Liu Z, Guo D. Unsupervised video summarization via clustering validity index. *Multimed Tools Appl*. 2020;79(45-46):33417-33430. doi:10.1007/s11042-019-7582-8
25. Song J, He T, Gao L, Xu X, Hanjalic A, Shen HT. Unified Binary Generative Adversarial Network for Image Retrieval and Compression. *Int J Comput Vis*. 2020;128(8-9):2243-2264. doi:10.1007/s11263-020-01305-2

26. Nair MS, Mohan J. Static video summarization using multi-CNN with sparse autoencoder and random forest classifier. *SIViP*. 2021;15(4):735-742. doi:10.1007/s11760-020-01791-4
27. Zhao B, Li X, Lu X. TTH-RNN: Tensor-Train Hierarchical Recurrent Neural Network for Video Summarization. *IEEE Trans Ind Electron*. 2021;68(4):3629-3637. doi:10.1109/TIE.2020.2979573
28. Narasimhan M, Rohrbach A, Darrell T. CLIP-It! Language-Guided Video Summarization. Published online December 7, 2021. Accessed October 5, 2022. http://arxiv.org/abs/2107.00650
29. Lin J, Zhong S hua, Fares A. Deep hierarchical LSTM networks with attention for video summarization. *Computers & Electrical Engineering*. 2022;97:107618. doi:10.1016/j.compeleceng.2021.107618
30. Rhevanth M, Ahmed R, Shah V, Mohan BR. Deep Learning Framework Based on Audio–Visual Features for Video Summarization. In: Gupta D, Sambyo K, Prasad M, Agarwal S, eds. *Advanced Machine Intelligence and Signal Processing*. Vol 858. Lecture Notes in Electrical Engineering. Springer Nature Singapore; 2022:229-243. doi:10.1007/978-981-19-0840-8_17
31. Sreeja MU, Kovoor BC. A multi-stage deep adversarial network for video summarization with knowledge distillation. *J Ambient Intell Human Comput*. Published online January 24, 2022. doi:10.1007/s12652-021-03641-8
32. Zhu W, Lu J, Han Y, Zhou J. Learning multiscale hierarchical attention for video summarization. *Pattern Recognition*. 2022;122:108312. doi:10.1016/j.patcog.2021.108312
33. Jolliffe IT, Cadima J. Principal component analysis: a review and recent developments. *Phil Trans R Soc A*. 2016;374(2065):20150202. doi:10.1098/rsta.2015.0202
34. Singhal A, Google I. Modern Information Retrieval: A Brief Overview. *IEEE Data Engineering Bulletin*. 2001;24.
35. Montgomery DC, Runger GC. *Applied Statistics and Probability for Engineers*.; 2018.
36. Shanableh T, Assaleh K. Feature modeling using polynomial classifiers and stepwise regression. *Neurocomputing*. 2010;73(10-12):1752-1759. doi:10.1016/j.neucom.2009.11.045
37. Shanableh T. A regression-based framework for estimating the objective quality of HEVC coding units and video frames. *Signal Processing: Image Communication*. 2015;34:22-31. doi:10.1016/j.image.2015.02.008
38. Shanableh T. Detection of frame deletion for digital video forensics. *Digital Investigation*. 2013;10(4):350-360. doi:10.1016/j.diin.2013.10.004
39. Abiodun OI, Jantan A, Omolara AE, Dada KV, Mohamed NA, Arshad H. State-of-the-art in artificial neural network applications: A survey. *Heliyon*. 2018;4(11):e00938. doi:10.1016/j.heliyon.2018.e00938
40. Hochreiter S. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *Int J Unc Fuzz Knowl Based Syst*. 1998;06(02):107-116. doi:10.1142/S0218488598000094
41. Yale Song, Vallmitjana J, Stent A, Jaimes A. TVSum: Summarizing web videos using titles. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE; 2015:5179-5187. doi:10.1109/CVPR.2015.7299154
42. Gygli M, Grabner H, Riemenschneider H, Van Gool L. Creating Summaries from User Videos. In: *ECCV*. ; 2014.
43. de Avila SEF, Jr. A da_Luz, de A. Araújo A, Cord M. VSUMM: An Approach for Automatic Video Summarization and Quantitative Evaluation. In: *2008 XXI Brazilian Symposium on Computer Graphics and Image Processing*. IEEE; 2008:103-110. doi:10.1109/SIBGRAPI.2008.31
44. Liu YT, Li YJ, Wang YCF. Transforming Multi-Concept Attention into Video Summarization. Published online June 2, 2020. Accessed November 6, 2022. http://arxiv.org/abs/2006.01410
45. Zhang K, Grauman K, Sha F. Retrospective Encoders for Video Summarization. In: Ferrari V, Hebert M, Sminchisescu C, Weiss Y, eds. *Computer Vision – ECCV 2018*. Vol 11212. Lecture Notes in Computer Science. Springer International Publishing; 2018:391-408. doi:10.1007/978-3-030-01237-3_24
46. Feng L, Li Z, Kuang Z, Zhang W. Extractive Video Summarizer with Memory Augmented Neural Networks. In: *Proceedings of the 26th ACM International Conference on Multimedia*. ACM; 2018:976-983. doi:10.1145/3240508.3240651
47. Zhu W, Han Y, Lu J, Zhou J. Relational Reasoning Over Spatial-Temporal Graphs for Video Summarization. *IEEE Trans on Image Process*. 2022;31:3017-3031. doi:10.1109/TIP.2022.3163855
48. Apostolidis E, Balaouras G, Mezaris V, Patras I. Combining Global and Local Attention with Positional Encoding for Video Summarization. In: *2021 IEEE International Symposium on Multimedia (ISM)*. IEEE; 2021:226-234. doi:10.1109/ISM52913.2021.00045
49. Wu J, Zhong S hua, Jiang J, Yang Y. A novel clustering method for static video summarization. *Multimed Tools Appl*. 2017;76(7):9625-9641. doi:10.1007/s11042-016-3569-x
50. de Avila SEF, Lopes APB, da Luz A, de Albuquerque Araújo A. VSUMM: A mechanism designed to produce static video summaries and a novel evaluation method. *Pattern Recognition Letters*. 2011;32(1):56-68. doi:10.1016/j.patrec.2010.08.004
51. Mussel Cirne MV, Pedrini H. VISCOM: A robust video summarization approach using color co-occurrence matrices. *Multimed Tools Appl*. 2018;77(1):857-875. doi:10.1007/s11042-016-4300-7
52. Atencio P, German S, Branch JW, Delrieux C. Video summarisation by deep visual and categorical diversity. *IET Computer Vision*. 2019;13(6):569-577. doi:10.1049/iet-cvi.2018.5436