Detecting Double and Triple Compression in HEVC Videos Using the Same Bit Rate

Seba Youssef[1] and Tamer Shanableh[2*]

[1,2]Department of Computer Science and Engineering, American University of Sharjah, UAE
[1]g00079894@aus.edu, [2]tshanableh@aus.edu

Abstract

Digital video forensics refers to the process of analysing, examining, evaluating and comparing a video for use in legal matters. In digital video forensics, the main aim is to detect and identify video forgery to ensure a video's authenticity. When a video is edited, the original bitstream is first decoded, edited and then re-compressed. Therefore detecting re-compression in videos is a major step in digital video forensics. Video editing can be applied many times leading to multiple compressions. Thus, finding out the compression history of a video becomes an important mean for detecting any manipulation and thereby identifying the legitimacy of a video. In this work, we propose a machine learning approach to detecting double and triple compression in videos coded using the High Efficiency Video Coding (HEVC) format. Feature variables are extracted from Coding Units (CUs) and summarized into picture and Group of Pictures (GoP) feature vectors. Two classifiers are used for classifying videos into single, double and triple compression, namely; Random Forest (RF) and bi-directional Long Short-Term Memory (bi-LSTM). The latter classifier is important in digital video forensics as it exploits the temporal dependencies between feature vectors. In the experimental results, 127 video sequences are used for verifying the accuracy of the proposed solutions. Results are reported in terms of classification accuracy, confusion matrices, precision and recall. The experimental results revealed that both double and triple compression can be accurately detected using the proposed solutions with results superior to existing work.

Keywords: Double compression detection; Triple compression detection; Machine learning; Pattern recognition; video coding; HEVC

Vol.:(0123456789)
SN Computer Science (2021) 2:406
https://doi.org/10.1007/s42979-021-00800-8
SN Computer Science

1. Introduction

The field of digital video forensics refers to the scientific analysis, examination, evaluation and comparison of video to be used in legal matters. A video needs to be first validated before being used in legal cases to ensure its authenticity and suitability to court. Because professional knowledge is no longer required to edit or manipulate digital videos, many research work focus on digital forensics with the focus of detecting video manipulation. For instance, passive or blind forensics tries to extract video features that differentiate between forged and unforged videos and possibly identify the location of forgery.

Finding out the compression history of videos is one important way for identifying the authenticity and trustworthiness of a video. Over the past decade, several methods have been proposed by researchers in the forensics community for the detection of double compression in coded video sequences [1]. Multiple compressions occur when a video has undergone a series of compressions and decompressions. The number of compressions of a video sequence keeps on increasing every time the video undergoes any manipulation. This makes the efficient detection of double compression in videos significant in the field of digital forensics due to the underlying manipulation they may have. In addition, due to the existence of social media platforms, even the authentic videos can have traces of double compression despite having no manipulation undergone. This is because a second compression is automatically applied whenever a video is uploaded on a social media platform such as Facebook, Messenger, Twitter, Instagram or WhatsApp. As a result, this makes focusing solely on double compression detection not enough and thus there is a need for building reliable triple or even multiple compression detection techniques. Multiple studies have focused on double compression detection in videos while, to our knowledge, none of them addressed the issue of triple compression detection.

The detection of double and triple compression in coded videos can be viewed as a pre-step for forgery detection after which the existence and type of forgery can be identified. The solution aims to identify the effect of having the same recompression bitrate on the detection of recompression in HEVC videos. In double compression detection, we classify the sequences as unforged which are the original sequences that have undergone single compression (Class 1) and forged which are the sequences that have undergone two compressions (Class 2). Whereas in triple compression detection, three classes are used for classification with Class 1 representing the unforged samples, Class 2 representing the forged samples with two compressions and Class 3 representing the forged samples with 3 compressions. Machine and deep learning techniques are then used to report the classification accuracy, true positive and false negative rates of each of the conducted experiments.

Therefore, the aim of this paper is to address the issue of both double and triple compression detection in HEVC videos as well as to propose a new set of features that can be used to improve the existing accuracies for double compression detection. We focus on the detection of recompression in HEVC coded videos with the same re-compression bitrate. To the extent of our knowledge, only systems related to double compression detection of HEVC videos are reported in the literature. A number of parameters have been commonly used for double compression detection in HEVC videos such as the PU (Prediction Unit) type, the DCT coefficient and the TU (Transform Unit) type [2-5]. Multiple experiments were also conducted for double compression detection in shifted GoP structures where I-pictures have been relocated after recompression. For such problem, the average prediction residual sequence was commonly used as the main feature for detection [6-8].

To tackle the problem of double compression detection with the same recompression bitrate, Jiang et al. [9] proposed an efficient solution that uses the same re-compression bitrate and makes use of the GOP-based PU type statistics extracted from each picture. The solution relies on the temporal variation patterns of the PU type across GOPs in single and double compression instead of using separate pictures or the full video sequence as common with most other experiments in previous works. Three types of PUs are first extracted from the video sequences (Intra, Skipped, Predicted) and then the ratio of the Intra and Skipped PU types is calculated so that for each GOP unit, a PU sequence is generated. Then, for each GOP or PU sequence, the mean and standard deviation of these ratios are calculated to obtain the final PU-type statistic used for the re-compression detection. When the same re-compression bitrate is used, the proposed method achieves accuracies ranging from 93% to 96% with 93% for the lowest bitrate of 800kbps. Likewise, the authors in [10] reported accuracies up to 93% when using the same re-compression bitrates. It was proposed to detect double compression in HEVC videos using the Sequence of Number of Prediction Units of its Prediction Mode (SN-PUPM). An SVM classifier is used and the period analysis method is implemented on each video sequence for the detection of double compression. The same GOP size has been used for single and double compression. In an experiment that uses the same re-compression bitrate for the detection of double compression, Liang et. al. [11] achieved accuracies ranging up to 87% for the same re-compression bitrates where they extracted a 25-D feature using the histogram of the partition modes of PUs named as HPP features. The first PU in each GOP was used to extract PU information from which the 25-D HPP features are extracted. The HPP features of all GOPs of a video sequence are then averaged to obtain the final detection feature. Finally, an SVM classifier with a polynomial kernel was used with the 25-D HPP features used as an input to obtain the final detection accuracy.

More recently, in [12] a one-class classification solution in proposed for detecting double compression of MPEG-4 videos. This is needed when single compressed videos are not available for building a supervised learning system. In [13] an HEVC picture-wise double compression detection solution is proposed using a hybrid neural network. Feature variables are based on sizes of coding units and prediction modes. Using 32 YUV video sequences for training and 20 sequences for testing, the highest classification accuracy reported was 96.7%. In [14], a novel method based on the HEVC intra prediction mode is proposed for detecting double compression. The classification system is based on SVM and uses 136 YUV video clips for training and testing. Lastly, in [15], a novel double compression detection method for H.264 videos with fixed and adaptive GOP structure is proposed. The byte count of abnormal frames in video sequences and the last adaptive I-frames are combined to generate the features used for detecting double compression.

Similar to [5], [9], [13] and [14], our work uses the HEVC video coding standard but extracts features at both picture and Group-of-Pictures levels. We propose classification systems using machine leaning and deep learning with a dataset consisting of 77 YUV video sequences for training and 50 sequences for testing. More importantly, our system addresses the problem of detecting double and triple compression.

2. The HEVC Compression Standard

HEVC was developed with the main aim of addressing the current need for high resolution videos with an improved coding efficiency [16]. In HEVC, inter and intra picture prediction are implemented along with transform coding, motion estimation and motion compensation. In this work, we rely on the fact that HEVC is a lossy coding technique so we attempt to examine the effect of HEVC re-compression on the coding parameter information from which we can differentiate between single, double and triple compression [4]. The HEVC coding standard replaces the known macroblock structure in H.264 with a coding tree unit (CTU) consisting of LCUs (Largest Coding Units). These LCUs are divided into smaller sub-CUs in a recursive manner into 32x32, 16x16 or 8x8 blocks. These sub-CUs can be further divided into Prediction Units or PUs with sizes ranging from 4x4 to 32x32. The PUs represent the main block containing the prediction information, which is also sent to the decoder. PUs are divided into three main types, skipped PUs (S-PU), inter-predicted PUs (P-PUs) and intra-predicted PUs (I-PUs) [17]. In this work, statistics related to PUs and CUs are being used. PUs are then divided into Transform Units or TUs, which are used as an input to the Discrete Cosine Transform. HEVC supports four transform sizes for each N x N TU where N = 4,8,16 or 32. In intra prediction mode, HEVC defines 35 different intra prediction modes in which the mode with the lowest cost (Rate Distortion cost) is chosen. The intra-prediction modes are categorized into three types, DC Prediction Mode, Planar Prediction Mode and Angular Prediction Mode. In inter-prediction, either Discrete Cosine Transform (DCT) or Discrete Sine Transform (DST) is applied on TUs. The transform is applied on the residual coefficients which are calculated by finding the difference between the predicted block and the current block in the spatial domain after which quantization and entropy coding are applied.

3. Proposed System

We propose a classification system for detecting double and triple compression of video. Each video can be classified as compressed once, twice or three times as illustrated in Fig. 1. To the best of our knowledge, the detection of triple compression is novel, where the system is trained on singly, doubly and triply compressed videos.

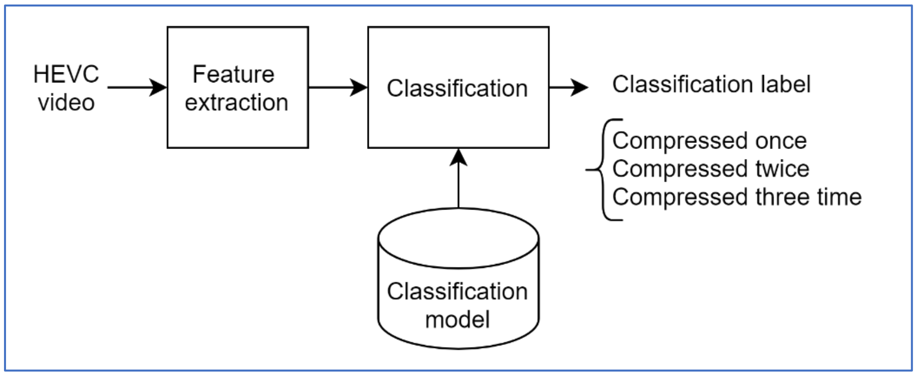

Fig. 1 Overview of proposed classification system

In the rest of this section, we introduce the proposed picture-level and GoP-level feature extraction solutions followed by the proposed classification arrangements.

3.1 Proposed feature extraction

In this work, we propose two sets of features; picture-level and GoP-level features. For both feature sets, the features extracted are based on the 8 CU features as listed in Table 1. Such CU features can be summarized into picture or GoP level features. The CU-level features initially extracted are the number of sub-CUs in each 64x64 CU, the ratio of motion vector difference bits, the number of CU bits, the percentage of intra, skipped and inter sub-CUs, the energy of the prediction residual for the CU, and the quantization parameter (QP) of each CU.

Table 1 Proposed features extracted from HEVC CUs

Feature ID	Feature Description
1	Number of sub-CU partitions
2	Ratio of MVD bits with reference to the total number of bits
3	Number of CU bits
4	Percentage of intra partitions in a CU
5	Percentage of skipped partitions in a CU
6	Percentage of inter (forward) partitions in a CU
7	Energy of prediction residual of CU
8	QP of CU

These features are first extracted from the encoder for single, double and triple compression and are then summarized to obtain features on a picture level. The features obtained on a picture level are further summarized to obtain features for each video GoP. Fig. 2 presents an overview of the proposed feature extraction phase.

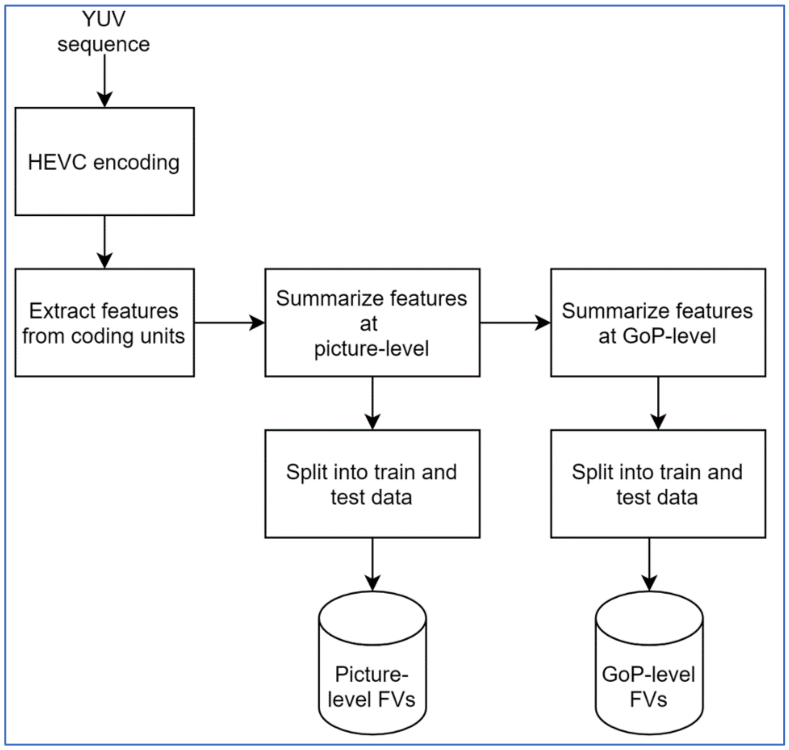
YUV sequence
HEVC encoding
Extract features from coding units
Summarize features at picture-level
Summarize features at GoP-level
Split into train and test data
Split into train and test data
Picture-level FVs
GoP-level FVs

Fig. 2 An Overview of the feature extraction phase

3.1.1 Picture-level Feature Set

In the picture-level feature set, the CU features corresponding to each picture are combined together and used to calculate the final set of picture-level features using the equations in Table 2. The feature vectors are summarized on a picture-level, meaning that for all CUs of a given picture, the mean and standard deviation are calculated to obtain the value of the feature on a picture-level. The estimated PSNR for each picture is also calculated and added to the feature set [18].

Table 2 Proposed Picture-level features

Feature ID	Feature Description
1	Mean of CU quantization parameter
2	Standard deviation of CU quantization parameter
3	Mean of number of sub-CU partitions
4	Standard deviation of number of sub-CU partitions
5	Mean of ratio of MVD bits
6	Standard deviation of ratio of MVD bits
7	Mean of number of CU bits
8	Standard deviation number of CU bits
9	Mean of percentage of intra partitions in a CU
10	Standard deviation of percentage of intra partitions in a CU
11	Mean of percentage of skipped partitions in a CU
12	Standard deviation of percentage of skipped partitions in a CU
13	Mean of percentage of inter partitions in a CU
14	Standard deviation of percentage of inter partitions in a CU
15	Mean of prediction residual energy of CU
16	Standard deviation of prediction residual energy of CU

The details of these feature variables are as follows. Feature ID-1 is computed using Equation (1):

$$\mu_{q} = {}^{1}/_{N} \sum_i Q_i(j) \qquad (1)$$

Where N is the total number of CUs per picture in the video sequence and $Q_i(j)$ is the quantization parameter of the ith CU in the current picture j. Using the same symbols of Equitation (1), feature ID-2 is computed using Equation (2):

$$\sigma_q = \sqrt{E\left[\left(Q_i(j) - \mu_q\right)^2\right]} \qquad (2)$$

Where E denotes the expected value.

Feature ID-3 is computer using Equation (3):

$$\mu_{cuParts} = {}^{1}/_{N} \sum_i A_i(j) \qquad (3)$$

Where N is the total number of CUs per picture in the video sequence and $A_i(j)$ is the number of sub-CUs of the ith CU in the current picture j. Using the same symbols of Equitation (3), feature ID-4 is computed using Equation (4):

$$\sigma_{cuParts} = \sqrt{E[(A_i(j) - \mu_{cuParts})^2]} \qquad (4)$$

Feature ID-5 is computed using Equation (5):

$$\mu_{mvd = 1/_N \sum_i M_i(j)} \qquad (5)$$

Where N is the total number of CUs per picture in the video sequence and $M_i(j)$ is the ratio of MVD bits of the ith CU in the current picture j. Using the same symbols of Equitation (5), feature ID-6 is computed using Equation (6):

$$\boldsymbol{\sigma_{mvd}} = \sqrt{\boldsymbol{E[(M_i(j) - \mu_{mdv})^2]}} \qquad \mathbf{(6)}$$

Feature ID-7 is computed using Equation (7):

$$\mu_{\boldsymbol{cuBits = 1/_N \sum_i B_i(j)}} \qquad \mathbf{(7)}$$

Where N is the total number of CUs per picture in the video sequence and $B_i(j)$ is the number of CU bits of the ith CU in the current picture j. Using the same symbols of Equitation (7), feature ID-8 is computed using Equation (8):

$$\boldsymbol{\sigma_{cuBits}} = \sqrt{\boldsymbol{E[(B_i(j) - \mu_{cuBits})^2]}} \qquad \mathbf{(8)}$$

Feature ID-9 is computed using Equation (9):

$$\mu_{\boldsymbol{cuIntra = 1/_N \sum_i I_i(j)}} \qquad \mathbf{(9)}$$

Where N is the total number of CUs per picture in the video sequence and $I_i(j)$ is the percentage of intra partitions of the ith CU in the current picture j. Using the same symbols of Equitation (9), feature ID-10 is computed using Equation (10):

$$\boldsymbol{\sigma_{cuIntra}} = \sqrt{\boldsymbol{E[(I_i(j) - \mu_{cuIntra})^2]}} \qquad \mathbf{(10)}$$

Feature ID-11 is computed using Equation (11):

$$\mu_{\boldsymbol{cuSkipped = 1/_N \sum_i S_i(j)}} \qquad \mathbf{(11)}$$

Where N is the total number of CUs per picture in the video sequence and $S_i(j)$ is the percentage of skipped partitions of the ith CU in the current picture j. Using the same symbols of Equitation (11), feature ID-12 is computed using Equation (12):

$$\boldsymbol{\sigma_{cuSkipped}} = \sqrt{\boldsymbol{E\left[\left(S_i(j) - \mu_{cuSkipped}\right)^2\right]}} \qquad \mathbf{(12)}$$

Feature ID-13 is computed using Equation (13):

$$\mu_{\boldsymbol{cuInter = 1/_N \sum_i P_i(j)}} \qquad \mathbf{(13)}$$

Where N is the total number of CUs per picture in the video sequence and $P_i(j)$ is the percentage of inter partitions of the ith CU in the current picture j. Using the same symbols of Equitation (13), feature ID-14 is computed using Equation (14):

$$\boldsymbol{\sigma_{cuInter}} = \sqrt{\boldsymbol{E[(P_i(j) - \mu_{cuInter})^2]}} \qquad \mathbf{(14)}$$

Feature ID-15 is computed using Equation (15):

$$\mu_{\boldsymbol{\epsilon = 1/_N \sum_i R_i(j)}} \qquad \mathbf{(15)}$$

Where N is the total number of CUs per picture in the video sequence and $R_i(j)$ is the sum of absolute residual values of the ith CU in the current picture j. Using the same symbols of Equitation (15), feature ID-16 is computed using Equation (16):

$$\boldsymbol{\sigma_{\epsilon}} = \sqrt{\boldsymbol{E[(R_i(j) - \mu_{\epsilon})^2]}} \qquad \mathbf{(16)}$$

3.1.2 GoP-level Feature Set

The Group of Pictures or GoP-level feature set is calculated from the picture-level feature set previously discussed, such that the features of pictures belonging to each GoP are combined and used to calculate the final GoP-level features. It is important to mention that throughout the experiments, the GoP size was set to 15 (each GoP contains 15 pictures) and thus each sequence of 100 pictures contains 7 GoPs or 7 feature vectors. Table 3 shows the equations used to calculate the feature variables are a GoP level.

Table 3 Proposed GoP-level feature

Feature ID	Feature Name
1	Mean of CU quantization parameter
2	Standard deviation of CU quantization parameter
3	Mean of number of sub-CU partitions
4	Standard deviation of number of sub-CU partitions
5	Mean of ratio of MVD bits
6	Standard deviation of ratio of MVD bits
7	Mean of number of CU bits
8	Standard deviation number of CU bits
9	Mean of percentage of intra partitions in a CU
10	Standard deviation of percentage of intra partitions in a CU
11	Mean of percentage of skipped partitions in a CU
12	Standard deviation of percentage of skipped partitions in a CU
13	Mean of percentage of inter partitions in a CU
14	Standard deviation of percentage of inter partitions in a CU
15	Mean of prediction residual energy of CU
16	Standard deviation of prediction residual energy of CU
17	Mean of estimated PSNR
18	Standard deviation of estimated PSNR

The details of these feature variables are as follows. Feature ID-1 is computed using Equation (17):

$$\mu_q = {}^{1}/_{G} \sum_j ({}^{1}/_{N} \sum_i Q_i(j)) \qquad (17)$$

Where N is the total number of CUs per picture in the video sequence and j is the index of picture in the current GoP (each GoP contains 15 pictures) . Qi(j) is the quantization parameter of the ith CU in the current picture j and G is the number of pictures per GoP. Using the same symbols of Equation (17), feature ID-2 is computed using Equation (18):

$$\sigma_q = \sqrt{E\left[\left(Q_i(j) - \mu_q\right)^2\right]} \qquad (18)$$

Where E denotes the expected value.

Feature ID-3 is computed using Equation (19):

$$\mu_{cuParts}={}^{1}/_{G}\sum_j({}^{1}/_{N}\sum_i A_i(j)) \qquad (19)$$

Where N is the total number of CUs per picture in the video sequence and j is the index of picture in the current GoP. Ai(j) is the number of sub-CUs of the ith CU in the current picture j and G is the number of pictures per GoP. Using the same symbols of Equation (19), feature ID-4 is computed using Equation (20):

$$\sigma_{cuParts} = \sqrt{E[(A_i(j) - \mu_{cuParts})^2]} \qquad (20)$$

Feature ID-5 is computed using Equation (21):

$$\mu_{mvd}={}^{1}/_{G}\sum_j({}^{1}/_{N}\sum_i M_i(j)) \qquad (21)$$

Where N is the total number of CUs per picture in the video sequence and j is the index of picture in the current GoP. Mi(j) is the ratio of MVD bits of the ith CU in the current picture j and G is the number of pictures per GoP. Using the same symbols of Equation (21), feature ID-6 is computed using Equation (22):

$$\sigma_{mvd} = \sqrt{E[(M_i(j) - \mu_{mvd})^2]} \qquad (22)$$

Feature ID-7 is computed using Equation (23):

$$\mu_{cuBits}={}^{1}/_{G}\sum_j({}^{1}/_{N}\sum_i B_i(j)) \qquad (23)$$

Where N is the total number of CUs per picture in the video sequence and j is the index of picture in the current GoP. Bi(j) is the number of CU bits of the ith CU in the current picture j and G is the number of pictures per GoP. Using the same symbols of Equation (23), feature ID-8 is computed using Equation (24):

$$\sigma_{cuBits} = \sqrt{E[(B_i(j) - \mu_{cuBits})^2]} \qquad (24)$$

Feature ID-9 is computed using Equation (25):

$$\mu_{cuIntra}={}^{1}/_{G}\sum_j({}^{1}/_{N}\sum_i I_i(j)) \qquad (25)$$

Where N is the total number of CUs per picture in the video sequence and j is the index of picture in the current GoP. Ii(j) is the percentage of intra partitions of the ith CU in picture j and G is the number of pictures per GoP. Using the same symbols of Equation (25), feature ID-10 is computed using Equation (26):

$$\sigma_{cuIntra} = \sqrt{E[(I_i(j) - \mu_{cuIntra})^2]} \qquad (26)$$

Feature ID-11 is computed using Equation (27):

$$\mu_{cuSkipped}={}^{1}/_{G}\sum_j({}^{1}/_{N}\sum_i S_i(j)) \qquad (27)$$

Where N is the total number of CUs per picture in the video sequence and j is the index of picture in the current GoP.

Si(j) is the percentage of skipped partitions of the ith CU in picture j and G is the number of pictures per GoP. Using the same symbols of Equation (27), feature ID-12 is computed using Equation (28):

$$\sigma_{\text{cuSkipped}} = \sqrt{E\left[\left(S_i(j) - \mu_{\text{cuSkipped}}\right)^2\right]} \qquad (28)$$

Feature ID-13 is computed using Equation (29):

$$\mu_{\text{cuInter}} = {}^1/_G \sum_j ({}^1/_N \sum_i P_i(j)) \qquad (29)$$

Where N is the total number of CUs per picture in the video sequence and j is the index of picture in the current GoP.

Pi(j) is the percentage of inter partitions of the ith CU in picture j and G is the number of pictures per GoP. Using the same symbols of Equation (29), feature ID-14 is computed using Equation (30):

$$\sigma_{\text{cuInter}} = \sqrt{E[(P_i(j) - \mu_{\text{cuInter}})^2]} \qquad (30)$$

Feature ID-15 is computed using Equation (31):

$$\mu_{\epsilon} = {}^1/_G \sum_j ({}^1/_N \sum_i R_i(j)) \qquad (31)$$

Where N is the total number of CUs per picture in the video sequence and j is the index of picture in the current GoP.

Ri(j) is the sum of absolute residual values of the ith CU in picture j and G is the number of pictures per GoP. Using the same symbols of Equation (31), feature ID-16 is computed using Equation (32):

$$\sigma_{\epsilon} = \sqrt{E[(R_i(j) - \mu_{\epsilon})^2]} \qquad (32)$$

Feature ID-17 is computed using Equation (33):

$$\mu_{\text{PSNR}} = {}^1/_G \sum_j P(j)) \qquad (33)$$

Where j is the index of picture j in the current GoP. P(j) is the estimated PSNR of picture j and G is the number of pictures per GoP. Lastly, using the same symbols of Equation (33), feature ID-18 is computed using Equation (34):

$$\sigma_{\text{PSNR}} = \sqrt{E[(P(j) - \mu_{\text{PSNR}})^2]} \qquad (34)$$

3.2. Classification

As mentioned in the previous section, feature variables are extracted from coded HEVC CUs. The variables are then summarized into picture and Group of Pictures (GoP) level feature vectors. Classification is performed using a Random Forest classifier [19-22] as well as a bi-LSTM network [23, 24]. Fig. 3 shows the classification system setup used for both RF and LSTM classifiers. In the RF classifier, the feature vectors are normalized by computing their z-scores. After the training phase is completed, test feature vectors are classified into single, double or triple compression (i.e three classification classes). This classification can be performed at a picture, a GoP or a video level. Results of both classifiers, and both feature sets for double and triple compression detection are reported in the experimental results section.

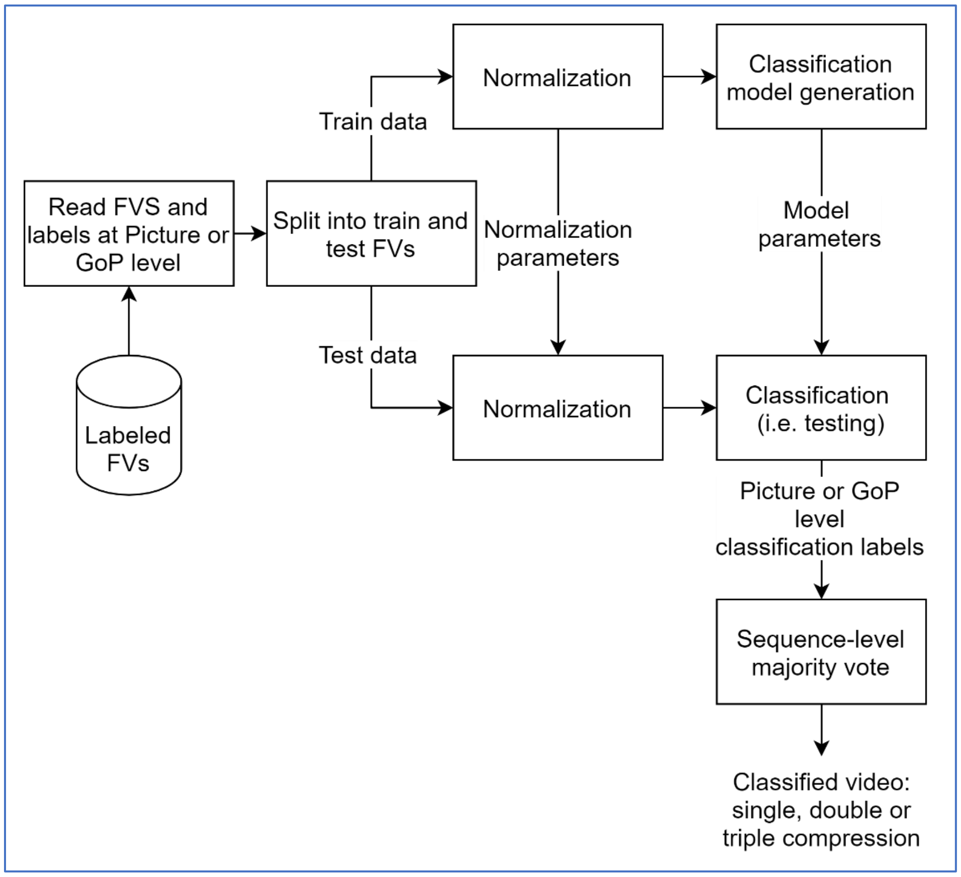
Read FVS and labels at Picture or GoP level
Labeled FVs
Split into train and test FVs
Train data
Test data
Normalization
Normalization
Normalization parameters
Classification model generation
Model parameters
Classification (i.e. testing)
Picture or GoP level classification labels
Sequence-level majority vote
Classified video: single, double or triple compression

Fig. 3 HEVC re-compression detection using the proposed classification solution

3.2.1 Sequence-level classification using Random Forest (RF)

Random Forests are part of the ensemble machine learning algorithms where a group of weak classifiers combine to form a strong classifier. The algorithm is based on the bagging approach where multiple decision trees are combined to create a new model with low variance in terms of classification and thus increase classification accuracy. The random forest approach is a combination consisting of a set of decision trees used for classification. The Random Forest is one of the most used machine learning algorithms as it is known to produce great classification results even without hyper-parameter tuning.

In this work, the algorithm is trained and tested twice, once using the picture-level features and once using the GoP-level features. The obtained classification accuracies are on a picture-level and GoP-level but since we are interested in sequence-level accuracies, majority voting is then used. In sequence-level majority voting, each sequence is classified as single, double or triple compression based on the majority predicted label of its pictures in case of picture-level features and the majority predicted label of its GoPs in case of GoP-level features.

3.2.2 Sequence-level classification using a bi-LSTM Network

Long Short-Term Memory networks, also known as LSTMs, are a special version of the Recurrent Neural Network or RNN used for the prediction of sequential data. The bi-LSTM is another version of the LSTM network where both previous and following data samples are used in the prediction of a current input. In other words, the bi-LSTM relies on past and future data to predict current data. In our solution, we propose the use of the bi-LSTM network for recompression detection in HEVC coded videos using the fact that the features used to classify one picture will depend on that of the previous and future pictures.

Picture-level and GoP-level features are extracted from the video sequences and used as input to the bi-LSTM network, directly producing the sequence level accuracies for re-compression detection. The diagram in Fig.4 illustrates the network architecture. Basically, features are extracted on a picture-level and GoP-level from HEVC videos, after which the Sequence Input layer involves inputting the video sequences to the network. The bi-LSTM layer has hidden unites needed to exploit the temporal dependencies between the input feature vectors. The bi-LSTM layer is followed by a dropout layer that randomly sets input elements to zero with a 50% probability. This is need to reduce overfitting in the training process. Lastly, the output is fed into a fully connected layer with 3 classification classes.

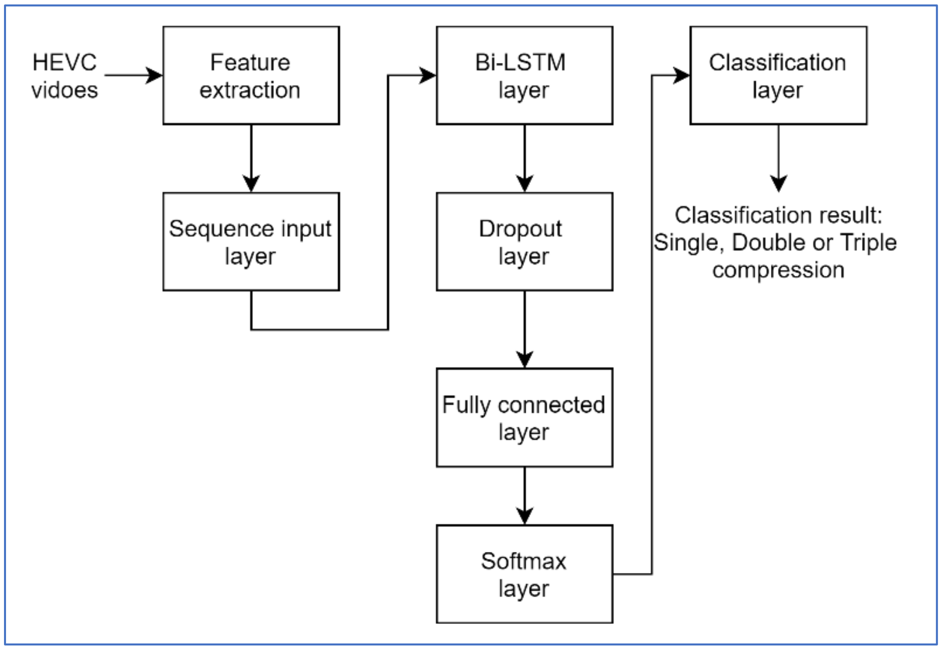

Fig. 4 Bi-LSTM Network Architecture

5. Experimental Setup and Results

In this section, we introduce the video dataset used for training and classification, we introduce the experimental setup and present the experimental results.

5.1 Dataset

In the experiments conducted for HEVC double and triple compression detection, 26 YUV420 sequences are used, thirteen of which are 1080p and thirteen are 720p. All the sequences used are collected from the online video test media database[1].

Table 4 presents a list of the YUV sequences used to generate the dataset, which is the same set of sequences used in [9]. To keep the spatial resolution the same for all the YUV sequences, the 1080p sequences are resized to 720p (1280x720) in the spatial domain in a lossless manner so that there are no traces of lossy compression. Also, to increase the sample size, each YUV sequence is divided into multiple non-overlapping sequences consisting of 100 pictures each. As a result, a total of 127 shorter YUV sequences are obtained which are then used throughout the experiments. To obtain double compressed videos, the raw YUV sequences are first compressed and then decompressed and re-encoded using the same bitrate. For triple compression, the same recompressions of double compression are conducted following by a third compression of the same bitrate as illustrated in Fig. 5

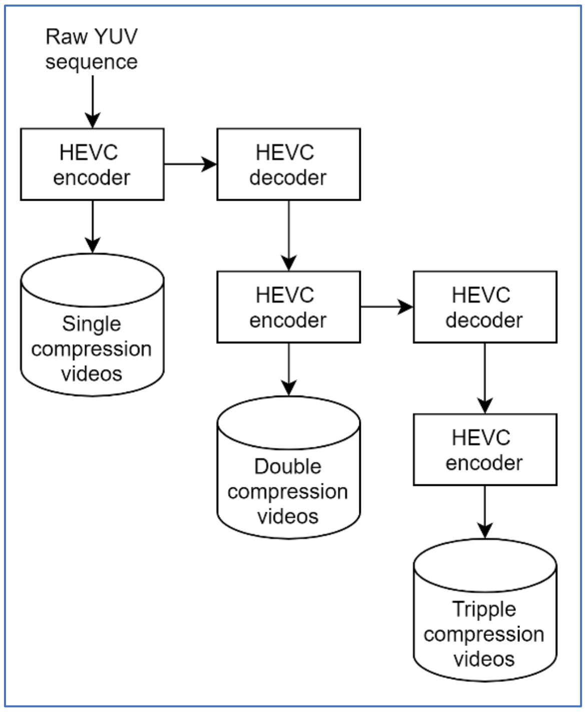
Raw YUV sequence
HEVC encoder
HEVC decoder
Single compression videos
HEVC encoder
HEVC decoder
Double compression videos
HEVC encoder
Tripple compression videos

Fig. 5 Preparing the video dataset with single, double and triple compression

[1] YUV sequences available at the online database: http://media.xiph.org/video/derf/.

All the obtained video sequences from the re-compressions are divided into two groups for training and testing according to the splitting used in the previous work as shown in Table 5 [9]. This splitting structure ensures that no two YUV sequences originally belonging to the same sequence are included in both the training and testing datasets. As a result of the split, out of the 127 sequences, the training set will consist of 77 sequences and the testing set will consist of the remaining 50 sequences.

For each class (i.e. single, double and triple compression), a total of 889 (127 sequences x 7 FVs) feature vectors are obtained which are divided into training and testing set such that the training set consists of 539 (77 sequences x 7 FVs) samples and the testing set consists of 350 (50 sequences x 7 FVs) samples per class. The extracted GoP-level features are then used to train and test a Random Forest classifier and a bi-LSTM network.

Table 4 YUV sequences used to generate the dataset

1080p YUV Sequences	**720p YUV Sequences**
blue_sky, crowd_run, pedestrian_area, riverbed, rush_field_cuts, rush_hour, snow_mnt, speed_bag, station2, sunflower, touchdown_pass, tractor, west_wind_easy	ducks_take_off, FourPeople, in_to_tree, Johnny, KristenAndSara, mobcal, old_town_cross, park_joy, parkrun, shields, stockholm, vidyo1, vidyo3

Table 5 YUV sequences used in training and testing

Train Sequences	**Test Sequences**
stockholm, KristenAndSara, Johnny, shields, vidyo3, ducks_take_off, sunflower, rush_hour, crowd_run, rush_field_cuts, blue_sky, speed_bag, touchdown_pass, west_wind_easy, pedestrian_area, station2	in_to_tree, park_joy, old_town_cross, vidyo1, parkrun, mobcal, FourPeople, snow_mnt, tractor, riverbed.

5.2 Classification results

This section discusses the results obtained from the proposed compression detection techniques. The proposed features are extracted from the videos of the above dataset and used for classifying a video into single, double and triple compression. The results are repeated using both of the classification arrangements introduced in Section 4. The double compression detection results have been compared to three of the existing solutions in literature. The performance measures used in all experiments are accuracy, precision and recall. The accuracy or classification rate refers to the percentage of correctly classified instances and is calculated using the equation below.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} * 100\% \qquad (35)$$

Where TP refers to True Positive count, TN refers to True Negative count, FP refers to False Positive count and FN refers to False Negative count. Precision is a measure of exactness or quality. It is the percentage of test cases that are classified as X and have a true label of X. It is also referred to as the exactness of a classifier and is calculated using the below equation.

$$Precision = \frac{TP}{TP + FP} \quad (36)$$

Finally, recall is a measure of completeness or quantity. It is defined as the percentage of test cases that have a true label of X and are actually labelled as X and is calculated using the equation below.

`

$$Recall = \frac{TP}{TP+FN} \quad (37)$$

In this set of experiments, the results are reported for four different bitrates (800,1000,1200,1400) kbps. For each of the four bitrates, experiments are conducted for features extracted on a picture and a GoP level for each of the video sequences.

The solver used for training the bi-LSTM network is the Adam optimizer with an initial learn rate of 1e-4 and a mini batch size of 16. As for the Random Forest classifier, 20 trees are grown in the training and testing.

Table 6 provides a detailed comparison between the results obtained by our proposed solution and the results found in existing literature. We presents results using our proposed picture-level and GoP-level feature extraction solutions. The results obtained from the proposed solution are noticeably higher than those reported in Jiang's, Xu's and Liang's methods in [9], [10] and [11]. In Fig. 6, a comparison is done for the average accuracies obtained across each of the proposed and existing methods for the conducted experiments, showing that the highest average accuracy is obtained using the proposed GoP-level feature set and bi-LSTM network.

Picture-level features using Random Forest classifier and bi-LSTM network produced good but lower results compared to GoP-level features. The highest results are obtained when using GoP-level features and bi-LSTM network where a 100% classification rate is obtained for all the different bitrates. This can be justified by the fact that GoP-level features are richer than single picture-level features. Additionally, the use of bi-LSTM networks exploits the temporal dependencies between feature vectors in a video sequence. In the RF classifier on the other hand, a feature vectors is classified in isolation of surrounding feature vectors.

Table 6 Classification results of double compression detection with comparison to existing work

Bitrate	PICTURE-LEVEL FEATURES		GOP-LEVEL FEATURES		EXISTING WORK		
B1,B2	Proposed Solution (RF)	Proposed Solution (bi-LSTM)	Proposed Solution (RF)	Proposed Solution (LSTM)	Jiang Method [9]	Xu Method [10]	Liang Method [11]
800,800	94%	98%	98%	100%	93%	92.5%	85%
1000,1000	96%	96%	97%	100%	95%	93%	87%
1200,1200	96%	96%	97%	100%	95.5%	92.5%	86.25%
1400,1400	96%	96%	98%	100%	96%	92%	86.25%
Average	**95.5%**	**96.5%**	**97.5%**	**100%**	**94.88%**	**92.5%**	**86.13%**

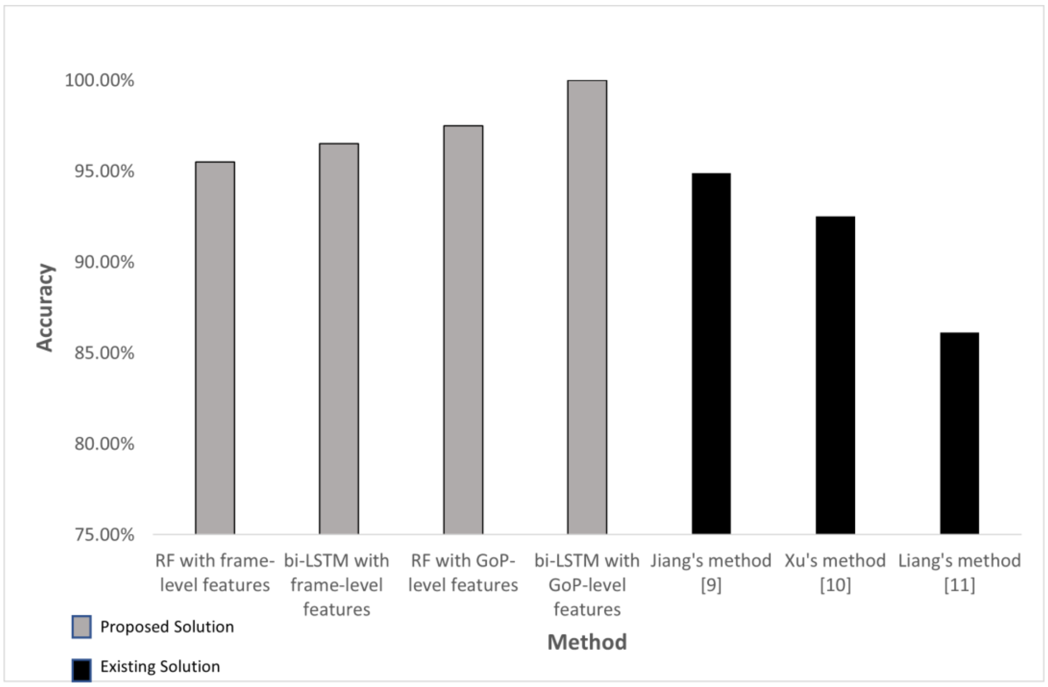

Fig. 6 A comparison of the average accuracies obtained for double compression detection

In Table 7, the confusion matrices for the lowest bitrate (800 kbps) are reported for each of the four proposed methods as this bitrate is the most challenging amongst all. This is because low bitrates cause higher compression and higher information loss leading to less representative feature vectors.

Table 7 Confusion matrices, precision and recall for double compression detection for bitrate=800

		Confusion Matrix			Precision	Recall
Picture-level Features	Proposed Solution (RF)		C1	C2	0.81	0.95
		C1	95.32	4.68		
		C2	22.38	77.62		
	Proposed Solution (bi-LSTM)		C1	C2	0.96	1
		C1	48.00	0.00		
		C2	2.00	50.00		
GoP-level Features	Proposed Solution (RF)		C1	C2	0.85	0.98
		C1	98.29	1.71		
		C2	17.14	82.86		

			C1	C2		
	Proposed Solution (bi-LSTM)	C1	100.0	0.00	1	1
		C2	0.00	100.0		

The same experiments conducted for double compression detection are implemented again for double and triple compression detection with promising results being obtained for each of the four bitrates. To the best of our knowledge, no existing literature addresses the problem of detecting double and triple compression. Again, highest results were achieved when using GoP-level features and bi-LSTM network for classification with results ranging from 98% to 98.7%. Again, this can be justified by the fact that GoP-level features are richer than single picture-level features. Moreover, the use of bi-LSTM networks exploits the temporal dependencies between feature vectors in a video sequence. In the RF classifier on the other hand, a feature vectors is classified in isolation of surrounding feature vectors.

Table 8 presents a detailed comparison of the results obtained for each of the four bitrates using the different feature sets and classifiers.

Table 8 Results of detecting double and triple compression using three classes

	FRAME-LEVEL FEATURES		GOP-LEVEL FEATURES	
B1,B2,B3	Proposed Solution (RF)	Proposed Solution (LSTM)	Proposed Solution (RF)	Proposed Solution (bi-LSTM)
800,800,800	82.67%	94%	76.67%	98%
1000,1000,1000	79.33%	92%	84.00%	98.7%
1200,1200,1200	75.33%	92.67%	86.67%	98.7%
1400,1400,1400	80%	94%	82.67%	98.7%
Avg. Accuracy	**79.33%**	**93.17%**	**82.5%**	**98.55%**

Table 9 shows the confusion matrices, precision and recall obtained for the most challenging case with the lowest bitrate of 800 kbps.

Table 9 Confusion matrices, precision and recall for double and triple compression detection for bitrate=800

		Confusion Matrix				Precision	Recall
Picture-level Features	Proposed Solution (RF)		C1	C2	C2	0.67	0.51
		C1	95.84	3.50	0.66		
		C2	21.68	53.46	24.86		
		C3	8.18	40.36	51.46		
	Proposed Solution (bi-LSTM)		C1	C2	C2	1	0.85
		C1	33.30	0.00	0.00		
		C2	0.00	27.30	0.00		
		C3	0.00	6.00	33.30		
GoP-level Features	Proposed Solution (RF)		C1	C2	C2	0.74	0.56
		C1	98.29	1.43	0.29		
		C2	19.43	61.43	19.14		
		C3	6.29	37.43	56.29		
	Proposed Solution (bi-LSTM)		C1	C2	C2	1	0.98
		C1	33.30	1.30	0.00		
		C2	0.00	31.30	0.00		
		C3	0.00	0.70	33.3		

Fig. 7 then shows a comparison for the average accuracies obtained for each method across the four different bitrates, indicating that the highest average accuracy for triple compression detection is also obtained when using GoP-level features and bi-LSTM deep learning network.

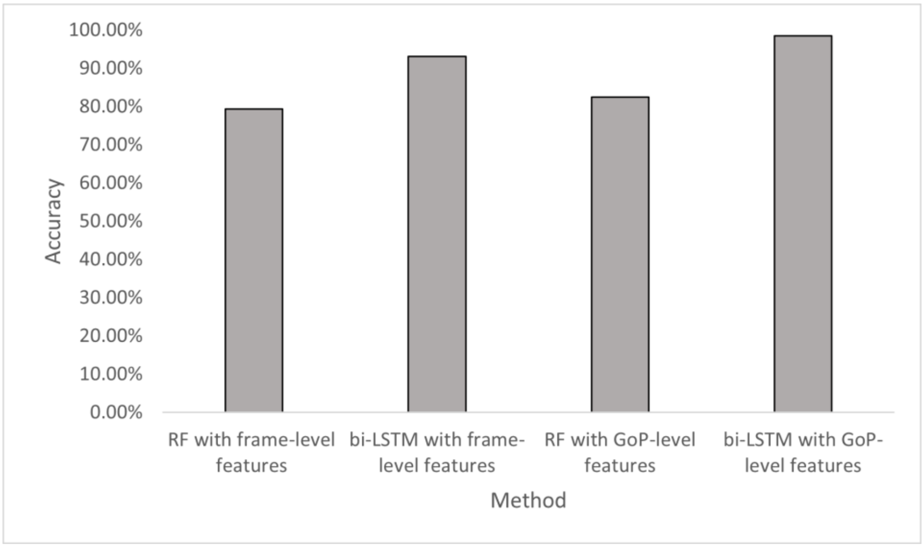

Fig. 7 A comparison for the average accuracies obtained in double and triple compression detection

A 98% classification accuracy is achieved for the lowest and most challenging bitrate of 800 kbps and a 98.7% accuracy is achieved for the remaining three bitrates. Despite the promising results obtained, the classification rate of double and triple compression detection is still lower than that of double compression detection in all the experiments conducted due to having three classes, making the classification more challenging. More specifically, in the detection of double compression only, the best classification accuracy at picture and GoP levels are 96.5% and 100% respectively. Whereas, in the novel detection of double and triple compression, the best classification accuracy at picture and GoP levels are 93.2% and 98.6% respectively. Clearly, using three classes for double and triple compression detection instead of two will result in lower classification rates as distinguishing between the second and third compression becomes much more difficult compared to distinguishing between single (original) and triple compression. This evident from the results reported in the confusion matrices. The results obtained have not been compared to existing solutions as to the extent of our knowledge, no existing literature was found to tackle the problem of triple compression detection in HEVC coded videos.

6. Conclusion

In this work, we developed a system that improves on the existing solutions of double compression detection in HEVC videos. We also developed a novel system for double and triple compression detection with very promising results. Two feature sets have been used, picture-level and GoP-level features, and four different bitrates have been tested. The results obtained for double compression detection have been compared to three of the existing solutions in the literature. Highest results were achieved when using the GoP-level features where a 97.5% average accuracy is achieved using the RF classifier and a 100% average accuracy is achieved using the bi-LSTM network for double compression detection. The same experiments were conducted for triple compression detection where to the best of our knowledge, no existing solutions have been implemented to tackle this problem. The highest average accuracy of 98.55% was achieved for triple compression detection when using the GoP-level features with a bi-LSTM network. Clearly, GoP-level features are richer than single picture-level features and use of bi-LSTM networks exploits the temporal dependencies between feature vectors in a video sequence. In the RF classifier on the other hand, a feature vectors is classified in isolation of surrounding feature vectors, which results in less accurate classification. From the results obtained, we can clearly state the effectiveness of the proposed solution for both double and triple compression detection in HEVC coded videos when having the same recompression bitrate.

Conflict of Interest: Authors S. Youssef and T. Shanableh declare that they have no conflict of interest.

References

1. Stamm MC, Wu M, Liu KJR. Information Forensics: An Overview of the First Decade. IEEE Access. 2013;1:167-200. https://doi.org/10.1109/ACCESS.2013.2260814
2. Huang M, Wang R, Xu J, Xu D, Li Q, editors. Detection of Double Compression for HEVC Videos Based on the Co-occurrence Matrix of DCT Coefficients. Digital-Forensics and Watermarking; Cham: Springer International Publishing; 2016. https://doi.org/10.1007/978-3-319-31960-5_

3. Jia RS, Li ZH, Zhang ZZ, Li DD. Double HEVC Compression Detection with the Same QPs Based on the PU Numbers. ITM Web Conf; 2016. https://doi.org/10.1051/itmconf/2016/0702010
4. Li Qian. Detection of Double Compression in HEVC Videos Based on TU Size and Quantized DCT Coefficients. IET Information Security. 2018;13:1049. https://doi.org/10.1049/iet-ifs.2017.0555
5. Yu L, Yang Y, Li Z, Zhang Z, Cao G. HEVC double compression detection under different bitrates based on TU partition type. EURASIP Journal on Image and Video Processing. 2019;1(1):67. https://doi.org/10.1186/s13640-019-0468-x
6. He P, Jiang X, Sun T, Wang S. Double compression detection based on local motion vector field analysis in static-background videos. Journal of Visual Communication and Image Representation. 2016;35:55-66. https://doi.org/10.1016/j.jvcir.2015.11.014
7. Stamm MC, Lin WS, Liu KJR. Temporal Forensics and Anti-Forensics for Motion Compensated Video. IEEE Transactions on Information Forensics and Security. 2012;7(4):1315-29. https://doi.org/10.1109/TIFS.2012.2205568
8. Vazquez-Padin D, Fontani M, Bianchi T, Comesana P, Piva A, Barni M, editors. Detection of video double encoding with GOP size estimation. IEEE International Workshop on Information Forensics and Security (WIFS); 2012. https://doi.org/10.1109/WIFS.2012.6412641
9. Jiang X, He P, Sun T, Wang R. Detection of Double Compressed HEVC Videos Using GOP-Based PU Type Statistics. IEEE Access. 2019;7:95364-75. https://doi.org/10.1109/ACCESS.2019.2928857
10. Xu Q, Sun T, Jiang X, Dong Y. HEVC Double Compression Detection Based on SN-PUPM Feature. Digital Forensics and Watermarking. Cham: Springer International Publishing; 2017. https://doi.org/10.1007/978-3-319-64185-0_1
11. Liang X, Li Z, Yang Y, Zhang Z, Zhang Y. Detection of Double Compression for HEVC Videos With Fake Bitrate. IEEE Access. 2018;6:53243-53. https://doi.org/10.1109/ACCESS.2018.2869627
12. Q. Li, S. Chen, S. Tan, B. Li and Huang J. One-Class Double Compression Detection of Advanced Videos Based on Simple Gaussian Distribution Model. IEEE Transactions on Circuits and Systems for Video Technology; 2021. doi: 10.1109/TCSVT.2021.3069254
13. He P., Li H., Wang H., Wang S., Jiang X. and Zhang R. Frame-wise Detection of Double HEVC Compression by Learning Deep Spatio-temporal Representations in Compression Domain. IEEE Transactions on Multimedia; 2020. doi: 10.1109/TMM.2020.3021234
14. Jiang X., Xu Q., Sun T., Li B. and He P. Detection of HEVC Double Compression With the Same Coding Parameters Based on Analysis of Intra Coding Quality Degradation Process. IEEE Transactions on Information Forensics and Security. 2020;15:250-263. http://doi.org/10.1109/TIFS.2019.2918085
15. Yao H., Ni R. and Zhao Y. Double compression detection for H.264 videos with adaptive GOP structure. Multimed Tools and Applications. 2020;79:5789–5806. https://doi.org/10.1007/s11042-019-08306-5

16. Sullivan GJ, Ohm J, Han W, Wiegand T. Overview of the High Efficiency Video Coding (HEVC) Standard. IEEE Transactions on Circuits and Systems for Video Technology. 2012;22(12):1649-68. https://doi.org/10.1109/TCSVT.2012.2221191

17. Elrowayati AA, Abdullah MFL, Manaf AA, Alfagi AS. Tampering detection of double-compression with the same quantization parameter in HEVC video streams. IEEE International Conference on Control System, Computing and Engineering (ICCSCE); 2017. https://doi.org/10.1109/ICCSCE.2017.8284400

18. 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. https://doi.org/10.1016/j.image.2015.02.008

19. Antonio C, Jamie S, Ender K. Decision Forests: A Unified Framework for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning; 2012. https://doi.org/10.1561/0600000035

20. Breiman L. Random Forests. Machine Learning. 2001;45(1):5-32. https://doi.org/10.1023/A:1010933404324

21. Geurts P, Ernst D, Wehenkel L. Extremely randomized trees. Machine Learning. 2006;63(1):3-42. https://doi.org/10.1007/s10994-006-6226-1

22. Breiman L, Friedman J, Olshen R, Stone CJ. Classification and Regression Trees. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 1983;1:14-23. http://doi.org/10.1002/widm.8

23. Hochreiter S., Schmidhuber J. Long short-term memory. Neural computation. 1997:9(8):1735-80. https://doi.org/10.1162/neco.1997.9.8.1735

24. Schuster M, Kuldip P. Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing. 1997;45:2673-81. http//doi.org/10.1109/78.650093

[1] **Seba Youssef** is a Computer Engineering Masters student at the American University of Sharjah. She received her Bachelor degree in Computer Engineering from the American University in Cairo in 2017. While pursuing her Bachelor degree, she worked on multiple research projects in the field of computer vision, image processing and deep learning. She also worked at the University Academic Computing Technolgies office where she aided in the development of university related systems. During her Masters degree, she worked as a teaching assistant for a number of undergraduate courses. She also worked on research projects in the fields of video processing, big data and data mining. Her research interests include image and video processing, data analytics and machine learning.

[2]**Tamer Shanableh** a senior member of the IEEE and a professional engineer. Was born in Scotland, UK in 1972. He studied at the University of Essex where he received his Ph.D. in electronic systems engineering in 2002 and his MSc in software engineering in 1998.

He then worked as a senior research officer at the University of Essex for three years, during which, he collaborated with BTexact on inventing video transcoders. He then joined Motorola UK Research Labs and contributed to establishing a new profile within the ISO/IEC MPEG-4 known as the Error Resilient Simple Scalable Profile. He joined the American University of Sharjah in 2002 and is currently a professor of computer science. During the summer breaks, Dr. Shanableh worked as a visiting professor at Motorola Labs in five different years. He spent his sabbatical leave as a visiting academic at the Multimedia and Computer Vision and Lab at Queen Mary, University of London, U.K. Dr. Shanableh authored more than 80 publications and has 6 patents. His research interests include digital video processing and pattern recognition.

Declarations

Funding: No funding sources

Conflicts of interest: Authors S. Youssef and T. Shanableh declare that they have no conflict of interest.

Availability of data and material: All data used has been collected from http://media.xiph.org/video/derf/.

Code availability: Not applicable

Ethics approval: No ethical approval/waiver is required

Consent to participate: Not applicable

Consent for publication: Not applicable