Multimedia Tools and Applications (2019) 78:32735–32754
https://doi.org/10.1007/s11042-018-6882-8

Predicting split decisions of coding units in HEVC video compression using machine learning techniques

Mahitab Hassan [1,2] · Tamer Shanableh [1]

Received: 24 January 2018 / Revised: 24 September 2018 / Accepted: 13 November 2018 /
Published online: 23 November 2018

Abstract
In this work, we propose to reduce the complexity of HEVC video encoding by predicting the split decisions of coding units. We use a sequence-dependent approach in which a number of frames belonging to the video being encoded are used for generating a classification model. At each coding depth of the coding units, features representing the coding unit at that particular depth are extracted from both the present and previously encoded coding units. The feature vectors are then used for generating a dimensionality reduction model and a classification model. The generated models at each coding depth are then used to predict the split decisions of subsequent coding units. Stepwise regression, random forest reduction and principal component analysis are used for dimensionality reduction; whereas, polynomial networks and random forests are utilized for classification. The proposed solution is assessed in terms of classification accuracy, BD-rate, BD-PSNR and computational time complexity. Using seventeen video sequences with four different classes of resolution, an average classification accuracy of 86.5% is reported for the proposed classification system. In comparison to regular HEVC coding, the proposed solution resulted in a BD-rate loss of 0.55 and a BD-PSNR of −0.02 dB. The average reported computational complexity reduction is found to be 39.2%.

Keywords HEVC · Pattern recognition · Video compression

1 Introduction

The High Efficiency Video Coding (HEVC) standard is one of the successors of the well-known MPEG-4 AVC (H.264 or MPEG-4 Part 10). It is designed to target various

✉ Tamer Shanableh
tshanableh@aus.edu

Mahitab Hassan
mahitab.hassan@alumni.aus.edu; mahitab.hassan@ae.ibm.com

[1] Department of Computer Science and Engineering, American University of Sharjah, Sharjah, United Arab Emirates

[2] Present address: IBM Cloud, Dubai, United Arab Emirates

applications, especially those dealing with Ultra High Definition (HD) content. The HEVC project was formally initiated when a joint Call for Proposals was issued by the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG) in January 2010 [30]. The prime focus was directed towards significantly improving the compression performance relative to existing standards.

After its completion in January 2013, the HEVC standard provides twice the compression capabilities as that offered by its predecessor. Given that the appropriate encoder settings are used, around 50% bit-rate reduction is possible, while maintaining minimal video quality level loss [21]. However, this coding efficiency is introduced at the cost of increasing the encoding computational complexity, which can reach up to 40% more than that of H.264/AVC [34].

Among other factors, both the enhanced compression efficiency and the increased encoding computational complexity can be attributed to HEVC's usage of flexible partitioning structures. HEVC uses quad-tree Coding Tree Units (CTUs), Prediction Units (PUs), and Residual Quad-Trees (RQTs) rather than macroblocks (MBs). In order to achieve the best configuration in terms of selecting the optimal partitioning structure, an exhaustive rate-distortion optimization (RDO) process takes place, which is the main reason behind the intensification of the computational complexity. Most of the encoding time involves recursively repeating the RDO process at each Coding Unit (CU) depth level for each structure, where every combination of encoding structure is tested and the one that minimizes the rate-distortion (RD) cost is chosen [31].

Several early termination algorithms for optimizing the encoding process in HEVC can be found in the literature, where their aim is to reduce the computational complexity while any minimizing performance degradation. Among many, some approaches utilize the textural or structural characteristics of a given CU [1, 7, 9, 12, 22, 24, 28, 35, 38, 40, 41], while others use machine learning techniques [4, 6, 11, 14–17, 29, 36, 39]. The optimizations are not limited to HEVC inter-coding as some also considered enhancing intra-coding [16, 22, 24, 28]. In this field of research, utilizing machine learning techniques as a tool to minimize RD efficiency losses is limited, and most algorithms proposed do not achieve superior results in terms of computation complexity reduction without introducing significant video quality level losses.

In this work, we use a sequence-dependent approach to model the relationship between CU feature variables and split decisions. The feature variables are extracted from both the present CU and its surrounding spatial and temporal CUs. Additionally, we use dimensionality reduction techniques for the three CU depths of 64×64, 32×32 and 16×16. This is needed to reduce the number of extracted features. The feature extraction and modeling is also performed at three CU coding depths. We use stepwise regression, random forest reduction and principal component analysis (PCA) for dimensionality reduction. Moreover, we utilize polynomial networks and random forests for classification.

This paper is organized as follows. Section 2 presents a review of algorithms proposed in the literature that are reduced the encoding computational complexity. The overall CU split prediction system proposed is overviewed in Section 3. The feature extraction process and dimensionality reduction are discussed in detail in Section 4 in addition to the classification tools and arrangements used in this work. The experimental setup and experimental results are presented in Section 5. Lastly, Section 6 concludes the paper.

2 Related work

As mentioned earlier, HEVC introduced significant coding efficiency improvements at the cost of increasing the computational complexity. Therefore, existing research work is conducted to limit this computational complexity whilst minimizing the adverse effect on compression efficiency. The work reported in [1, 7, 9, 12, 22, 24, 28, 35, 38, 40, 41] investigated the textural or structural characteristics of CUs at a given CU depth to optimize the HEVC encoding procedure. [41] proposed an inter-prediction optimization scheme, where the CTU structure is analyzed in a reverse order. Alternatively, a subjective-driven complexity control approach is presented in [7], which examines the relationship between visual distortion and maximum depth of all largest CUs. Another complexity control algorithm is proposed in [12], where an early termination condition is defined at each CU depth based on the content of the video sequence being encoding, the configuration files and the target complexity.

In [40], the authors present a hierarchical structure-based fast mode decision scheme. A fast CU decision algorithm is presented in [38], where the coded block flag and RD costs are checked to determine if intra- and inter- PUs can be skipped. In [35], a two-layered motion estimation based fast CU decision process is proposed, which uses the sum of absolute differences (SAD) estimation to extract the SAD costs for a CU and its sub-CUs. [22] speeds up the HEVC intra-coding process mainly by using encoded CU depths and RD costs of co-located CTU to predict both the current CU's depth search range and the RD cost for CU splitting termination. Local texture descriptors or image characteristics were used in [9, 24, 28] to allow faster CU size selection. A spatiotemporal based CU encoding technique is explored in [1], where sample-adaptive-offset (SAO) parameters were utilized to predict the textural complexity of the CU being encoded. The work in [5] introduces an interesting approach for predicting CU splitting based on deep learning using a reinforced learning algorithm. The algorithm is also capable of predicting the reduction in rate-distortion cost. The solution is applied to all-intra configuration and results in a BD-rate loss of 2.5%. In [8], the authors proposed an offline training algorithm based on random forests to predicting early termination of CU splitting. Neighboring CU sizes are also used in determining the depth of the current CU. The algorithm is applied to all-intra mode and reported a complexity reduction of 48.3% with a BD-rate increase of 0.8%.

Other approaches utilized the Bayesian decision rule and other machine learning techniques to improve the time complexity of an HEVC encoder. For instance, the work in [15, 16] uses the Bayes' rule to optimize PU and CU skip algorithms, respectively. In [14], the authors present a joint online and offline learning-based fast CU partitioning method that uses the Bayesian decision rule to optimize the CU partitioning process. The Bayesian decision theory is also utilized in [29] along with the correlation between the variances of the residual coefficients and the transform size to enhance the PU size decision process. Alternatively, a fast CU splitting and pruning algorithm is proposed in [4], which is applied at each CU depth according to a Bayes decision rule method based on low-complexity RD costs and full RD costs. A fast CU size and PU mode prediction algorithm that uses the k-means clustering method is introduced by [17].

On the other hand, [11] presents an early mode decision algorithm based on the Neyman-Pearson approach. In [36], a fast pyramid motion divergence (PMD)-based CU selection algorithm is proposed, where a k nearest neighbors (k-NN) like method is used to determine

the optimal CU size. The work in [39] used a machine learning-based fast coding unit (CU) depth decision method, where the quad-tree CU depth levels are modeled as a three-level of hierarchical binary decision problem. The work proposed in [6] implemented early termination techniques on CUs, PUs, and TUs using a set of decision trees grown with the aid of Waikato Environment for Knowledge Analysis (WEKA) [10], an open source data mining tool.

3 System overview

In the proposed prediction system, the first 10% of frames of a video sequence are used for training. Hence, modeling and prediction will be specific to one video as opposed to training the classification system using many video sequences. The former training approach is known as "video-dependent" modeling, while the latter is known as "video-independent" modeling. The problem with the video-independent modeling is that it follows a one-size-fits-all approach in which there is an implicit assumption that the videos used for training are suitable for predicting the CU split decisions of all other videos. Video-dependent modeling, on the other hand, makes sure that the prediction model is most suitable for predicting the CU split decisions of the remaining video content.

The concept of video-dependent modeling was previously introduced by the author in [23, 25, 27]. The first 10% of video frames were used for training and the prediction model is then used throughout the sequence in a video transcoding context. If needed, the training can be repeated periodically or in the case of detecting scene cuts. Reducing the percentage of train frames might result in a less accurate classification model as the number of feature vectors in the train set are reduced. Increasing the percentage, on the other hand, might result in a more accurate classification model. However, the time at which the model is applied to predict the split decisions of CUs will be delayed which decreases the overall gain in terms of computational complexity.

Figures. 1 and 2 present the flowcharts of the proposed training system where FV refers to Feature Vectors. Figure 1 illustrates the data collection process of the training system. The video encoder will run with normal compression operations for the first 10% of the video frames during which, for each CU, features are extracted and recorded at the highest level, which is typically 64×64. The corresponding split decision is also recorded. If the encoder decides to split the CU, then the split decisions at the 32×32 and 16×16 levels will be recursively calculated during which, the training system will record the features and corresponding split decisions at 32×32 and 16×16 CU levels. The details of the selected feature variables are discussed in the next section.

The output of this data collection process is three sets of data. Each data set contains feature vectors and the corresponding split flags for 64×64, 32×32 and 16×16 CU levels. The second step in the training system is to map the feature vectors to the split decisions. This is illustrated in Fig. 2. The result of this step is 3 training models that can be used for the prediction of CU split decisions at 64×64, 32×32 and 16×16 CU levels. Prior to model generation, there is an optional dimensionality reduction step. Again, this is applied at the three CU levels and the dimensionally reduced models are stored and used for reducing the dimensionality of the feature vectors during the testing phase, as shall be explained next. The system modeling and dimensionality reduction techniques used in this work are explained in the next section.

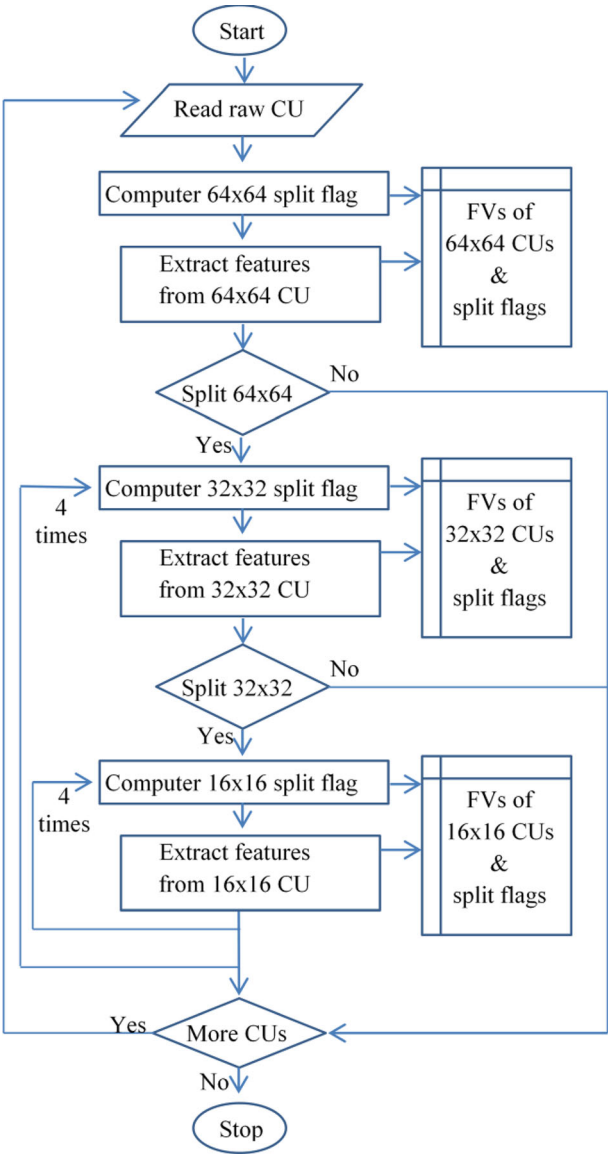

Fig. 1 Flowchart of data collection during the training phase

Once the system is trained, the generated models are used to predict the split decisions of the remaining CUs of the underlying video sequence. This process is illustrated in Fig. 3. Basically, feature variables are extracted at the highest CU level, which in this work it is 64 × 64. The corresponding train model is then used to predict the split flag. If predicted as 'no split,' then early termination is applied. Otherwise, the second train model is applied for each of the 32 × 32 CU levels and 4 split flags are predicted. If any of the flags are predicted as 'split', then the process is repeated at the 16 × 16 CU levels using the third train model. At each level, feature vectors are calculated and reduced in dimensionality if required. Again, dimensionality reduction models are calculated during the training phase.

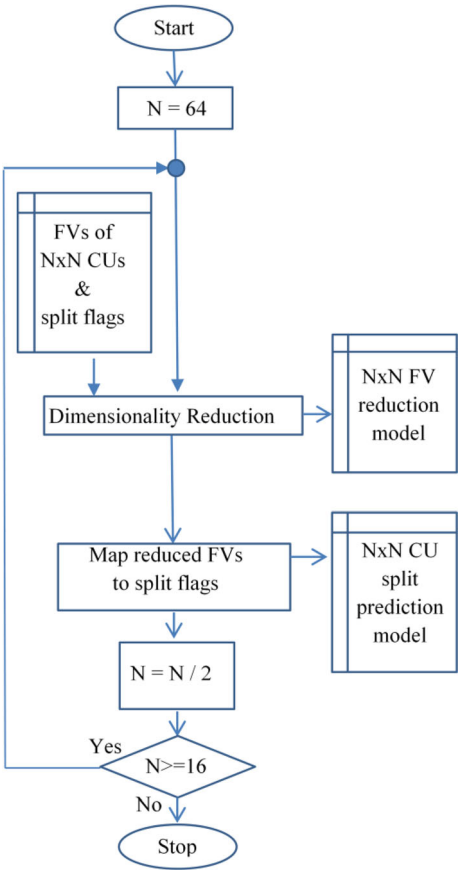

Fig. 2 Flowchart of CU split modeling in the training phase

4 System training

This section introduces the proposed feature extraction and dimensionality reduction process. It also reviews the machine learning techniques used.

4.1 Feature extraction and dimensionality reduction

In this work, feature extraction is applied at each of the three coding levels (i.e. 64 × 64, 32 × 32 and 16 × 16). Common to all levels are features extracted from surrounding CTUs. The surrounding CTUs are previously encoded and include the CTUs at the following locations relative to the current CU: left, top-left, top, top-right and co-located from the previous frame. The total number of surrounding CTUs is therefore 5. The complete list of extracted features and their description are listed in Table 1, where MVs refer to Motion Vectors. The first 15 features in Table 1 belong to the current CU, whereas the remaining 55 features belong to surrounding CTUs. The total number of features is therefore 70.

As illustrated in Fig. 2 above, the dimensionality of these features can be reduced prior to generating the training model. In this work, we generate experimental results with and without dimensionality reduction. We propose the use of the following dimensionality reduction

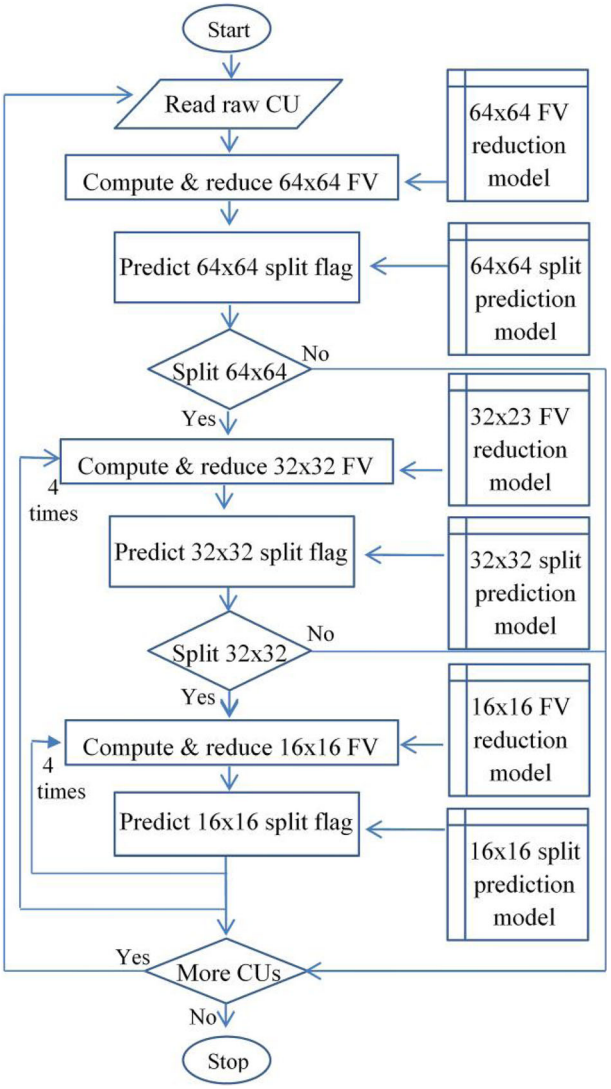

Fig. 3 Flowchart of applying the train models to predict the CU split flags

techniques: stepwise regression, principle component analysis (PCA) and reduction based on random forests. In the following, we briefly summarize the use of each relative to the proposed solution. It is important to mention that all dimensionality reduction techniques are applied to the train data set as illustrated in Fig. 2. The generated model is then applied to the test data set.

Stepwise regression is a feature selection algorithm; however, it can be used as a dimensionality reduction technique as reported in [26]. In this work, we treat the feature vectors of CUs as predictors and the split decisions as response variables. As such, the problem can be formalized in a regression context. The idea of stepwise regression is to start with one feature variable and compute its correlation with the split decision. Then, another feature variable is added and the correlation is computed again. The significance of adding another feature variable is assessed by means of examining the P value at a 0.05 level of significance. If the

Table 1 Feature variables representing CUs

Feature (length)	Description
CU depth (1)	Coding depth: 0 = 64 × 64, 1 = 32 × 32, 2 = 16 × 16
Prediction mode (1)	0 = inter, 1 = intra
PU type costs (11)	PU RD cost of (Skip, 2Nx2N, 2NxN, Nx2N, NxN, 2NxuN, 2NxdN, lNx2N, and rNx2N), and intra-PU modes (2Nx2N and NxN)
Merge flag (1)	Merge flag of current CU
Skip flag (1)	Skip flag of current CU
Total distortions of surrounding CTU (5)	Total distortion cost of each of the surrounding CTUs
Average coding depths of Surrounding CTUs (5)	Average coding depth of each CTUs of the surrounding CTUs
Variance of coding depths of surrounding CTUs (5)	Variance of coding depth of each CTUs of the surrounding CTUs
Average and variance of MVx and MVy of surrounding CTUs (40)	Average and variance of MVx and MVy of surrounding CTUs for lists List0 and List1. Normalized by frame distance.

added feature variable is found significant, then it is retained; otherwise, it is removed from the list of variables. Likewise, once a variable is retained, the stepwise regression algorithm proceeds by revisiting the previous feature variables and reassessing their significant, taking into account that a new variable has been retained. The algorithm terminates when there are no further feature variables to add or to eliminate. A full description of the algorithm can be found in [20].

Once applied to the train data set at 64 × 64, 32 × 32 and 16 × 16 CU levels, the result of the stepwise regression is simply 3 sets of indices of the retained feature variables, one set for each CU coding depth. These indices can be used to reduce the dimensionality of the feature vectors during the testing phase. Since we are using a video-dependent approach to learning in this work, the number of retained feature variables varies from one video sequence to the other. Full information about the experimental setup are given in the experimental results section; nonetheless, for completeness, we briefly discuss the results of applying the stepwise regression algorithm here. The number of retained variables for each video sequence is given in Table 2. The table lists the average number of retained CU variables with QP values of {22, 27, 32 and 37}. Since 3 models are generated, the table lists the retained variables at 64 × 64, 32 × 32 and 16 × 16 CU coding levels. A full example showing specific names of retained variables for the *RaceHorses* sequence is shown in Table 3.

In this work, we also experiment with dimensionality reduction using random forests. In this approach, we generate a large set of trees against the CU split decision. Each tree is trained on a small number of feature variables. The usage statistics of each feature variable can be used to find an informative subset of features. More specifically, if a feature variable is repeatedly selected as best split, it is consequently a good candidate to retain. More information about this algorithm can be found in [18].

Here, a random forest of 100 trees is grown, where the maximum number of decision splits or branch nodes is set to be the initial set of 70 features. The training dataset is sampled for each decision tree with replacement and the feature variables selected at random for each decision split are chosen without replacement within the same decision tree. The importance of each of these features in predicting the correct classification of a test instance from the out-of-bag data is computed and used to select the features whose raw importance score make up 80% of the total importance score. The out-of-bag data is the set of instances that were left out during the training process of a given tree in the random forest.

Table 2 Retained variables using stepwise regression

Video sequence	Avg. # retained features		
	64 × 64	*32 × 32*	*16 × 16*
RaceHorses (384 × 192)	10	11	15
BlowingBubbles (384 × 192)	4	13	12
BQSquare (384 × 192)	7	17	17
BasketballPass (384 × 192)	9	15	13
RaceHorses (832 × 448)	7	14	19
PartyScene (832 × 448)	14	27	28
BQMall (832 × 448)	14	21	22
BasketballDrill (832 × 448)	13	13	16
ParkScene (1920 × 1024)	20	30	26
Kimono1 (1920 × 1024)	17	16	14
Cactus (1920 × 1024)	20	22	28
BQTerrace (1920 × 1024)	19	24	27
BasketballDrive (1920 × 1024)	16	18	23
Traffic (2560 × 1600)	22	25	26
PeopleOnStreet (2560 × 1600)	17	28	31
NebutaFestival (2560 × 1600)	16	20	24
SteamLocomotiveTrain (2560 × 1600)	18	20	24
Average	14	19	21

The number of retained variables, using random forests, for each video sequence is given in Table 4. Similar to Table 2 above, the table lists the average number of retained CU variables with QP values of {22, 27, 32 and 37}. Since 3 models are generated, the table lists the retained variables at 64 × 64, 32 × 32 and 16 × 16 CU coding levels. A full example showing specific names of retained variables for the *RaceHorses* sequence is shown in Table 5.

Lastly, we also experiment with dimensionality reduction using PCA. In this approach, an orthogonal transformation is used to transfer the data from the feature domain to the principle component domain. The first principle component of the transformed data account for the highest variability in the feature data. One drawback of PCA is that it results in a reduced data set that cannot be directly interpreted. This is not the case for the other two dimensionality reduction techniques used in this paper. The number of

Table 3 Stepwise regression, example retained variables for 32 × 32 depth level with QP = 32

Feature (length)	Retained variables
CU depth (1)	–
Prediction mode (1)	Prediction mode
PU type costs (11)	Skip RD cost, 2Nx2N-inter RD cost, 2Nx2N-intra RD cost
Merge flag (1)	–
Skip flag (1)	Skip flag status
Total distortions of surrounding CTU (5)	Total distortion in Left CTU, Total distortion in Collocated CTU
Average coding depths of Surrounding CTUs (5)	Average coding depth in Left CTU, Average coding depth in Upper CTU, Average coding depth in Collocated CTU
Variance of coding depths of surrounding CTUs (5)	Variance of coding depth in Left CTU, Variance of coding depth in Upper CTU
Average and variance of MVx and MVy of surrounding CTUs (40)	Variance of MVx (List1) in Left CTU, Average of MVy (List0) in Upper Right CTU

Table 4 Retained variables using feature importance with random forests

Video sequence	Avg. # retained features		
	64 × 64	*32 × 32*	*16 × 16*
RaceHorses (384 × 192)	8	13	10
BlowingBubbles (384 × 192)	12	14	14
BQSquare (384 × 192)	12	13	14
BasketballPass (384 × 192)	9	14	14
RaceHorses (832 × 448)	15	15	15
PartyScene (832 × 448)	13	14	14
BQMall (832 × 448)	14	15	13
BasketballDrill (832 × 448)	15	14	14
ParkScene (1920 × 1024)	15	14	14
Kimono1 (1920 × 1024)	15	15	12
Cactus (1920 × 1024)	15	15	14
BQTerrace (1920 × 1024)	15	15	14
BasketballDrive (1920 × 1024)	15	15	15
Traffic (2560 × 1600)	14	14	14
PeopleOnStreet (2560 × 1600)	13	13	14
NebutaFestival (2560 × 1600)	15	15	14
SteamLocomotiveTrain (2560 × 1600)	13	14	13
Average	13.4	14.2	13.6

principle components retained depends on the chosen Proportion of Variance (PoV) explained, which is 90% in this work. Again, the PCA is applied to the training dataset at 64 × 64, 32 × 32 and 16 × 16 CU levels. The resulting principle components are then stored and used for reducing the test data set.

The number of retained variables, using PCA, for each video sequence is given in Table 6. Similar to Tables 2 and 4 above, the table lists the average number of retained CU variables with QP values of {22, 27, 32 and 37}. Since 3 models are generated, the table lists the retained variables at 64 × 64, 32 × 32 and 16 × 16 CU coding levels. Unlike stepwise regression and random forest variable selection, PCA results in a reduced data set that cannot be directly interpreted; hence, there are no specific retained variable names to list.

Table 5 Random forest feature importance, example retained variables for 32 × 32 depth level with QP = 32

Feature (length)	Retained variables
CU depth (1)	–
Prediction mode (1)	–
PU type costs (11)	Skip RD cost, 2Nx2N-inter RD cost, 2NxN RD cost, rNx2N RD cost, Nx2N RD cost, lNx2N RD cost, 2NxdN RD cost, 2NxuN RD cost, 2Nx2N-intra RD cost
Merge flag (1)	–
Skip flag (1)	Skip flag status
Total distortions of surrounding CTU (5)	Total distortion in Upper CTU, Total distortion in Upper Left CTU
Average coding depths of Surrounding CTUs (5)	Average coding depth in Upper CTU
Variance of coding depths of surrounding CTUs (5)	Variance of coding depth in Upper Right CTU, Variance of coding depth in Upper Left CTU
Average and variance of MVx and MVy of surrounding CTUs (40)	–

Table 6 Retained variables using PCA

Video sequence	Avg. # retained features		
	64 × 64	*32 × 32*	*16 × 16*
RaceHorses (384 × 192)	8	24	25
BlowingBubbles (384 × 192)	20	26	26
BQSquare (384 × 192)	20	22	22
BasketballPass (384 × 192)	19	20	21
RaceHorses (832 × 448)	19	30	30
PartyScene (832 × 448)	29	30	30
BQMall (832 × 448)	27	29	29
BasketballDrill (832 × 448)	24	24	25
ParkScene (1920 × 1024)	29	31	31
Kimono1 (1920 × 1024)	29	29	27
Cactus (1920 × 1024)	27	28	29
BQTerrace (1920 × 1024)	31	31	32
BasketballDrive (1920 × 1024)	29	29	29
Traffic (2560 × 1600)	22	23	24
PeopleOnStreet (2560 × 1600)	30	34	34
NebutaFestival (2560 × 1600)	29	30	31
SteamLocomotiveTrain (2560 × 1600)	28	29	30
Average	24.7	27.6	27.9

4.2 Classification methods

In this work, we model the relationship between the CU features and split decisions using two classification tools; namely polynomial networks [33], random forest [3] .

In the polynomial networks, we experiment with a second order polynomial classifier. In the case of random forests, 100 trees are grown, where the maximal number of branch nodes is the square root of the number of retained feature variables. This is determined based on the out-of-bag estimates of the features' importance in the tree ensemble. Based on the retained features, the training dataset is sampled for each decision tree with replacement. The variables selected at random for each decision split are chosen without replacement within the same decision tree. As the purpose of growing trees is classification, only one observation or class label can be seen per tree leaf. The arrangement in which we combine the classification tools with the dimensionality reduction techniques are presented in Table 7.

Table 7 Arrangement of classification solutions

Solution	Classifier	Dimensionality reduction
Stepwise & Polynomial	Polynomial networks with second order expansion	Stepwise regression
PCA & Polynomial	Polynomial networks with second order expansion	PCA with PoV of 90
R.F. Select & R.F.	Random Forest	Feature importance with random forests
R.F.	Random Forest	Not used

5 Experimental results

We assess the performance and efficiency of the proposed solutions by implementing them in the HM reference software version 13.0 [13]. All video sequences are encoded with QPs of {22, 27, 32, and 37}. The main profile and the standard random-access temporal configuration are used. The same motion estimation parameters are used in both the original and proposed video coders. For a fair comparison with [14], we use the HEVC video test sequences. The sequence resolutions are Class A (2560 × 1600), Class B (1080 pixels), Class C (800 × 480), and Class D (400 × 240). A total of 17 video sequences are used as reported in Table 8. We ran the experimental results on a PC with Intel Core i7-37400QM, 2.7-GHz CPU with a 16-GB DDR3 RAM. Compression efficiency is quantified in terms of BD-rate and BD-PSNR. The compression times using the proposed solutions are also computed and compared with the corresponding times obtained from running the unmodified HEVC encoder. Furthermore, the classification accuracy of the proposed classification systems is presented. We start by reporting the classification accuracy of the CU split prediction. Table 9 presents the overall classification accuracies of the 4 proposed solutions. These results are the average for all video sequences coded with QPs of {22, 27, 32 and 37}. The table reports the classification results for the 3 models generated according to the CU coding depth. The results indicate that the average classification accuracy enhances slightly as the depth of the CU coding increases. The results also show that the classification solutions using random forest are the most accurate. In Table 10, the average classification results for individual video sequences using the "R.F. Select & R.F." solution are shown. The resolution of the video does not seem to affect the classification accuracy. Moreover, the accuracies do not seem to vary much according to the underlying video sequence as well.

In the following experiments, we report the coding efficiency of the proposed solutions using a number of approaches; namely BD-rate, BD-PSNR [13] and Computational Complexity Reduction (CCR). CCR is computed as

Table 8 List of video sequences used

Class	ID	Video sequence	Resolution	Frame rate	Bit depth
D	D1	RaceHorses	384 × 192	30	8
	D2	BlowingBubbles	384 × 192	50	8
	D3	BQSquare	384 × 192	60	8
	D4	BasketballPass	384 × 192	50	8
C	C1	RaceHorses	832 × 448	30	8
	C2	PartyScene	832 × 448	50	8
	C3	BQMall	832 × 448	60	8
	C4	BasketballDrill	832 × 448	50	8
B	B1	ParkScene	1920 × 1024	24	8
	B2	Kimono1	1920 × 1024	24	8
	B3	Cactus	1920 × 1024	50	8
	B4	BQTerrace	1920 × 1024	60	8
	B5	BasketballDrive	1920 × 1024	50	8
A	A1	Traffic	2560 × 1600	30	8
	A2	PeopleOnStreet	2560 × 1600	30	8
	A3	NebutaFestival	2560 × 1600	60	10
	A4	SteamLocomotiveTrain	2560 × 1600	60	10

Table 9 Overall classification average of CU split prediction

Solution	64 × 64	32 × 32	16 × 16	Average
Stepwise & Polynomial	83.2	85.7	86.4	85.1
PCA & Polynomial	78.5	81.9	83.2	81.2
RF Select & RF	85.3	86.3	86.7	86.1
RF	86.2	86.5	86.8	86.5
Average	83.3	85.1	85.8	84.7

$$\mathrm{CCR} = \left(Time_{ref.} - Time_{proposedSolution}\right) / Time_{ref.} \quad (1)$$

Where $Time_{ref.}$ is the time needed for regular encoding and $Time_{proposedSolution}$ is the time needed to encode a sequence using the proposed fast encoder. In Table 11, the coding results of the 4 proposed solutions are listed. As mentioned earlier, all results are obtained with QPs of {22, 27, 32 and 37}. The best coding results in terms of BD-rate and BD-PSNR are achieved by the "R.F. Select & RF" and "R.F." solutions. The reported results also indicate that, in general, the coding efficiency enhances as the resolution of the video increases. The effect of the proposed solution on the video quality in terms of BD-PSNR ranges from −0.05 to −0.02 dB. This gives a clear indication that the effect of the proposed solution on the visual quality is minimal. Nonetheless, we show example images of regular encoding and encoding using the proposed in Fig. 4. As shown in the figure, there are no subjective artifacts as a result of the proposed solution.

We report the time complexity in terms of CCR% in Table 12, where sequences are referred to by the IDs listed in Table 8. We present the CCR% for two cases; the first case is the time savings without taking the training or the model generation time into account, while the second case is the CCR% with the training time being taken into account. In Table 12, it seems that

Table 10 Classification accuracy of CU splits for the "R.F. Select & R.F." solution

Sequence ID	R.F. Select & R.F.			
	64 × 64	32 × 32	16 × 16	Average
D1	79.0	83.8	83.0	81.9
D2	85.1	84.7	83.7	84.5
D3	82.6	87.7	87.2	85.8
D4	87.1	88.1	83.6	86.3
C1	85.3	84.7	86.0	84.9
C2	89.7	88.3	87.8	86.8
C3	91.6	87.7	85.2	86.9
C4	89.1	88.9	87.7	87.2
B1	88.4	90.6	89.7	87.8
B2	80.3	77.0	82.3	84.0
B3	89.1	86.4	88.2	87.1
B4	88.1	90.1	89.6	87.7
B5	86.6	83.8	86.6	86.2
A1	89.6	91.1	91.2	88.1
A2	91.3	87.4	83.4	87.3
A3	64.0	80.5	88.3	83.2
A4	82.7	87.0	90.7	86.5
Average	85.3	86.3	86.7	86.1

Table 11 Coding efficiency of proposed solutions

Sequence ID	Stepwise & Polynomial		PCA & Polynomial		R.F. Select & R.F.		R.F.	
	BD-rate	BD-PSNR	BD-rate	BD-PSNR	BD-rate	BD-PSNR	BD-rate	BD-PSNR
D1	2.58	−0.12	1.42	−0.07	1.58	−0.08	1.48	−0.07
D2	1.84	−0.07	0.87	−0.04	0.57	−0.02	0.53	−0.02
D3	1.23	−0.06	0.75	−0.04	0.46	−0.02	0.42	−0.02
D4	4.34	−0.21	2.45	−0.11	0.33	−0.02	0.50	−0.03
Avg.	2.50	−0.12	1.38	−0.06	0.74	−0.03	0.73	−0.03
C1	1.74	−0.07	2.09	−0.08	0.84	−0.03	0.78	−0.03
C2	1.04	−0.05	1.45	−0.07	0.69	−0.03	0.51	−0.02
C3	1.94	−0.08	1.84	−0.08	0.48	−0.02	0.38	−0.02
C4	1.59	−0.07	1.74	−0.07	0.52	−0.02	0.58	−0.02
Avg.	1.58	−0.07	1.78	−0.07	0.63	−0.03	0.56	−0.02
B1	0.83	−0.03	1.29	−0.04	0.63	−0.02	0.62	−0.02
B2	0.43	−0.02	0.76	−0.03	0.47	−0.02	0.38	−0.01
B3	0.81	−0.02	1.10	−0.02	0.56	−0.01	0.52	−0.01
B4	0.75	−0.02	0.91	−0.02	0.48	−0.01	0.69	−0.01
B5	1.60	−0.04	1.17	−0.03	0.46	−0.01	0.50	−0.01
Avg.	0.88	−0.02	1.05	−0.03	0.52	−0.01	0.54	−0.01
A1	0.66	−0.02	1.52	−0.05	0.54	−0.02	0.45	−0.02
A2	1.35	−0.06	3.72	−0.16	1.07	−0.05	0.98	−0.04
A3	0.11	0.00	0.04	0.00	0.05	0.00	0.11	0.00
A4	−0.04	0.00	−0.1	0.00	−0.2	0.00	−0.27	0.00
Avg.	0.52	−0.02	1.30	−0.05	0.36	−0.02	0.32	−0.02
Overall Avg.	1.34	−0.06	1.35	−0.05	0.56	−0.02	0.54	−0.02

without giving consideration to the training time, all solutions provide similar complexity reductions. However, because some of the training solutions are more computationally demanding, when the training time is taken into account, the range of CCR% increases from (39.1% - 37.5%) to (38.9% - 31.8%). Clearly, the most demanding training solution is the one that uses random forest. This is followed by random forest with dimensionality reduction. In this work, no attempt was taken to reduce the training time. However, one solution would be to reduce the number of train feature vectors by means of sampling, therefore reducing the model generation time. The reported results also indicate that, in general, the CCR% enhances as the resolution of the video increases. For comparison with existing work, we refer to the work reported in [14, 38]. These results contain the BD-rate and time savings only. However, the time savings are computed as

$$\Delta\text{Time} = \left(T_{ref.} - Time_{proposedSolution}\right) / Time_{proposedSolution} \tag{2}$$

This is different than the CCR reported in Table 11 above, where time savings are calculated by dividing by $Time_{reference}$ instead of $Time_{proposedSolution}$. For a fair comparison, in Table 13, we calculated the time savings of the proposed solution accordingly. In Table 13, we compare the results of our best solution against existing work.

It was not to clear to the authors if the reviewed work considered the training time when reporting the ΔTime%; hence, in Table 13, we report our results with and without taking into account the training time. In the table, Δ1Time% refers to the time savings without training time and Δ2Time% refers to the time savings taking into consideration the training time.

(a)

(b)

(c)

(d)

Fig. 4 Example images of regular encoding, (a & c) and encoding with the proposed RF select & RF solution (b & d). Image number 20 of RaceHorses and PartyScene with QP of 27

As previously mentioned, it seems that the results of the proposed solution enhance with the increasing sequence resolution. This is in contrast to the results reported in [14], where the BD-rate seems to improve with decreasing sequence resolution. Moreover, the time reduction in [14] does not seem to be affected by the resolution of the video sequences.

All in all, the results in Table 13 indicate that the proposed solution of "R.F. Select & R.F." has a clear advantage in terms of BD-rate and time savings. With reference to the average reported BD-rate of existing work (i.e. 0.765 dB), the proposed solution provides an enhancement of 27.1%. Additionally, this quality enhancement comes with further reduction in computational time. More specifically, with reference to the average reported time savings (i.e. -45.5%), the proposed solution provides an enhancement of 57% and 34% for Δ1Time% and Δ2Time%, respectively.

Table 12 Complexity reduction of proposed solutions using CCR (%)

Sequence ID	Stepwise & Polynomial		PCA & Polynomial		R.F. Select & R.F.		R.F.	
	Without train time	With train time	Without train time	With train time	Without train time	With train time	Without train time	With train time
D1	25.2	25.2	17.8	17.8	24.8	15.4	25.3	10.4
D2	33.7	33.7	31.2	31.1	32	19.7	31.8	11.4
D3	36.8	36.7	37	36.9	35.2	20.5	35.7	13.4
D4	34.8	34.7	31.6	31.5	30.6	17.8	32	11.1
Avg	32.7	32.6	29.4	29.3	30.6	18.4	31.2	11.6
C1	30.4	30.3	27.8	27.8	31.7	28.3	31	26.0
C2	32.4	32.3	31.8	31.7	33.1	28.9	33.4	27.4
C3	33.6	33.5	32.1	32.1	33	29.2	32.1	26.6
C4	39.5	39.4	38	37.9	39.2	35.6	38.4	33.1
Avg	34	33.9	32.4	32.4	34.2	30.5	33.7	28.3
B1	43.2	42.7	44.5	44.4	43.9	42.0	43.6	40.9
B2	41.5	41.5	42.7	42.6	44.1	43.0	42.7	41.2
B3	46.6	46.5	41.8	41.7	41.3	39.5	38.8	36.0
B4	47.3	47.1	47.5	47.5	48.4	46.5	49.7	46.6
B5	39.7	39.7	38.1	38.0	42.6	41.4	41.3	39.7
Avg	43.7	43.5	42.9	42.9	44.1	42.5	43.2	40.9
A1	47.3	46.8	47.3	44.8	48.7	46.3	48.7	46.1
A2	28.7	28.3	26.8	22.4	30.6	26.7	27.9	24.1
A3	50.2	50.2	47.3	44.1	51.7	50.9	52.9	52.1
A4	53.4	53.4	53.8	51.0	55.1	53.7	55.9	54.7
Avg	44.9	44.7	43.8	40.6	46.5	44.4	46.3	44.2
all Avg.	39.1	38.9	37.5	36.7	39.2	34.4	38.9	31.8

Table 13 Comparison with existing work [9, 20]

Seq. ID	R.F. Select & R.F.			Random Forest		Reviewed [20]		Reviewed [9]	
	BD-rate	Δ1Time%	Δ2Time%	BD-rate	Δ1Time%	BD-rate	ΔTime%	BD-rate	ΔTime%
D1	1.58	−33.90	−18.60	1.48	−34.60	0.90	−49.10	0.65	−26.60
D2	0.57	−49.30	−25.30	0.53	−48.40	0.33	−54.70	0.52	−36.20
D3	0.46	−59.30	−27.60	0.42	−60.70	0.47	−56.00	0.64	−43.20
D4	0.33	−45.30	−22.10	0.50	−48.20	0.65	−51.20	1.02	−28.40
C1	0.84	−47.50	−40.30	0.78	−45.80	0.97	−57.30	1.22	−25.70
C2	0.69	−51.40	−42.10	0.51	−52.70	0.25	−56.00	0.41	−34.30
C3	0.48	−50.80	−42.60	0.38	−48.80	0.77	−50.70	0.77	−40.70
C4	0.52	−66.00	−56.60	0.58	−63.50	0.52	−53.30	1.06	−35.20
B1	0.63	−83.70	−77.90	0.62	−82.80	0.49	−50.30	0.40	−41.60
B2	0.47	−81.50	−77.70	0.38	−75.80	0.64	−61.80	0.46	−35.70
B3	0.56	−74.70	−69.60	0.52	−68.40	0.57	−48.00	0.48	−42.70
B4	0.48	−111.00	−103.30	0.69	−112.80	0.87	−59.90	1.13	−44.00
B5	0.46	−75.70	−72.20	0.50	−72.50	1.33	−57.40	1.21	−46.40
A1	0.54	−101.00	−92.00	0.45	−101.60	0.71	−55.20	0.57	−43.60
A2	1.07	−44.90	−37.40	0.98	−39.60	0.79	−42.50	0.94	−42.30
A3	0.05	−109.60	−105.70	0.11	−113.80	0.78	−58.80	1.01	−47.60
A4	−0.23	−128.10	−121.30	−0.27	−131.10	0.98	−48.70	1.43	−22.10
Avg.	0.56	−71.4	−60.7	0.54	−70.7	0.71	−53.6	0.82	−37.4

Table 14 Comparison with existing work [25, 39]

Seq. ID	R.F. Select & R.F.		R.F.		Reviewed [25]		Reviewed [39]	
	BD-rate	CCR%	BD-rate	CCR%	BD-rate	CCR%	BD-rate	CCR%
D1	1.58	24.80	1.48	25.30	1.99	30.90	1.26	28
D2	0.57	32.00	0.53	31.80	1.67	25.60	1.55	42
D3	0.46	35.20	0.42	35.70	2.34	33.04	1.02	53
D4	0.33	30.60	0.50	32.00	1.17	44.54	0.98	40
C1	0.84	31.70	0.78	31.00	1.54	31.05	1.19	30
C2	0.69	33.10	0.51	33.40	2.53	39.83	1.3	44
C3	0.48	33.00	0.38	32.10	1.98	42.35	1.56	43
C4	0.52	39.20	0.58	38.40	1.72	44.19	0.95	43
B1	0.63	43.90	0.62	43.60	1.54	30.34	1.8	55
B2	0.47	44.10	0.38	42.70	2.36	36.66	0.98	36
B3	0.56	41.30	0.52	38.80	2.48	43.76	1.81	51
B4	0.48	48.40	0.69	49.70	1.62	31.91	1.7	57
B5	0.46	42.60	0.50	41.30	4.60	46.00	1.39	39
A1	0.54	48.70	0.45	48.70	3.25	49.70	1.86	59
A2	1.07	30.60	0.98	27.90	1.95	38.67	1.1	39
A3	0.05	51.70	0.11	52.90	2.18	37.90	–	–
A4	−0.23	55.10	−0.27	55.90	1.99	30.90	–	–
Avg.	0.56	39.2	0.54	38.9	2.2	37.5	1.36	43.9

Lastly, the results in Table 14 present a comparison with the recent work reported in [36], which uses CCR% for measuring the time complexity reduction. Table 14 lists the results for 15 video sequences that are used in both [36] and the proposed solution.

The results in the table show that the BD-rate of the proposed solution is on average 0.56, whereas that of the proposed solutions [32, 36] are 2.2 and 1.36 respectively. This noticeable enhancement in the BD-rate reduction comes at a slight advantage of increased computational complexity, where the CCR of the reviewed work is 37.5% [36] and that of the proposed work is 39.2%. The work in [32] has a CCR of 43.9% which comes at the expense of effecting the video quality as evident in the BD-rate.

Clearly, higher computational savings are possible, however at the expense of BD-rate. For example, the work reported in [19] proposed a solution for split prediction of CUs based on the motion features and rate-distortion cost of the NxN inter mode. The solution reuses motion vectors to expedite compression. The CCR results reported range from 55% to 61%. However, this speedup comes at the expense of high BD-rate loss of 1.93% to 2.33%. The work in [37] proposed a fast CU mode decision solution for HEVC trans-rating.

The solution uses modes and motion vectors of the correlated CUs to decide on an early SKIP decision. It also restricts the CU depth search by estimating a weighted average of the depth levels of correlated CUs. Additionally, CUs with a higher rate-distortions are divided into smaller CUs without evaluating RD costs of the remaining partitioning modes. The CCR results reported range from 55%, but again, this speedup comes at the expense of high BD-rate loss of 2.26%.

6 Conclusion

We proposed a solution for reducing the complexity of determining the CU split decisions in HEVC video coding. The solution uses a video sequence-dependent approach to collect

features that represent CUs at various coding depths. The features are extracted from both the underlying CU and the previously encoded CUs. A classification model is then built based on these feature vectors and corresponding split decisions at various CU coding depths. Additionally, dimensionality reduction is optionality used as a preprocess to model generation. Therefore, the output of the training phase is a set of classification models for predicting split decisions and a set of dimensionality reduction models. We have used a number of dimensionality reduction and classification techniques, including stepwise regression, random forest variable selection, PCA, polynomial classifiers and random forest classifiers. Experimental results were carried out on many test video sequences with different resolutions. Comparison with existing work revealed that the proposed solution has an advantage in terms of coding efficiency and time savings. It was shown that, on average, the classification accuracy of the CU split models is 86.5%. In comparison to regular HEVC coding, the proposed solutions resulted in a BD-rate of 0.55 and a BD-PSNR of −0.02 dB. The average reported computational complexity reduction was 39.2%. Future work includes the investigation of model suitability over very long sequences and the possibility of model regeneration on demand. Future work can also include investing the suitability of the proposed work for model generation using a sequence independent approach.

References

1. Ahn S, Lee B, Kim M (2015) A Novel Fast CU Encoding Scheme Based on Spatiotemporal Encoding Parameters for HEVC Inter Coding. IEEE Transactions on Circuits and Systems for Video Technology 25(3):422–435
2. Bjøntegaard G (2008) Improvements of the BD-PSNR model. document VCEG-AI11, ITU-T SG16/Q6
3. Breiman L (2001) Random Forests. Mach Learn 45(1):5–32
4. Cho S, Kim M (2013) Fast CU Splitting and Pruning for Suboptimal CU Partitioning in HEVC Intra Coding. IEEE Transactions on Circuits and Systems for Video Technology 23(9):1555–1564
5. Chung C-H, Peng W-H, Hu J-H (2017) HEVC/H.265 Coding Unit Split Decision Using Deep Reinforcement Learning. International Symposium on Intelligent Signal Processing and Communication Systems, Xiamen
6. Correa G, Assuncao PA, Agostini LV, da Silva Cruz LA (2015) Fast HEVC Encoding Decisions Using Data Mining. IEEE Transactions on Circuits and Systems for Video Technology 25(4):660–673
7. Deng X, Xu M, Jiang L, Sun X, Wang Z (2016) Subjective-Driven Complexity Control Approach for HEVC. IEEE Transactions on Circuits and Systems for Video Technology 26(1):91–106
8. Du B, Siu W-C, Yang X (2015) Fast CU Partition Strategy for HEVC Intra-Frame Coding Using Learning Approach via Random Forests. Proceedings of APSIPA Annual Summit and Conference, Hong Kong
9. Goswami K, Lee J, Kim B (2016) Fast algorithm for the High Efficiency Video Coding (HEVC) encoder using texture analysis. Inf Sci 364-365:72–90
10. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The WEKA data mining software: An update. ACM SIGKDD Explorations Newslett 11(1):10–18
11. Hu Q, Zhang X, Shi Z, Gao Z (2016) Neyman-Pearson-Based Early Mode Decision for HEVC Encoding. IEEE Transactions on Multimedia 18(3):379–391
12. Jiménez-Moreno A, Martínez-Enríquez E, Díaz-de-María F (2016) Complexity Control Based on a Fast Coding Unit Decision Method in the HEVC Video Coding Standard. IEEE Transactions on Multimedia 18(4):563–575
13. Kim I-K, McCann KD, Sugimoto K, Bross B, Han W-J, Sullivan GJ (2013) High Efficiency Video Coding (HEVC) Test Model 13 (HM13) Encoder Description. Document: JCTVC-O1002, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 15th Meeting, Geneva
14. Kim H, Park R (2016) Fast CU Partitioning Algorithm for HEVC Using an Online-Learning-Based Bayesian Decision Rule. IEEE Transactions on Circuits and Systems for Video Technology 26(1):130–138

15. Lee J, Kim S, Lim K, Lee S (2015) A Fast CU Size Decision Algorithm for HEVC. IEEE Transactions on Circuits and Systems for Video Technology 25(3):411–421
16. Lim K, Lee J, Kim S, Lee S (2015) Fast PU Skip and Split Termination Algorithm for HEVC Intra Prediction. IEEE Transactions on Circuits and Systems for Video Technology 25(8):1335–1346
17. Liu Z, Lin T, Chou C (2016) Efficient prediction of CU depth and PU mode for fast HEVC encoding using statistical analysis. J Vis Commun Image Represent 38:474–486
18. Livingston F (2005) Implementation of Breiman's random forest machine learning algorithm. ECE591Q Machine Learning Journal Paper 1–13
19. Mallikarachchi T, Talagala D, Arachchi H, Fernando A (2018) Content-adaptive feature-based CU size prediction for fast low-delay video encoding in HEVC. IEEE Transactions on Circuits and Systems for Video Technology 28(3)
20. Genuer R, Poggi J-M, Tuleau-Malot C (2010) Variable selection using Random Forests. Pattern Recogn Lett 31(14):2225–2236
21. Ohm JR, Sullivan GJ, Schwarz H, Tan TK, Wiegand T (2012) Comparison of the Coding Efficiency of Video Coding Standards—Including High Efficiency Video Coding (HEVC). IEEE Transactions on Circuits and Systems for Video Technology 22(12):1669–1684
22. Park S (2016) CU encoding depth prediction, early CU splitting termination and fast mode decision for fast HEVC intra-coding. Signal Process Image Commun 42:79–89
23. Peixoto E, Shanableh T, Izquierdo E (2014) H.264/AVC to HEVC Video Transcoder based on Dynamic Thresholding and Content Modeling. IEEE Transactions on Circuits and Systems for Video Technology 24(1)
24. Radosavljević M, Georgakarakos G, Lafond S, Vukobratović D (2015) Fast coding unit selection based on local texture characteristics for HEVC intra frame. 2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP), Orlando, pp 1377–1381
25. Shanableh T (2011) Prediction of Structural Similarity Index of Compressed Video at a Macroblock Level. IEEE Signal Processing Letters 18(5):335–338
26. Shanableh T, Assaleh K (2010) Feature modeling using polynomial classifiers and stepwise regression. Neurocomputing, Elsevier 73:10–12
27. Shanableh T, Peixoto E, Izquierdo E (2013) MPEG-2 to HEVC video transcoding with content-based modeling. IEEE Transactions on Circuits and Systems for Video Technology 23(7)
28. Shen L, Zhang Z, Liu Z (2014) Effective CU Size Decision for HEVC Intracoding. IEEE Trans Image Process 23(10):4232–4241
29. Shen L, Zhang Z, Zhang X, An P, Liu Z (2015) Fast TU size decision algorithm for HEVC encoders using Bayesian theorem detection. Signal Process Image Commun 32:121–128
30. Sullivan G, Ohm J, Han W, Wiegand T (2012) Overview of the High Efficiency Video Coding (HEVC) Standard. IEEE Transactions on Circuits and Systems for Video Technology 22(12):1649–1668
31. Sullivan GJ, Wiegand T (1998) Rate-distortion optimization for video compression. IEEE Signal Process Mag 15(6):74–90
32. Tai K-H, Hsieh M-Y, Chen M-J, Chen C-Y, Yeh C-H (2017) A fast HEVC encoding method using depth information of collocated CUs and RD cost characteristics of PU modes. IEEE Transactions on Broadcasting 63(4)
33. Toh K, Tran Q, Srinivasan D (2004) Benchmarking a reduced multivariate polynomial pattern classifier. IEEE Trans Pattern Anal Mach Intell 26(6):740–755
34. Vanne J, Viitanen M, Hamalainen TD, Hallapuro A (2012) Comparative rate-distortion-complexity analysis of HEVC and AVC video codecs. IEEE Trans Circuits Syst Video Technol 22(12):1885–1898
35. Xiong J, Li H, Meng F, Wu Q, Ngan KN (2015) Fast HEVC Inter CU Decision Based on Latent SAD Estimation. IEEE Transactions on Multimedia 17(12):2147–2159
36. Xiong J, Li H, Wu Q, Meng F (2014) A Fast HEVC Inter CU Selection Method Based on Pyramid Motion Divergence. IEEE Transactions on Multimedia 16(2):559–564
37. Yang S-H, Zhong C-C (2017) Fast Coding-Unit Mode Decision for HEVC Transrating. IEEE International Conference on Computer and Information Technology, Finland
38. Yoo H, Suh J (2014) Fast coding unit decision based on skipping of inter and intra prediction units. Electron Lett 50(10):750–752
39. Zhang Y, Kwong S, Wang X, Yuan H, Pan Z, Xu L (2015) Machine Learning-Based Coding Unit Depth Decisions for Flexible Complexity Allocation in High Efficiency Video Coding. IEEE Trans Image Process 24(7):2225–2238
40. Zhao W, Onoye T, Song T (2015) Hierarchical Structure-Based Fast Mode Decision for H.265/HEVC. IEEE Transactions on Circuits and Systems for Video Technology 25(10):1651–1664
41. Zupancic I, Blasi SG, Peixoto E, Izquierdo E (2016) Inter-Prediction Optimizations for Video Coding Using Adaptive Coding Unit Visiting Order. IEEE Transactions on Multimedia 18(9):1677–1690

Mahitab Hassan was born in Alexandria, Egypt in 1994. She received the B.Sc. and M.Sc. degrees in computer engineering from the American University of Sharjah, Sharjah, UAE, in 2015 and 2017, respectively. From 2015 to 2017, she was a Graduate Teaching Assistant with the Computer Science and Engineering Department at the American University of Sharjah. Currently she is a Developer Experience Advocate with IBM Cloud in Dubai, UAE. Her research interests include machine learning, digital video processing, and ubiquitous learning systems.

Tamer Shanableh was born in Scotland, UK in 1972. He received his Ph.D. in Electronic Systems Engineering in 2002 from the University of Essex, UK. From 1998 to 2001, he was a senior research officer at the University of Essex, during which, he collaborated with BTexact on inventing video transcoders. He joined Motorola UK Research Labs in 2001. During his affiliation with Motorola, he 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. Dr. Shanableh spent the summers of 2003, 2004, 2006, 2007 and 2008 as a visiting professor at Motorola multimedia Labs. He spent the spring semester of 2012 as a visiting academic at the Multimedia and Computer Vision and Lab at the School of Electronic Engineering and Computer Science, Queen Mary, University of London, London, U.K. His research interests include digital video processing and pattern recognition.