

## Predicting Split Decisions in MPEG-2 to HEVC Video Transcoding

Tamer Shanableh\*<sup>1</sup> and Mahitab Hassan<sup>2</sup>

<sup>1</sup>\*Department of Computer Science and Engineering, American University of Sharjah, UAE

\*[tshanableh@aus.edu](mailto:tshanableh@aus.edu)

<sup>2</sup> IBM Cloud, Dubai, UAE

**Abstract:** This paper proposes learning-based approaches for transcoding MPEG-2 video into HEVC. In the training mode of the transcoder, mappings between extracted features and split decisions are calculated. While in the transcoding mode, the split decisions of coding units of the HEVC video are predicted. Two formulations are proposed for the prediction of split decisions based on multi model and multi-tier solutions. In the former solution, multi models are generated based on the total number of split flags in a coding unit. While in the latter solution, split decisions are modelled at three different coding depths. The proposed solutions are evaluated in terms of excessive bitrate, drop in PSNR, classification accuracy, model generation time and transcoding speedup. It is shown that the multi-tier solution maintains the rate-distortion behaviour of full re-encoding at the expense of lower gain in transcoding speedup. In comparison to existing work, it is shown that the proposed solutions offer a significant enhancement in terms of rate-distortion performance and classification accuracy.

**Keywords:** Video coding, video transcoding, HEVC, machine learning

### 1. Introduction

One of the main objectives of the High Efficiency Video Coding (HEVC) is to provide a significant rate-distortion improvement in comparison to H.264/AVC. Such an improvement paves the way for new applications requiring ultra-high definition resolutions [1].

Heterogeneous video transcoding can be applied to convert existing videos compressed with popular standards such as MPEG-2 and H.264/AVC into HEVC. The term heterogeneous is used to indicate facilitating interoperability between different video coding standards. One of the earliest work on heterogeneous video transcoding was reported by the author in [2,3] for transcoding between different video formats.

To date, not much work has been reported for HEVC heterogeneous video transcoding. Nonetheless, a number of video transcoders are reported for transcoding between MPEG-2 and H.264/AVC on one side and HEVC on the other side. Noteworthy are the heterogeneous HEVC transcoding solutions that are based on content modelling for predicting the coding depth of HEVC Coding Units (CU).

For example, in [4] it was proposed to extract features from the incoming H.264/AVC coded blocks and compare their values against adaptive thresholds to decide on the outgoing HEVC CU partitioning. These incoming features are based on Motion Vector (MV) statistics, number of DCT coefficients and energy of DCT coefficients. The adaptive thresholds are computed from the first K H.264/AVC frames and corresponding outgoing HEVC CUs, where K is typically set to 25 frames.

Another approach for H.264/AVC to HEVC transcoding is based on Linear Discriminant Functions (LDFs). Features are extracted from the incoming H.264/AVC video and mapped to split or no split decisions of the outgoing HEVC CUs. Again, the weights of the LDFs are computed from the first K incoming

frames as in the previous approach. Once computed, LDFs are used to classify outgoing CUs between split or no split [5]. One exception applies to incoming blocks with a high MV variance, such blocks are automatically classified as split [4].

Parallel processing has been used to speedup the H.264/AVC to HEVC transcoding processing. The transcoder made use of incoming coding information for further speedup as well. It was reported that 720p resolution video can be transcoded in real time [6].

Other transcoding solutions exist, for instance, in [7] it was proposed to segment incoming H.264/AVC frames into three regions based on coding complexity. After that, the coding structure of an outgoing CU is determined based on the incoming region type and motion vectors. A MV clustering techniques of incoming MVs is also proposed for reducing the complexity of H.264/AVC to HEVC video transcoding [8].

On the other hand, since the MPEG-2 video content is widely available, a MPEG-2 to HEVC transcoder is proposed in [9]. The transcoding results are attractive in terms of excessive bitrate and computational complexity. However, the learning approach used is limited in that it predicts the coding depth of the first sub CU in a 64x64 coding unit. The predicted depth is replicated to the rest of the sub CUs in the same CU.

With the co-existence of many video coding standards, codec interoperability tools like video transcoders are becoming important. With the latest ITU-T-ISO/IEC HEVC codec, a need has emerged for transcoding legacy formats into HEVC. In particular, the MPEG-2 video content, which is used in TV services, digital broadcast and DVDs is plentiful. It is well-known that the MPEG-2 compression efficiency is not as good as that offered by HEVC. One approach to make use of existing MPEG-2 content whilst reducing its bitrate and file size is to transcode such videos into the efficient HEVC video codec.

The main object of this work is to propose a MPEG-2 to HEVC transcoder with an efficient learning approach to predict

all split decisions of an outgoing HEVC CU. The proposed solutions are based on generating multi split decision models and three tier classification models. For that purpose, we represent the split decisions of a 64x64 CU using 21 binary digits. Each of the split decisions of a CU are classified separately using the multi model classification approach. Whereas in the three tier model, the split decisions are grouped according to one of the three coding depths and consequently, only three classification models are generated.

Both classification approaches are integrated in a transcoding system that uses the first K input and output frames for model generation. Therefore, the main contribution of this manuscript is to predict the outgoing split decisions by modelling the relationship between the incoming MPEG-2 coding parameters and the outgoing split decisions of CUs. The proposed transcoding system shall maintain the video quality in comparison to the case of full HEVC re-encoding, yet at the same time, speedup the video conversion processes.

Such a transcoding approach is needed since existing video content coded with MPEG-2 format is abundant as it is used in high definition and standard format TV and transmission systems. By transcoding from MPEG-2 to HEVC, existing MPEG-2 content can be made use of with HEVC decoders, thus, achieving compatibility with older formats. Additionally, by transcoding into HEVC format, the MPEG-2 video files can be greatly reduced in size without sacrificing its quality as shown in this paper.

The rest of the paper is organized as follows. Section 2 present a literature review on HEVC transcoding. Section 3, introduces the overall transcoding architecture, Section 4 introduces the proposed multi model and three-tier classification solutions, Section 5 presents the experimental results and Section 6 concludes the paper.

## 2. Literature review

In the literature, most of the HEVC transcoding work is reported for H.264 into HEVC transcoding and HEVC transrating. Nonetheless, a number of interesting transcoding solutions are reported for HEVC into none ISO standardized coder are reported. For instance, in [10] a HEVC to AVS2 transcoder is proposed which decodes the incoming video in multi-stages to make use of HEVC information in the transcoding process. The authors reported speedup in the range of 11x to 17x over AVS2 reference software. Likewise, [11] reported a HEVC into VP9 transcoder with a 60% reduction in complexity at the expense of acceptable R-D penalties. On the other hand, HEVC translating is reported in [12] and [13]. The fastest solution in [12] achieved a complexity reduction of 82% with a bitrate penalty up to 3%. The transcoder in [13] is based on the decoder-encoder cascade with statistical analysis to leverage CU and PU structures from the incoming HEVC video.

It is worth mentioning that the majority of reported HEVC transcoders is related to H.264 to HEVC transcoding for both intra only and intra/inter modes. For instance, [14] proposed fast intra H.264 to HEVC transcoder by leveraging the incoming DCT coefficients and intra predictions to predict

the coding depth in HEVC CUs. The transcoder is 1.7-2.5 times faster than ordinary HEVC encoding with a bitrate penalty up to 3%. The work in [15] proposed a similar transcoder that utilizes Bayesian classifiers to speedup the transcoding process.

Additionally, inter-frame H.264 to HEVC transcoder are reported. In [16] a low complexity transcoder is proposed which makes use of machine learning and parallel processing. A parallel algorithm is proposed to makes use of a multi-core CPU and GPU. In [17] a transcoding algorithm is proposed that makes use of intra and inter coding information and MVs in H.264 to accelerate the re-encoding in HEVC. In [18], incoming H.264 blocks are fused according to a motion similarity criterion. The map is used to help in constructing the quadtree of HEVC coded frames. H.264 MVs are used as a starting point for motion estimation in HEVC as well. Time savings of 63 % are reported with bitrate penalty of 1.4 %.

In this work we complement the existing literature by proposing a MPEG-2 to HEVC transcoding solution.

## 3. Transcoding Architecture

A HEVC video frame contains equal size coding units referred to as CUs. A typical size of a CU is 64x64 pixels in the luma part. A CU can be divided into smaller blocks referred to as sub-CUs. The sizes of such sub-CUs can be 32x32, 16x16 or 8x8. Sub-CUs can be recursively divided resulting in a quadtree structure. An example of which is shown in Figure 3. The quadtree can have a maximum depth of three, which corresponds to a sub-CU size of 8x8 pixels. The final quadtree structure of a CU is computed using a brute-force method that considers all possible splitting arrangements using R-D optimization. The optimization takes into account motion estimation and compensation which makes it the most complex task of the encoder. Once the quadtree structure is computed, Prediction Units (PUs) are decided upon for each sub-CU.

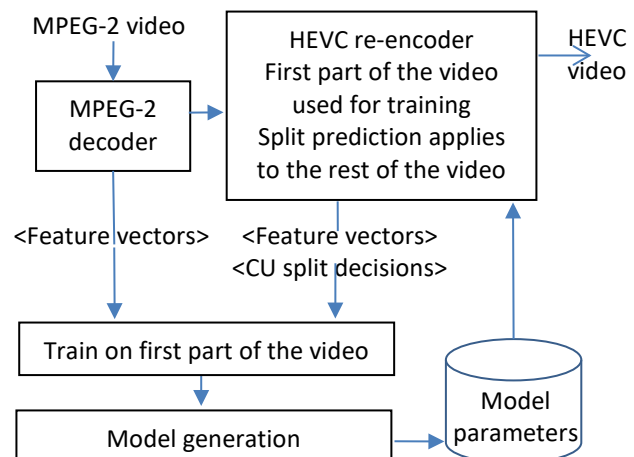


Figure 1. Overall MPEG-2 to HEVC block diagram.

A sub-CU can remain as is or be further divided into 2 or 4 PUs with symmetric shapes for intra coding and symmetric/asymmetric shapes for inter coding. For the transformation and quantization of prediction residuals, each

sub-CU is recursively partitioned into what is known as Transform Units (TUs).

Since the computation of the CU's quadtree is the most complex, the proposed transcoding system focuses on predicting the CU splits that result in the final quadtree. The overall block diagram of the proposed transcoder is shown in Figure 1. The first part of the decoded video is used for model generation. Once generated, the model parameters are used in the HEVC re-encoder to predict the quadtree structure of coding units. More specifically; to predict HEVC split decisions, the transcoding system needs to operate in two modes; a training mode and a transcoding mode as illustrated in Figure 2. In the training mode, which contains an MPEG-2 decoder followed by a full HEVC encoder, the first  $K$  frames of the incoming MPEG-2 and the corresponding outgoing HEVC videos are used for model generation. During this mode, feature vectors are extracted from the incoming bit stream and the decoded video is re-encoded using HEVC. During the re-encoding, the split decisions of each CU are stored. Having collected the feature vectors and corresponding split decisions, the training mode computes a mapping between the two and generates model weights that can be used in the transcoding mode.

On the other hand, in the transcoding mode, the feature vectors of the incoming MBs together with the previously generated model weights are used to predict the split decisions of the outgoing CUs; hence, greatly simplifying the operations of the HEVC coder.

The feature vectors extracted from the MPEG-2 video are based on motion vectors, coding information and texture variance. The exact features are listed in Table 1.

Table 1. Description of MPEG-2 feature variables

Feature variables	Num. of variables
Incoming MBs are arranged in 4x4 blocks corresponding to the size of a CU. The variance of the MVs is computed.	2
Raw MVs values ( $x,y$ ) of 4x4 MBs	32
Incoming MBs are arranged in 2x2 blocks corresponding to the size of a CU at depth 1. The variance of the MVs of each 2x2 block is computed.	8
MB types in the 4x4 block of MBs	16
MB Coded Block Patterns (CBP) in the 4x4 block of MBs	16
Number of coding bits per MB of 4x4 MBs	16
Texture variances in the 4x4 block of MBs	16

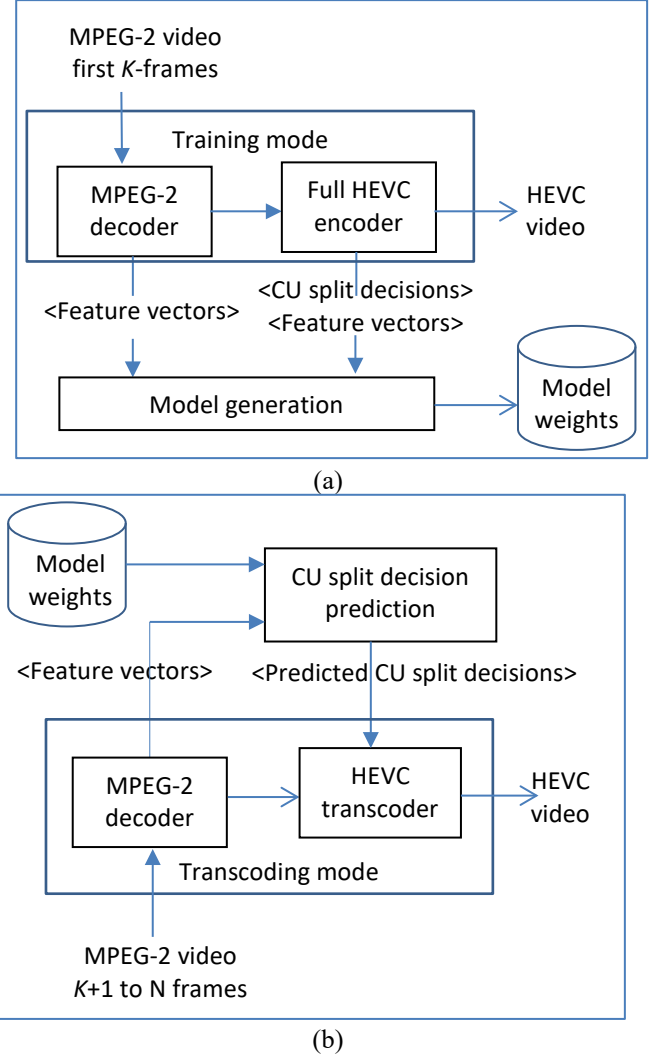


Figure 2. MPEG-2 to HEVC transcoding architecture (a) Training phase (b) Transcoding phase

In a 4x4 square of MBs, which corresponds to a 64x64 CU, the following feature variables are used: the variance of the 16 MVs in both the  $x$  and  $y$  directions, raw MV values, 16 MB types, 16 coded block patterns for each MB, texture variance of each of the 16 MBs. In addition to that, MBs are arranged into 2x2 squares pertaining to 32x32 sub-CUs and the corresponding MV variances are computed and added to the feature vector.

We propose a solution for predicting the whole coding tree unit of a given 64x64 CU, therefore, the coding depth of each sub CU is predicted by the transcoder. In the next section, we formulate the solution as a multi model classification problem.

#### 4. Proposed classification solutions

In the following sub sections, we propose two solutions for the classification of split decisions. The first solution uses feature variables from MPEG-2 for the prediction of all split decisions of the corresponding HEVC CUs. The total number of split decisions is 21 as explained in the next section.

Therefore, 21 models are generated in this classification solution. The rational behind this approach is that the feature variables pertaining to 4x4 MPEG-2 MBs correspond to one HEVC CU, hence the quadtree structure of the later can be predicted based on the extracted feature variables. We will refer to this solution as multi-model classification.

Additionally, we propose another classification solution that takes into account the three levels of splitting a CU. Considering that a typical CU has a size of 64x64 pixels, three split levels can be applied to generate sub-CUs with sizes of 32x32, 16x16 and 8x8. As such, the feature variables extracted for this task are applied to different numbers of MPEG-2 MBs pertaining to the sizes of the largest CU and sub-CUs. More specifically, three models are generated pertaining to the three split levels, where each model has its own feature matrix. We will refer to this solution as three tier classification solution.

#### 4.1. Multi model classification solution

In this proposed solution, we predict all the split decisions of a CU. To achieve this task, the split decisions of a CU in this work are represented using a sequence of 1s and 0s as illustrated in Figure 3.

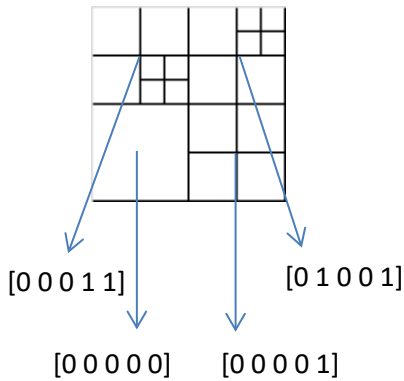


Figure 3. Representation of split decisions.

For instance, the top left 32x32 CU of the 64x64 CU in the figure is represented as [0 0 0 1 1]. The last digit indicates whether or not the 32x32 CU is split. In this example, it has been split, so the last digit is 1. The first 4 digits correspond to the four 16x16 partitions in raster scan order. Here, the first three 0s indicate that there is no further split in the first 16x16 CUs. The fourth digit in this example is; 1 hence the last 16x16 CU is split into four 8x8 sub-CUs. And so forth for the rest of the partitions. The total digits needed for a CU are 20. An extra digit is needed to indicate whether the whole CU is split or not. Thus, 21 digits are used to represent the partitioning of a CU. This splitting approach is chosen due to its simplicity, it is also consistent with the recursive function calls needed to compress a CU in the HM reference software [19].

In this work, modelling the relation between the feature vectors and split decision is performed using C4.5 decision trees and linear classification. The reason for choosing these

two classifiers is that decision trees have been used successfully in optimizing HEVC encoders as reported in [20] and linear classifiers have been used successfully in MPEG-2 to HEVC transcoding as reported in [9].

The training of the decision trees uses Kullback–Leibler Divergence (KLD) to choose the best feature variable for each decision branch. The leaves of decision trees represent the CU's split / no split decision. The trained decision trees are then used for predicting the CUs' split decisions. In the multi model solution, all 21 split decisions are predicted individually. Whereas in the three tier solution, the split decision at 64x64 level is predicted and if it is a split then the 32x32 split decision is predicted and so forth. The classification accuracy is reported by calculating the percentage of correct split decisions to the total number of instances used for testing.

The multi model linear classifier is used in the same approach, however training is performed using a closed-form formula that results in model weights. Split prediction is performed by means of applying the model weights, using dot product, to the feature variables as explained in details in Section 4.1.

In the following, we provide a formulation for the multi model classification problem using linear classification.

Denote by  $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n]^T$  is the sequence of MPEG-2 feature vectors where  $\mathbf{X} \in \mathcal{R}^{n \times m}$ , where  $m$  is the dimensionality of the feature vector, and  $n$  is the total number of feature vectors. The corresponding CU split decisions are denoted by  $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_n]^T$  where  $\mathbf{S} \in \mathcal{R}^{n \times L}$ ,  $L$  is the dimensionality of the split decisions, which is 21, and  $n$  is the total number of CUs which is the same as the total number of feature vectors.

In this formalization, the mapping between the feature vectors and the split decisions is performed using a linear classification approach [21]. However, this is not a straightforward process as 21 split decisions need to be classified. As a result, the prediction of split units is formulated as a multi model classification problem.

The training procedure is repeated  $L$  times, where  $L=21$ , to compute  $L$  optimum weight vectors  $\{\mathbf{w}_i^l\}_{l=1..L}$ , where  $\mathbf{w} \in \mathcal{R}^{m \times 21}$ . Each set of weight vectors corresponds to one split decision out of 21. Recall from Figure 3 that 21 decisions are required to identify the split structure for a given CU. Since a split decision is binary, two weight vectors are needed for each split decision,  $l$ ;  $\mathbf{w}_0^l$  and  $\mathbf{w}_1^l$ . Each vector is computed by minimizing the second norm between a linear combination of MPEG-2 train feature vectors ( $\mathbf{X}\mathbf{w}_i$ ) and a HEVC split decision at index  $l$  represented by the column vector  $\mathbf{y}_i^l$ ,  $\arg\min_{\mathbf{w}_i} \|\mathbf{X}\mathbf{w}_i - \mathbf{y}_i^l\|_2$ , such that

$$\mathbf{w}_i^l = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}_i^l \quad (1)$$

The subscript  $i$  is either 0 or 1 corresponding to the 2 classes of split and no split. The  $\mathbf{y}_0^l$  is a column vector from the  $\mathbf{S}$  matrix at split index  $l$ , and  $\mathbf{y}_1^l$  is the ones' complement of  $\mathbf{y}_0^l$ .

To classify the  $L$  split decisions of an incoming MPEG-2 feature vector represented by the row vector  $\mathbf{x}_j$ , the optimum weights obtained from (1) are used in (2)



$$s_j^l = \underset{i}{\operatorname{argmax}}(\mathbf{x}_j \mathbf{w}_i^l) \quad (2)$$

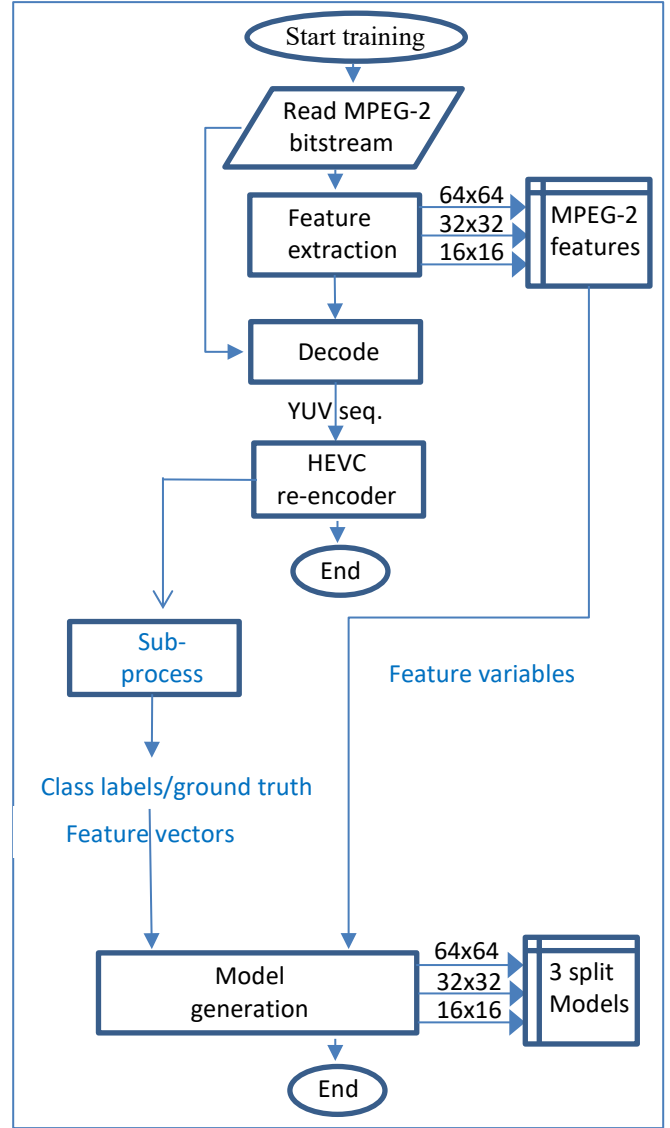
This classification process is repeated 21 times ( $l=1..21$ ) for each split decision. The advantage of using this approach for the classification of the 21 split decisions is related to the computation of its model weights. The weights in this solution are calculated using a non-iterative approach as shown in Equation (1), consequently affecting the speed of the model generation as reported in the experimental results section.

Additionally, a cleanup post-process is needed to make sure that there is no contradiction between the sub-CU and CU-level split flags. That is, if any of the sub-CU split flags is “1” then the CU level split flag will be set to “1” as well. For example, if there exists a 32x32 split flag in 64x64 CU then the split flag of the later is set to “1”. Likewise, if there is a split flag within a sub-CU, then the split flag for that sub-CU is set to “1”. For example, if there exists a 16x16 split flag in 32x32 sub-CU then the split flag of the later is set to “1”.

### 1.2. Three tier classification solution

In this proposed solution, feature variables are extracted from both the MPEG-2 decoder and HEVC re-encoder. Such feature variables result in higher rate-distortion transcoding performance as the HEVC re-encoder features are more relevant to the split decisions of the re-encoding part.

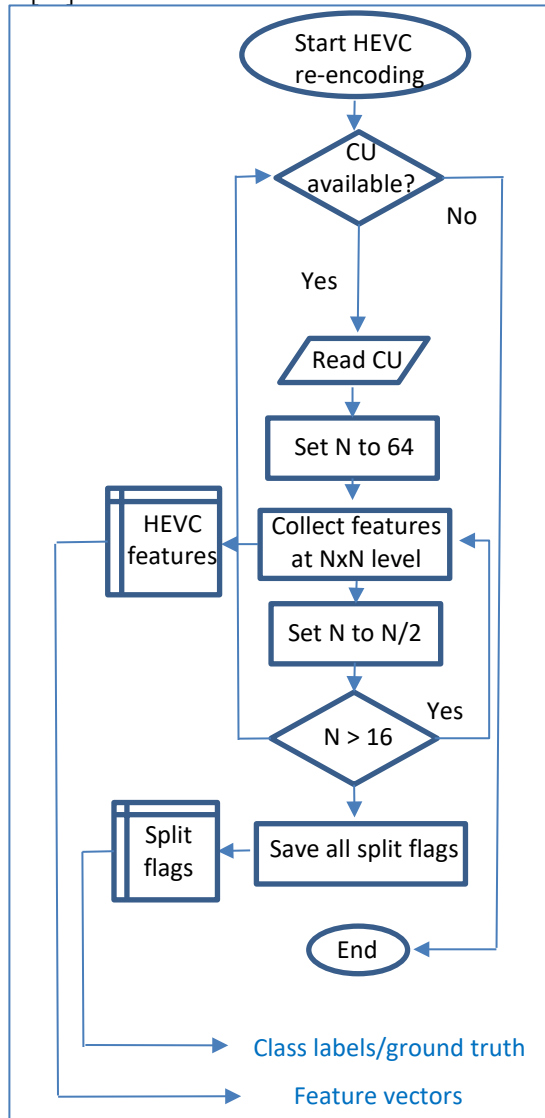
During the training phase, feature variables are collected from the MPEG-2 decoder from 4x4 MBs, 2x2 MBs and 1x1 MBs. Since the size of MPEG-2 MBs are 16x16 pixels, these MB arrangements correspond to CU sizes of 64x64, 32x32 and 16x16 respectively. Additionally, for the HEVC re-encoder part, a three tier approach is used for feature extraction as follows. For each CU, feature variables are collected at three CU depths; 64x64, 32x32 and 16x16. This results in three feature matrices combining feature variables from both MPEG-2 and HEVC. Each matrix has corresponding split decision flags/ground truth that are recorded during the training phase. Consequently, three classification models are generated at three CU depths. This model generation process is illustrated in Figure 4.



(a) Overview of model generation

Having generated the three split classification models, the system can start operating in the transcoding mode. During which, features are extracted from the MPEG-2 decoder from 4x4, 2x2 and 1x1 MBs. The HEVC re-encoder starts by extracting features at 64x64 level only. At this point, a feature vector is formed from the MPEG-2 4x4 MBs and the 64x64 CU. The 64x64 classification model is then used to classify this FV as split or no split. If classified as no split, then early CU termination takes place and no further split predictions are performed. Otherwise, if the split decision is predicted as split, then 4 feature vectors are created, one for each of the 32x32 sub-CUs. The 32x32 classification model is used to predict the split decision of each FV. This process is repeated for the 16x16 sub-CUs where the 16x16 classification model can be used to predict their split decisions. This split decision arrangement is illustrated in Figure 5. It is worth mentioning that early termination algorithms for optimizing HEVC

encoders use similar multi-tier arrangements as reported in [20] and [22].



(b) Detailed sub process  
Figure 4. Flowchart of the proposed 3-tier model generation.

Similar to the multi model solution, we implement the 3-tier solution using decision trees and linear classifiers. As mentioned previously, the 3-tier solution generates 3 models as opposed to the multi model solution where 21 models are generated. In an attempt to further reduce the model generation time, in this solution we use a reduced set of feature vectors as listed in Table 2.

Table 2. MPEG-2 and HEVC feature variables for 3-tier modeling

Feature name	Comments
--------------	----------

MPEG-2 MB skip flag	Set to 1 if all 4x4, 2x2 and 1x1 MBs are skipped. This corresponds to 64x64, 32x32 or 16x16 CUs.
Avg MVs of co-located CU	Average value of MVs of the temporally collocated CU scaled by ref. picture distance.
PU split mode	Computed prior to splitting a CU
Skip RD cost	
2Nx2N RD cost	
2NxN RD cost	
Nx2N RD cost	
NxN RD cost	
2NxnN RD cost	
2NxdN RD cost	
INx2N RD cost	
MergeFlag	Binary variables
SkipMergeFlag	
AvgDepth	Of surrounding CUs that are previously encoded

These features are collected at 3 tiers pertaining to 64x64, 32x32 and 16x16 CU sizes. The last 12 feature variables are proposed by [20] for the optimization of the HEVC encoder.

#### 4.3 Summary of proposed solutions

Conceptually, the proposed solutions are similar in terms of using the first part of the video for model generation and applying it for the prediction of split decisions for the rest of the video. However, training and feature extraction are different in both solutions.

As illustrated in Figure 2, the multi model classification solution uses the first part of the video for training the parameters of the 21 models. Once the models are generated, the transcoder extracts FVs from the incoming MPEG-2 video and uses them to predict the 21 split decisions of corresponding HEVC CUs.

As illustrated in Figures 4 and 5, the three tier solution uses the first part of the video for training the parameters of 3 split models for 64x64, 32x32 and 16x16 CUs/sub-CUs. Once the models are generated, the transcoder extracts FVs from the incoming MPEG-2 video and the HEVC re-encoder and uses them to predict the split decisions of corresponding HEVC CUs one level at a time (i.e. 64x64, 32x32 and 16x16).

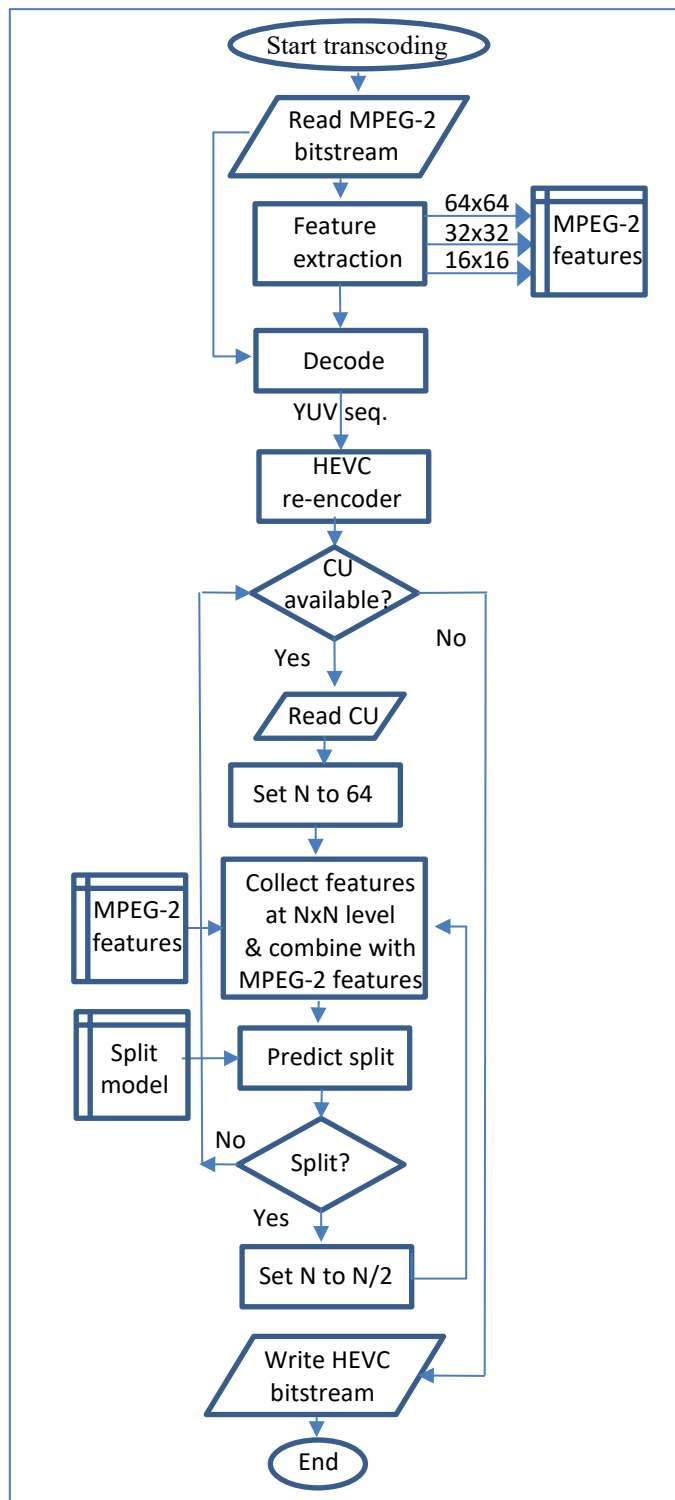


Figure 5. Flowchart of the proposed split prediction using the 3-tier solution.

## 5. Experimental results

We use a similar experimental setup to the one reported in [9]. Six test video sequences are used, namely; BasketballDrill (832x480, 50Hz), PartyScene (832x480, 50Hz), BQMall

(832x480, 60Hz), RaceHorses (832x480, 30Hz), Vidyol (1280x720, 60Hz) and BasketballDrive (1920x1080, 50Hz). The sequences are MPEG-2 encoded using an IPPP... structure.

It is typical in video transcoding to use such a GoP structure without intermediate I or B frames such that picture drift, if any, is magnified. Variable bitrate coding is used with QPs of {12,15,20,23} out of 31. The HEVC coder uses the following corresponding set of QPs {25,27,29,30}. The selected QPs are set such that the reduction in transcoded bitrate is around 50%.

The coding structure is IPPP... using 4 reference frames. The maximum CU size is set to 64x64. The asymmetric motion partitions tool and the adaptive loop filter tool are both enabled. HEVC reference software HM13.0 is used [19]. In all cases, the HEVC uses the default fast motion estimation (a modified EPZS) and fast mode decision. The first 25 frames of each sequence are used for model generation. The ground truth, which is the true split flags, are computed by encoding the video sequences using the HEVC re-encoder with the same coding parameters explained above. Additionally, for the purpose of comparing the proposed solution against recent state-of-the-art solutions, nine other video sequences are used in this section.

The proposed prediction of split decisions is assessed in the context of video transcoding. This is achieved by comparing the resultant compression results against the brute-force method of full re-encoding. The comparison is performed in terms of excessive bitrate and PSNR. The results of integrating the proposed solutions with a video transcoder are also compared against an existing MPEG-2 to HEVC transcoder of [9].

Lastly, the proposed solutions are assessed in terms of prediction accuracy and computational time.

In the following, the presentation and discussion of the results are divided into 2 subsections, namely; results based on the multi model approach and results based on the 3-tier approach.

comparison to full re-encoding, the PSNR for the proposed

Table 3. Excessive bitrate (%) of proposed and reviewed

	Multi model - Trees				Multi model - linear				Reviewed[9]			
	$Q_{in} \rightarrow Q_{out}$				$Q_{in} \rightarrow Q_{out}$				$Q_{in} \rightarrow Q_{out}$			
	12→ 25	15→ 27	20→ 29	23→ 30	12→ 25	15→ 27	20→ 29	23→ 30	12→ 25	15→ 27	20→ 29	23→ 30
Race Horses	1.98	2.79	3.35	3.62	1.63	2.14	2.41	2.55	4.2	5.6	6.1	6.5
Basket ball Drill	2.81	3.29	3.46	3.67	2.05	2.23	1.97	1.97	5.4	5.1	4.7	4.3
Party Scene	1.98	2.6	2.8	3.12	1.13	1.58	1.66	1.74	2.2	2.4	2.6	2.7
BQ-Mall	3.67	4.43	4.97	5.35	2.63	3.11	3.35	3.48	5.5	6.5	6.6	6.8
Basket ball Drive	4.28	4.79	5.01	5.41	2.47	2.61	2.4	2.46	3.8	4.1	3.8	4
Vidyo1	3.32	3.79	4.69	4.48	0.67	0.35	0.47	0.38	0.9	0.2	0.2	0
<b>Avg.</b>	<b>3.74%</b>				<b>1.98%</b>				<b>5.80%</b>			

### 5.1 Multi model approach results

The percentages of transcoding excessive bitrates based in this solution are reported in Table 3. The excessive bitrates are computed in comparison to performing full HEVC re-encoding as  $(transcoding\_bitrate - full\_re-encoding\_bitrate) / full\_re-encoding\_bitrate * 100$ . Each video sequence is transcoded 4 times using the above-mentioned input-output quantization pairs. The percentage of excessive bit rate is reported for each input-output quantization pair. The overall averages are also reported in the table.

In the table, “Multi model – Trees” and “Multi model – linear” refer to the use of decision trees and linear classifiers with the proposed multi model solution. The average excessive bitrates of the four runs show that the proposed solutions, with reference to the reviewed work, are reduced by 36% and 66% using decision trees and linear classification respectively. This is a clear transcoding advantage which is due to higher accuracy of predicting split decisions for all sub CUs of a 64x64 CU. The bitrate results also indicate that the proposed multi model classification approach based on linear classification, results in lower excessive bitrate when compared to multi model classification based on decisions trees. This indicates that there is a linear mapping between the MPEG-2 features and the corresponding split decisions in the HEVC re-encoder. Moreover, the average PSNR differences in comparison to the re-encoding approach are reported in Table 4.

As the differences are minor, the results in Table 4 are the averages of the four runs per video sequence. The drop in PSNR is computed by subtracting the PSNR of the transcoder from that of the full re-encoder. The results indicate that in

solutions are similar. Clearly, a 0.05 or 0.06 dB difference in PSNR is subjectively negligible.



Table 4. Average drop in PSNR in dB of proposed and reviewed transcoding solutions

	Multi model Trees	Multi model Linear	Reviewed [9]
RaceHorses	-0.04	-0.05	-0.07
BasketballDrill	-0.05	-0.06	-0.07
PartyScene	-0.03	-0.03	-0.04
BQMall	-0.05	-0.06	-0.09
BasketballDrive	-0.05	-0.06	-0.06
Vidyo1	-0.09	-0.11	-0.11
Average	<b>-0.05</b>	<b>-0.06</b>	<b>-0.08</b>

The prediction accuracy of split decisions plays an important role in transcoding as it affects the accuracy of re-encoding. The prediction accuracy is measured by computing the percentage of correctly predicted split decisions in comparison to full re-encoding. The accuracy is reported in Table 5 for the proposed and reviewed solutions as an average of the four transcoding runs.

Table 5. Classification accuracy of CU split decisions.

	Multi model Trees	Multi model Linear	Reviewed [9]
RaceHorses	85.3%	86.8%	59.0%
BasketballDrill	88.4%	90.0%	62.0%
PartyScene	75.1%	79.4%	58.0%
BQMall	84.3%	87.2%	60.0%
BasketballDrive	88.7%	90.4%	58.0%
Vidyo1	94.7%	95.4%	73.0%
Average	<b>86.1%</b>	<b>88.2%</b>	<b>61.7%</b>

It is shown in the table that both of the proposed solutions have higher prediction accuracy in comparison to the reviewed solution. This is a clear indication that the proposed training approach is more accurate than the existing transcoding solution. The results also indicate that the prediction accuracy using the linear classifier resulted in the highest classification accuracy. This can have a positive influence on reducing the excessive bitrate as reported in Table 4 above.

The classification accuracies of the Vidyo1 sequence stands out in all solutions. Further investigations into the Vidyo1 transcoding scenario revealed that the percentage of non-split CUs in the re-encoded video is around 53%. Therefore, it seems that the classification of split decisions in such a sequence is more straightforward.

The performance of the multi-model solution using linear classification generated best results. To further verify the claim that there is a linear mapping between the MPEG-2 features and the corresponding split decisions in the HEVC re-encoder, we repeated the results using the same linear classifier but with second order polynomial expansion of the feature vectors prior to classification. Interestingly, we found that the average classification accuracy of CU split decisions over 4 transcoding runs dropped to 82.7% which is inferior to

the linear classifier without polynomial expansion. This gives further indication that there is a linear mapping between the feature variables of Table 1 and the CU split decisions.

The proposed solutions work for higher video resolutions as well. Although the highest supported spatial resolution in MPEG-2 is 1920x1080; nonetheless, the proposed solutions are tested using the old\_town\_cross sequence which has 3840x2160 pixels at 50Hz. In both solutions, the drop in PSNR was trivial; around 0.03dB, without any excessive bitrate. The prediction accuracy of split decisions was 95.6%. In summary, based on the results presented in Tables 3, 4 and 5, it is shown that the proposed prediction of CU split decisions using the proposed training solutions are advantageous. The advantage stems from reducing the excessive bitrate to 1.98%, which is the result of a rather accurate prediction of the CU split decisions.

In Table 6, we report the model generation time of the proposed solutions. The time computations measured for the experiments in Tables 6 and 7 are carried out on an Intel® Core™ i7 CPU @ 2.7GHz. The OS is 64-bit Windows 7 Enterprise N with 16 GB of memory. As mentioned previously, 21 models are generated, one for each split decision. Each model has a copy of the feature vectors and the corresponding split decision; therefore the models can be generated in parallel as they are independent of each other. In Table 6, the parallel execution time of the models is computed by finding the maximum time required to generate each of the 21 models.

It is shown in the table that the model generation time increases as the spatial resolution of the video increases. It is also shown that the model generation time using decision trees is higher than that of using linear classifiers. This is an expected result as the latter classifiers use a non-iterative, closed-form solution for the calculation of the model weights. The model generation time of the reviewed work and the multi model solution based on linear classification require the same time as in both solutions the linear classifiers use the same number of feature vectors.

Table 6. Model generation time in seconds averaged over 4 transcoding runs.

Sequence	Multi model Trees	Multi model Linear
RaceHorses	2.31s	0.97s
BasketballDrill	2.26s	1.28s
PartyScene	2.65s	1.25s
BQMall	2.12s	1.58s
Vidyo1	6.32s	4.48s
BasketballDrive	20.6s	4.53s
Average	<b>6.04s</b>	<b>2.35s</b>

The transcoding speedup factors for the proposed and reviewed solution are reported in Table 7. The speedup factor is computed by dividing the time required for HEVC full re-encoding by the time required for transcoding with split decision predictions. The speedup factors per sequence are computed for all QP combinations and the average speedup values are reported in Table 7.

no surprise that the transcoding speedup is up to 2.5 in

Table 8. Excessive bitrate (%) of proposed 3-tier solution

	Three-Tier (Trees)				Three-Tier (Linear)			
	$Q_{in} \rightarrow Q_{out}$				$Q_{in} \rightarrow Q_{out}$			
	12→25	15→27	20→29	23→30	12→25	15→27	20→29	23→30
RaceHorses	0.67	0.16	-0.02	0.44	0.9	0.8	2.9	3.1
BasketballDrill	-0.34	0	-0.36	0	2.3	1.5	0.8	0.6
PartyScene	0.74	0.23	-0.55	-0.82	10.0	7.8	4.9	4.0
BQMall	0.67	-0.06	-0.26	-0.26	6.7	5.2	2.6	1.7
Vidyo1	-0.19	-0.29	-0.29	-0.32	8.3	-0.4	-0.5	-0.7
Basketball Drive	-1.43	-1.11	-1.37	-1.59	-1.6	-1.3	-1.6	-1.3
Avg. all runs	0.00%				2.37%			

Table 7. Transcoding speedup factors averaged over 4 runs.

Sequence	Multi model Trees	Multi model Linear	Reviewed [9]
RaceHorses	1.4	1.5	2.4
BasketballDrill	1.7	1.8	2.5
PartyScene	1.2	1.3	2
BQMall	1.5	1.5	2.3
BasketballDrive	2.3	2.3	2.9
Vidyo1	2.6	3.0	3.1
<b>Average</b>	<b>1.8</b>	<b>1.9</b>	<b>2.5</b>

Approximately, the speedup factors for the test sequences vary depending on the video content. This is so because the split decisions and the coding tree units of CUs mainly depend on the spatio-temporal content of the video sequence. The time required to compress a sequence is partially related to the complexity of the coding tree units and split decisions. In the transcoding case, coding depths beyond the predicted ones are not tested. It is shown in the table that the average speedup factor for the reviewed solution is higher than that of the proposed solutions. This is expected as the excessive bitrate of the reviewed solution as reported in Table 3 is 5.8%. It is known in video compression that high encoding speedups can be easily achieved at the expense of the rate-distortion performance.

In summary, the proposed multi model solutions are superior to the reviewed transcoding work in terms of reducing the excessive bitrate by up to 66% as shown in Table 3. PSNR differences on the other hand, are all on average negligible with a slight advantage to the proposed solutions as shown in Table 4. In terms of classification accuracy of split decisions, the proposed solutions achieved around 86% and 88% accuracy in comparison to 62% of the reviewed work as reported in Table 5. These results justify the noticeable reduction in excessive bitrate that was reported in Table 3. With the 5.8% excessive bitrate of the reviewed solution, it is

comparison to the proposed solutions that achieved up to 1.9 speedup as reported in Table 7.

Therefore, the proposed solution is superior to the existing work in terms of the prediction accuracy of the CU split decisions and in terms of reducing the excessive transcoding bitrate.

## 5.2 Three-tier approach results

In this sub section, we present and discuss the results based on the three-tier solution using decision trees. The results are compared against the multi model and reviewed work results. We start by presenting the excessive bitrate results as reported in Table 3 above. The results in Table 8 indicate that the proposed 3-tier solution using decision trees is the most accurate as on average, there is no excessive bitrate in comparison to full re-encoding. Recall that in Table 3, the average excessive bitrates for the two multi model solutions and the reviewed work are 3.74%, 1.98% and 5.8% respectively. This excellent performance of the 3-tier solution is due to the use of feature variables from both MPEG-2 decoder and HEVC re-encoder part. The feature variables of the HEVC re-encoder are more relevant to the re-encoding part as these variables directly contribute to the split decisions of the HEVC re-encoder. It is also shown in the table that the three-tier solution with linear classification results in average excessive bitrate of 2.37%. As mentioned in the previously, the feature variables used in the 3-tier solution are listed in Table 2. These feature variables are different that the ones used for the multi-model solution, which are listed in Table 1. The feature variables in Table 2 are selected based on the fact that in the 3-tier solution, features are selected from the HEVC video whilst encoding the CUs. These features are limited in number and therefore more suitable for decision trees. Further investigation revealed that the classification accuracy of the split decisions at three depth levels is inferior using linear classification in the 3-tier framework.

In some cases, the excessive bitrates reported in Table 8 are negative, which indicates that the transcoder generated slightly lower bitrate than the full re-encoder. This comes at

the expense of a slight drop in PSNR as reported in Table 9.

The drop in PSNR is computed by subtracting the PSNR of the transcoder from that of the full re-encoder. Recall that the corresponding averages of the proposed multi model solutions in Table 4 above are -0.05 and -0.06 dB. The drop in PSNR is slightly higher when the 3-tier framework with linear classifiers is used. Again, this is due to the less accurate CU split prediction at the three levels of CU depths as elaborated upon next.

Table 9. Average drop in PSNR in dB of proposed and reviewed transcoding solutions

Sequence	3-Tier Trees	3-Tier Linear	Reviewed [9]
RaceHorses	-0.03	-0.08	-0.07
BasketballDrill	-0.02	-0.04	-0.07
PartyScene	-0.04	-0.26	-0.04
BQMall	-0.02	-0.13	-0.09
BasketballDrive	-0.01	-0.03	-0.06
Vidyo1	-0.02	-0.03	-0.11
Average	-0.02	-0.096	-0.08

In terms of classification accuracy, the proposed 3-tier solution has a lower classification result in comparison to the multi model approach. The average classification accuracy of the 3-tier solution using decision trees is 81% in comparison to 86.1% and 88.2% of the multi model solutions. This can be justified as follows; in the 3-tier solution, the classifier starts by predicting the split decision for the whole 64x64 CU, if the decision is 'no split', then no further split decisions are predicted.

Therefore, if the 64x64 split decision is incorrect then all of the subsequent 32x32 and 16x16 split decisions are incorrect as well. This is not the case for the multi model solutions as the 21 split decisions are predicted separately at the same time. This prediction is followed by a clean-up process in which the 64x64 split decision can be altered if the 32x32 or the 16x16 sub CUs are predicted as being split. Additionally, the average classification accuracies using the 3-tier framework for both linear classification and decisions trees are reported in Table 10.

Table 10. Classification accuracy of CU split using 3-tiers.

Sequence	3-Tier Trees	3-Tier Linear
RaceHorses	81.6%	78.2%
BasketballDrill	82.4%	77.7%
PartyScene	74.0%	63.7%
BQMall	79.9%	74.7%
BasketballDrive	79.2%	75.1%
Vidyo1	89.2%	83.9%
Average	81.0%	75.5%

It is shown in the table that linear classification with

the proposed 3-Tier solution that uses the feature variables of Table 2 results in less accurate classification results of CU splits. This justifies the lower performance in terms on excessive transcoding bitrate and drop in PSNR as reported in Tables 8 and 9 above. Further investigation revealed that the average classification accuracy at the first tier (64x64 CU level) was 87.6% using decision trees and 74.3% using linear classifiers. The classification accuracy of the latter is relatively low. Recall that the 32x32 split decisions are examined only if the corresponding 64x64 split decision is predicted as true, therefore, inaccurate prediction of the 64x64 split decision has an adverse effect on rate-distortion performance.

The model generation time and the transcoding speedup of the proposed 3-tier solution are reported in Table 11.

Table 11. Model generation time in seconds and transcoding speedup averaged over 4 runs.

Sequence	Modeling time (s)	Trans. Speedup
RaceHorses	2.13s	1.45
BasketballDrill	1.42s	1.52
PartyScene	3.61s	1.31
BQMall	1.53s	1.44
Vidyo1	2.79s	2.53
BasketballDrive	6.97s	1.64
Average	3.08s	1.65

The average time required for model generation for the proposed multi model solutions are 6.04s and 2.35s as reported in Table 6 above. In the proposed 3-tier solution, the corresponding average is 3.08s. Again, the 3-tier solution uses decision trees; hence, the modeling time is expected to be more than the multi model solution based on linear classification. In comparison to the model generation time of the multi model solution using decision trees, the present solution has lower computational complexity as less feature variables are used in this solution. The transcoding speedup on the other hand is lower than that of the multi model solutions. This indicates that the latter solutions predicted a higher percentage of non-splits resulting in faster transcoding and lower video quality.

In summary, the 3-tier solution using decision trees resulted in the best rate-distortion behavior. In terms of model generation time, the 3-tier approach based on linear classification is the fastest, yet this is achieved at the expense of rate-distortion performance.

Additionally, the use of feature variables from both MPEG-2 and HEVC and the prediction of CU splits at three different depths resulted in eliminating the excessive transcoding bitrate as shown in Table 8. This is a clear advantage over both, the reviewed work and the proposed multi model solutions that resulted in minimum excessive bit rate of 1.9% as reported in Table 3. This discussion also holds for the average drop in PSNR as reported in Table 9. This high accuracy in transcoding comes at an expense of lower speedup of 1.65 as reported in Table 5. The speedup of the proposed multi model solution on the other hand was 1.9 as reported in Table 7.

Therefore, the second proposed solution further reduced the transcoding excessive bitrate and enhanced the video quality at the expense of reducing the transcoding

Table 12 lists the video sequences and their resolutions employed for comparison with [23], [24], [25], [26] and [27] all video sequences are encoded with QPs of {22, 27, 32, and

Table 13. Accuracy and time savings of prediction CU split decisions.

S. ID	Proposed			Ref [23]		Ref [24]		Ref [25]		Ref [26]		Ref [27]	
	BD rate%	CCR%	$\Delta T\%$	BD rate%	$\Delta T\%$	BD rate%	CCR %	BD rate%	CCR %	BD rate%	CCR %	BD rate%	CCR %
D1	1.21	31.8	52.6	0.33	54.7	1.67	25.6	0.52	36.2	1.55	42	0.9	51
D2	0.57	36.4	68.2	0.47	56	2.34	33	0.64	43.2	1.02	53	0.8	46
D3	0.72	30.1	48.4	0.65	51.2	1.17	44.5	1.02	28.4	0.98	40	0.9	41
C1	0.67	31.8	52.2	0.25	56	2.53	39.8	0.41	34.3	1.3	44	1	47
C2	0.55	31.5	50.6	0.77	50.7	1.98	42.4	0.77	40.7	1.56	43	0.9	48
C3	0.75	36.5	63	0.52	53.3	1.72	44.2	1.06	35.2	0.95	43	0.8	45
B1	0.35	43.6	92.3	0.49	50.3	1.54	30.3	0.4	41.6	1.3	44	0.8	52
B2	0.51	43.5	88.8	0.57	48	2.48	43.8	0.48	42.7	1.81	51	1.1	49
B3	0.56	45.9	114.2	0.87	59.9	1.62	31.9	1.13	44	1.7	57	1	52
A1	0.6	48	113.8	0.71	55.2	3.25	49.7	0.57	43.6	1.86	59	0.9	53
A2	0.31	48.6	110.5	0.78	58.8	2.18	37.9	1.01	47.6	-	-	-	-
A3	-0.34	52.4	136.9	0.98	48.7	1.99	30.9	1.43	22.1	-	-	-	-
<b>Avg.</b>	<b>0.54</b>	<b>40</b>	<b>82.62</b>	<b>0.62</b>	<b>53.57</b>	<b>2.04</b>	<b>37.8</b>	<b>0.79</b>	<b>38.3</b>	<b>1.4</b>	<b>47.6</b>	<b>0.91</b>	<b>48.4</b>

speedup. There is a trade-off between transcoding accuracy and transcoding speedup as expected.

Lastly, there are a number of HEVC encoder optimization solutions reported in the literature such as [23], [24], [25], [26] and [27]. The proposed CU split prediction can be applied to the HEVC encoder and compared to existing literature. For a fair comparison, we apply our 3-tier CU split prediction to the HEVC encoder and assess the accuracy and efficiency using BD-rate [22] and time savings. A lower BD-rate value and a higher time saving are desired. In the literature, some researchers use the Computational Complexity Reduction (CCR%) which is computed as  $(Time_{reference} - Time_{proposedSolution}) / Time_{reference} * 100$ . Others use  $\Delta Time$  (%) which is defined as  $(Time_{reference} - Time_{proposedSolution}) / Time_{proposedSolution} * 100$ .

Table 12. List of video sequences used.

Class	ID	Video sequence	Res.	Hz	Bit depth
D	D1	BlowingBubbles	416x240	50	8
	D2	BQSquare	416x240	60	8
	D3	BasketballPass	416x240	50	8
C	C1	ParkScene	832x480	50	8
	C2	BQMall	832x480	60	8
	C3	BasketballDrill	832x480	50	8
B	B1	PartyScene	1920x1080	24	8
	B2	Cactus	1920x1080	50	8
	B3	BQTerrace	1920x1080	60	8
A	A1	Traffic	2560x1600	30	8
	A2	NebutaFestival	2560x1600	60	10
	A3	SteamLocomotiveTrain	2560x1600	60	10

37}.

In Table 13, we show the results of predicting the CU split decisions of HEVC in comparison to [23], [24], [25], [26] and [27].

In [23], the average BD-rate penalty is 0.62 with a  $\Delta Time$  advantage of 53.6%. The proposed solution is faster as the  $\Delta Time$  is 82.6%. The time difference is achieved with an enhanced BD-rate of 0.54 instead of 0.62. In [24], the average BD-rate penalty was 2.0 with a CCR of 37.8%. In this case, the proposed solution is more accurate as the BD-rate is noticeably lower. The proposed solution is also faster as the CCR is 40% versus 37.8% for the proposed solution.

In [25], the average BD-rate penalty is 0.79 which is inferior to the proposed solution and the CCR% saving is lower as well. In [26] and [27], the CCR are 47.6% and 48.4%, however, the computational savings come at a relatively high BD-rate cost of 0.91 and 1.4 respectively.

Lastly, the work reported in [28] proposed CU split prediction based on the motion features and rate-distortion cost of the NxN inter mode. A MV reuse scheme is also proposed 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.

In conclusion, applying the proposed solution to the HEVC encoder results in high time savings whilst enhancing the compression efficiency. Incorporating the proposed solution in the MPEG-2 to HEVC transcoder, as described in Section 4, resulted in an efficient video transcoder as evident in the transcoding results of Sections 5.1 and 5.2 above.

## 6. Conclusion

Two novel solutions for transcoding MPEG-2 video into HEVC are proposed where split decisions of HEVC CUs are represented as a sequence of 21 binary numbers. Consequently, the split decisions of a CU are predicted using



21 classification models. Alternatively, the split decisions are grouped into 3 depth levels and consequently, 3 classification models are generated. The two solutions were referred to as multi models and three-tier solutions respectively. In both solutions, the transcoder starts by operating in training mode in which feature vectors are mapped to split decisions. Once the classification models are generated, the system switches to transcoding mode.

It was shown in the experimental results that the 3-tier transcoding solution generates rate-distortion results that are very close to those of the full re-encoding counterpart. In fact, it was shown that the excessive bitrate can on average be eliminated in comparison to full re-encoding. Surely, this increase in transcoding accuracy comes at the expense of speedup when compared to other transcoding solutions. It was also shown that, depending on the classifier used, the model generation time varies from 2.35s-6.04s and 0.08s-3.1s for the multi model and 3-tier solutions respectively.

**Conflict of Interest:** Authors T. Shanableh and M. Hassan declare that they have no conflict of interest.

## 7. References

- [1] ISO/IEC 23008-2:2015, "Information technology--High efficiency coding and media delivery in heterogeneous environments--Part 2: High efficiency video coding," ISO/IEC Standard, May, 2015.
- [2] T. Shanableh and M. Ghanbari, "Heterogeneous Video Transcoding Into Lower Spatio Temporal Resolutions With Different Encoding Formats," IEEE Transactions on Multimedia, 2(2), pp. 101-111, June, 2000.
- [3] T. Shanableh and M. Ghanbari, "Hybrid DCT/pixel domain architecture for heterogeneous video transcoding," Signal Processing: Image Communication, 18(8), pp. 601-620, September, 2003.
- [4] E. Peixoto, T. Shanableh and E. Izquierdo, "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), pp. 99-112, Jan., 2014.
- [5] E. Peixoto, B. Macchiavello, E. Mintsu Hung, T. Shanableh and E. Izquierdo, "An H.264/Avc to Hevc Video Transcoder Based on Mode Mapping," IEEE International Conference on Image Processing (ICIP), Melbourne, Australia, September, 2013.
- [6] Y. Chen, Z. Wen, J. Wen, M. Tang and P. Tao, "Efficient Software H.264/AVC to HEVC Transcoding on Distributed Multicore Processors," IEEE Transactions on Circuits and Systems for Video Technology, 25(8), pp. 1423-1434, August, 2015.
- [7] Wei Jiang, Yaowu Chen and Xiang Tian, "Fast transcoding from H.264 to HEVC based on region feature analysis," Multimedia Tools and Applications, 73(3), pp. 2179-2200, December, 2014.
- [8] W. Jiang and Y.W. Chenc, "Low-complexity transcoding from H.264 to HEVC based on motion vector clustering," Electronics Letters, 49(19), pp. 1224-1226, September, 2013.
- [9] T. Shanableh, E. Peixoto and E. Izquierdo, "MPEG-2 to HEVC video transcoding with content-based modeling," IEEE Transactions on Circuits and Systems for Video Technology, 23(7), pp. 1191-1196, July, 2013.
- [10] Y. Chen, Y. Zhou and J. Wen, "Efficient software HEVC to AVS2 transcoding," Information (Switzerland), 7(3), September, 2016.
- [11] E. De La Torre, R. Rodriguez-Sanchez and J. L. Martínez, "Fast video transcoding from HEVC to VP9," IEEE Transactions on Consumer Electronics, 61(3), pp. 336-343, August, 2015.
- [12] L. Pham Van, J. De Praeter, G. Van Wallendael, S. Van Leuven, J. De Cock and R. Van De Walle, "Efficient Bit Rate Transcoding for High Efficiency Video Coding," IEEE Transactions on Multimedia, 18(3), pp. 364-378, March, 2016.
- [13] J. Wang, L. Li, G. Zhi, Z. Zhang and H. Zhang, "Efficient algorithms for HEVC bitrate transcoding," Multimedia Tools and Applications, 76(24), pp. 26581-26601, December, 2017.
- [14] C.-S. Lin, W.-J. Yang and C.-W. Su, "FITD: Fast Intra Transcoding from H.264/AVC to high efficiency video coding based on DCT coefficients and prediction modes," Journal of Visual Communication and Image Representation, v38, pp. 130-140, July, 2016.
- [15] A. J. Diaz-Honrubia, J. L. Martinez and P. Cuenca, "A fast intra H.264/AVC to HEVC transcoding system," Multimedia Tools and Applications, 77(5), pp. 6367-6384, March, 2018.
- [16] A. Diaz-Honrubia, G. Cebrián-Márquez, J. Martínez, P. Cuenca, J. Puerta and J. Gámez, "Low-complexity heterogeneous architecture for H.264/HEVC video transcoding," Journal of Real-Time Image Processing, 12(2), pp. 311-327, August, 2016.
- [17] M.F. Liu, G.-Y. Zhong, X.-H. He and L.-B. Qing, "Transcoding method from H.264/AVC to high efficiency video coding based on similarity of intraprediction, interprediction, and motion vector," Journal of Electronic Imaging, 25(5), September, 2016.
- [18] E. Mora, M. Cagnazzo and F. Dufaux, "AVC to HEVC transcoder based on quadtree limitation," Multimedia Tools and Applications, 76(6), pp. 8991-9015, March, 2017.
- [19] I.-K. Kim, K. D. McCann, K. Sugimoto, B. Bross, W.-J. Han and G. J. Sullivan, "High Efficiency Video Coding (HEVC) Test Model 13 (HM13) Encoder Description," JCT-VC O1002, 15th meeting of Joint Collaborative Team on Video Coding of ITU-T SG 16 WP 3 and ISO/IEC JTC 1, November, 2013.
- [20] G. Correa, P. A. Assuncao, L. V. Agostini and L. A. da Silva Cruz, "Fast HEVC Encoding Decisions Using Data Mining," IEEE Transactions on Circuits and Systems for Video Technology, 25(4), pp. 660-673, April, 2015.
- [21] T. Shanableh and K. Assaleh, "Feature modeling using polynomial classifiers and stepwise regression," Neurocomputing, 73(10-12), pp. 1752-1759, June, 2010.
- [22] G. Bjøntegaard, "Improvements of the BD-PSNR model, document VCEG-A111, ITU-T SG16/Q6," Berlin, Germany, July, 2008.
- [23] H. Kim and R. Park, "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), pp. 130-138, January, 2016.
- [24] J. Xiong, H. Li, Q. Wu and F. Meng, "A Fast HEVC Inter CU Selection Method Based on Pyramid Motion



Divergence,” in IEEE Transactions on Multimedia, vol. 16, no. 2, pp. 559-564, February, 2014.

[25] H. Yoo and J. Suh, “Fast coding unit decision based on skipping of inter and intra prediction units,” Electronics Letters, 50(10), pp. 750-752, August, 2014.

[26] K.-H. Tai, M.-Y. Hsieh, M.-J. Chen, C.-Y. Chen, and C.-H. Yeh, “A Fast HEVC Encoding Method Using Depth Information of Collocated CUs and RD Cost Characteristics of PU Modes,” IEEE transactions on broadcasting, 63(4), December, 2017.

[27] M. Tang, X. Chen, J. Gu, Y. Han, J. Wen, and S. Yang, “Accelerating HEVC Encoding Using Early-Split,” IEEE signal processing letters, 25(2), February, 2018.

[28] T. Mallikarachchi, D. Talagala, H. Arachchi and A. Fernando, “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), March, 2018.