J. Vis. Commun. Image R. 38 (2016) 824–837

Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

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

Machine learning based fast H.264/AVC to HEVC transcoding exploiting block partition similarity ☆

Linwei Zhu [a,b], Yun Zhang [a,*], Na Li [a], Gangyi Jiang [c], Sam Kwong [b,d]

[a] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China
[b] Department of Computer Science, City University of Hong Kong, Hong Kong, China
[c] Faculty of Information Science and Engineering, Ningbo University, Ningbo 315211, China
[d] Shenzhen Research Institute, City University of Hong Kong, Shenzhen, China

ARTICLE INFO

Article history:
Received 1 September 2015
Revised 9 March 2016
Accepted 24 April 2016
Available online 25 April 2016

ABSTRACT

Video transcoding is to convert one compressed video stream to another. In this paper, a fast H.264/AVC to High Efficiency Video Coding (HEVC) transcoding method based on machine learning is proposed by considering the similarity between compressed streams, especially the block partition correlations, to reduce the computational complexity. This becomes possible by constructing three-level binary classifiers to predict quad-tree Coding Unit (CU) partition in HEVC. Then, we propose a feature selection algorithm to get representative features to improve predication accuracy of the classification. In addition, we propose an adaptive probability threshold determination scheme to achieve a good trade-off between low coding complexity and high compression efficiency during the CU depth prediction in HEVC. Extensive experimental results demonstrate the proposed transcoder achieves complexity reduction of 50.2% and 49.2% on average under lowdelay P main and random access configurations while the rate-distortion degradation is negligible. The proposed scheme is proved more effective as comparing with the state-of-the-art benchmarks.

1. Introduction

With the development of computing and multimedia technologies, various types of multimedia devices become available, including computers, laptop, smart phones, television, pad, set-top box, even the wearable devices, like glasses and watch. The diversities of capabilities of these multimedia devices, conditions of their accessed networks, multimedia data syntax/formats and user requirements unfortunately create a gap in end-to-end communication and sharing the multimedia contents among different terminals. Video transcoding is one of the proper solutions to bridge the gap for video communication among different applications and systems [1]. It is a process of converting one compressed stream to another required stream, in which properties of the bit stream may be changed, including coding syntax, bit rate, resolution, frame rate, coding structure and quality [2].

Coding technologies and standards are necessary to transcode one bit stream to another. With the advance of the video coding standards, the compression efficiency is significantly improved as adopting many novel and advanced coding tools and technologies. For example, the compression efficiency is doubled from MPEG-2 to H.264/Advanced Video Coding (AVC) [3] and H.264/AVC to High Efficiency Video Coding (HEVC) [4]. These video compression standards actually co-exist in a certain range of applications, which makes transcoding desirable. Meanwhile, the syntax and coding technologies vary significantly from standard to standard, which makes it challenging to transcode a bit stream of previous standard to that of the latest, *e.g.* HEVC.

Over the last few decades, many researches have been developing video transcoding algorithms for different system requirements and applications. Cock et al. [2] used motion refined rewriting of single-layer H.264/AVC streams to multiple quality layers for Scalable Video Coding (SVC) streams. Zhang et al. [5] proposed a novel multidimensional no-reference video quality metric for video transcoding, where frame rate and frame size are taken into account simultaneously. Liu et al. [6] proposed a Quality of Experience (QoE) oriented transcoding approach to enhance the quality of mobile 3D video service, where transcoding parameters are configured according to the feedbacks of both the network and

☆ This paper has been recommended for acceptance by M.T. Sun.

* Corresponding author.
E-mail addresses: lwzhu2-c@my.cityu.edu.hk (L. Zhu), yun.zhang@siat.ac.cn (Y. Zhang), na.li1@siat.ac.cn (N. Li), jianggangyi@nbu.edu.cn (G. Jiang), cssamk@cityu.edu.hk (S. Kwong).

http://dx.doi.org/10.1016/j.jvcir.2016.04.020
1047-3203/

the user-end device information. A straightforward video transcoder is designed merely to cascade the decoder and encoder, noted as cascaded transcoder [1]. The incoming video stream is fully decoded, and then the target video bit-stream is generated with a new encoder by re-coding the former reconstructed video. In this transcoder, the bit rate, syntax and format can be fully changed. However, the information of the source compressed stream that has not been fully and effectively explored to facilitate thereafter encoding process, such as Motion Vector (MV) and block partition type. Though video coding technical details vary from standard to standard, there is still high correlation between the encoding outputs of two standards, *e.g.* block partition, reference frame indices, since they basically follow the same framework, block based hybrid structure.

To lower the coding complexity of the transcoder, Liu et al. [7] proposed a fast MPEG-2 to H.264/AVC transcoding method based on mode mapping and macroblock (MB) activity, in which skip and intra modes are directly mapped from MPEG-2 to H.264/AVC, and the MB mode of H.264/AVC is estimated according to the residual Discrete Cosine Transform (DCT) energy from MPEG-2 stream. Shu et al. [8] proposed a mode decision method by taking into account the error propagation to the following frame. It enhanced the overall robustness of the transcoded bit stream against the packet loss. Moreover, a fast Motion Estimation (ME) algorithm for MPEG-2 to H.264/AVC transcoding was proposed in [9], where MVs as well as MB mode information extracted from MPEG-2 bit-stream were utilized to speed up the ME of H.264/AVC encoder. An MV decomposition algorithm for H.263 to H.264/AVC transcoding was presented in [10]. Petljanski et al. [11] presented a MB mode estimation technique for MPEG-2 to H.264/AVC intra transcoding, in which the DCT coefficients of MPEG-2 stream were exploited to represent the video texture and predict intra MB mode of H.264/AVC. In summary, the above methods mainly focus on the transcoding among MPEG-2, H.263 and H.264/AVC.

In HEVC, quad-tree block partition is adopted and the Coding Unit (CU) size is from 8×8 to 64×64. In addition, more advanced and refined coding tools, such as flexible Prediction Unit (PU) and Transform Unit (TU) mode, are adopted to further reduce the prediction residue [4]. These techniques improve the coding efficiency, however, significantly increase the coding complexity. Meanwhile, the hike of the number of mode candidates in HEVC makes the optimization more challenging. Recent researches on video transcoding from previous hybrid compression standards to HEVC have also been reported. Peixoto et al. [12] proposed a fast H.264/AVC to HEVC transcoder, where the MVs of H.264/AVC stream were reused in HEVC coding process. In [13], power spectrum based Rate Distortion Optimization (RDO) model, input residue, modes and MVs were used to estimate the best CU, PU modes and their corresponding MVs. Shen et al. [14] proposed a parallelization optimization method for fast H.264/AVC to HEVC transcoder, which implemented fast mode decision with wavefront parallel processing and Single Instruction Multiple Data (SIMD) acceleration. Jiang and Chen [15] proposed a MV clustering based fast H.264/AVC to HEVC transcoding method. Chen et al. [16] developed a H.264/AVC to HEVC transcoder based on distributed multi-core processors, which claimed 720p@30fps real-time video transcoding. Generally, these above mentioned schemes use the information extracted from H.264/AVC bit-stream to form a mode mapping between H.264/AVC and HEVC with hard thresholds and categories, which will limit the transcoder's performance and adaptability. While, it jointly takes advantage of the current coding information in HEVC and video content to improve the transcoding performance.

To develop more advanced and reliable video coding or transcoding algorithms, learning algorithms have been introduced to solve the prediction or classification problems. Fernandez-Escribano et al. [17] presented an efficient MPEG-2 to H.264/AVC baseline profile transcoder, where machine learning tools were used to exploit the correlation between the MB mode in H.264/AVC and the distribution of motion compensation residue in MPEG-2. Furthermore, a dynamic motion estimation technique was proposed to further reduce the complexity of the decision process. Chiang et al. [18] proposed a statistical learning based fast coding algorithm for H.264/AVC, in which several representative features in H.264/AVC were analyzed and a statistical learning model was built. With the help of an off-line classification approach from statistical learning, the complexity of motion estimation and mode decision was significantly reduced. In [19], an on-line machine learning based solution was introduced for MPEG-2 to HEVC transcoding. After fully decoding and encoding a few frames, this transcoder analyzed the relationship between MPEG-2 stream information and CU depths in HEVC, so as to build the learning model. Then, with this learning model, the HEVC's CU depth of the rest frames can be predicted by classifiers. In [20], statistical thresholds are integrated in the machine learning based coding framework to help improve the accuracy. However, the features adopted are empirically determined. Thus, it is difficult to have a good trade-off between the complexity and RD performance for different sequences. Additionally, in [21], CU partition in HEVC was regarded as a binary classification problem solved by Support Vector Machine (SVM). In the SVM model training, the RD degradation caused by misclassification was introduced as weights, which reduced the number of misclassifications with higher negative impacts. Xiong et al. [22] proposed a fast CU decision scheme based on Markov Random Field (MRF) for fast HEVC inter coding, where the variance of the absolute difference is adopted as the key feature. In [23], Zhang et al. proposed an early termination structure and three-output joint classifier based on SVM, which can flexibly adjust the CU depth prediction accuracy and complexity reduction by changing the weighted factor in training. However, this model is trained off-line and the feature selection still could be further improved. Basically, machine learning based approaches are capable of integrating multiple features and learning from video content or previous coded information to improve the transcoding performance.

In this paper, a machine learning based fast H.264/AVC to HEVC transcoding method is proposed, where a fast CU decision algorithm is presented. Compared with our previous work in [23], this paper will address the problem in fast H.264/AVC to HEVC transcoding, and optimize the feature selection for the fast CU decision. Meanwhile, we propose an on-line probability based SVM method for CU depth prediction, in which an adaptive probability threshold decision algorithm is adopted to achieve a better trade-off between low complexity and high compression efficiency. The remainder of this paper is organized as follows: problems and motivations are presented in Section 2. Then, complexity redundancies are analyzed in Section 3. Section 4 describes the proposed machine learning based fast transcoding method. Experiments are implemented and the comparative results are analyzed in Section 5. Finally, conclusions are drawn in Section 6.

2. Problems and motivations

In conventional cascaded transcoder, the bit stream information is not effectively utilized to facilitate the encoding process. To tackle this problem and fully exploit the available information, the bit stream is not necessary fully decoded and some of the bit stream information can be shared to cascaded coding [17,19], such as MVs, reference frames and block partitions. However, there are

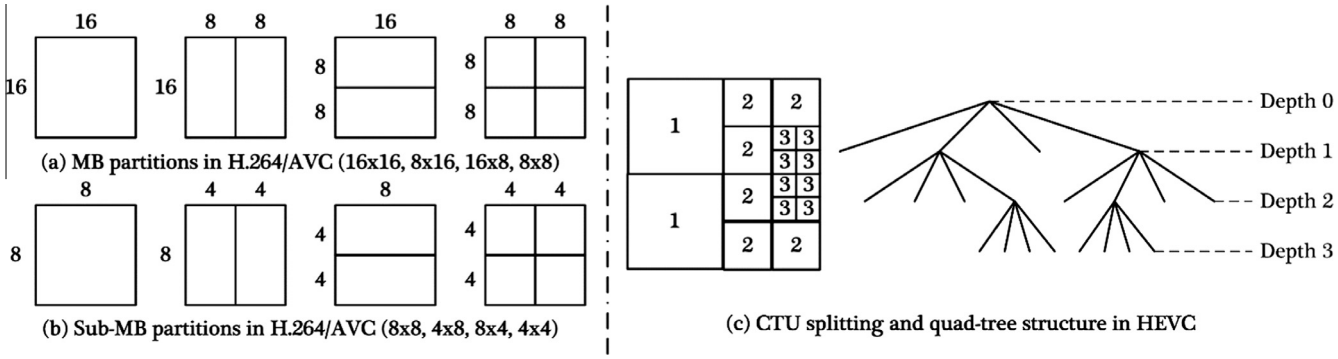
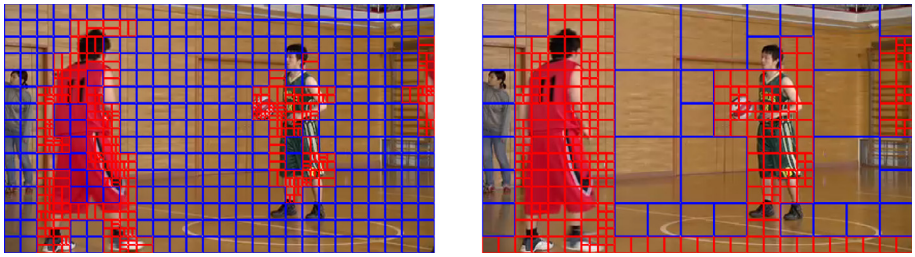


Fig. 1. Block partitions in H.264/AVC and CTU splitting, quad-tree structure in HEVC.



(a) H.264/AVC MB partition (BasketballPass) (b) HEVC CU partition (BasketballPass)

Fig. 2. Example of block partition similarity between H.264/AVC and HEVC.

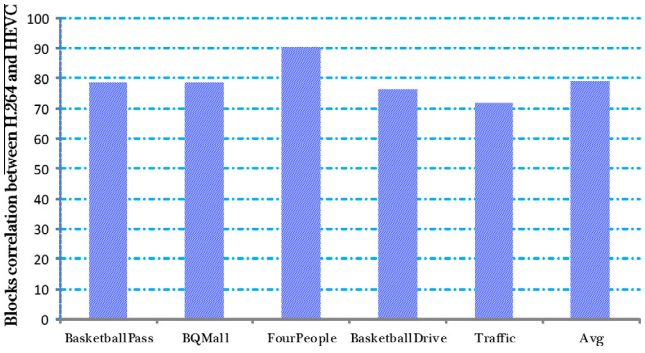


Fig. 3. Statistical block partition similarity between H.264/AVC and HEVC.

still several challenging issues in H.264/AVC to HEVC transcoding. One of them is the HEVC adopts quad-tree structured partition and the basic unit is up to 64×64, which is different from the 16×16 MB in H.264/AVC. The second is the HEVC has adopted more advanced and refined coding techniques than H.264/AVC in order to improve the coding efficiency, which also significantly increases the computational complexity. So, it is challenging to predict fine-grained coding parameters from coarse-grained coding parameters.

To analyze the block similarity between H.264/AVC and HEVC, we firstly have a brief review on the block partitions in H.264/AVC and HEVC. In H.264/AVC, a MB can be partitioned into one 16×16, two 16×8, two 8×16, or four 8×8 blocks, as shown in Fig. 1(a). Each sub-MB 8×8 block can be further partitioned into two 8×4, two 4×8, or four 4×4 blocks for prediction of its luma component, as shown in Fig. 1(b). In HEVC, the concept of Coding Tree Unit (CTU) is similar to the MB in H.264/AVC. However, the largest CU (LCU) size of the CTU is up to 64×64. Four sub-CUs of the same size would be split recursively, until to the minimum CU size 8×8. Then different encoding parameters could be used in each CU level. Fig. 1(c) shows an example of LCU partitions and its corresponding quad-tree structure. As shown in Fig. 1(c), the CU sizes of 64×64 to 8×8 are noted as Depth 0 to Depth 3, and the number 1, 2, and 3 mean Depth 1, Depth 2 and Depth 3, respectively. Basically, the concept of CU partition in HEVC is similar to that of H.264/AVC, but it technically has more levels and larger blocks.

As H.264/AVC and HEVC are both block based hybrid video coding standards, they have the similar trend of using smaller size of blocks in the texture/motion area, and using larger size blocks in the smooth/static area. There would be high correlation between H.264/AVC and HEVC block partitions. To verify the correlation, statistical experiments are performed by a cascaded transcoder made from the original H.264/AVC decoder and the original HEVC encoder. The H.264/AVC bit streams are generated under QP 28 with structure IPPP, where the HEVC encoder is configured as lowdelay P main (LP) setting under QP 28. Five sequences with diverse properties and resolutions are adopted in this statistical analyses, including BasketballPass (416×240), BQMall (832×480), FourPeople (1280×720), BasketballDrive (1920×1080), and Traffic (2560×1600).

One example of the optimal block partitions for BasketballPass sequence from H.264/AVC and HEVC encoders are shown in Fig. 2. In Fig. 2 (a), blocks of 16×16 (defined as larger block) are marked as blue boundary, while smaller blocks are marked as red boundary. In Fig. 2(b), the CUs with the size of 64×64 and 32×32 (defined as larger blocks) are marked as blue boundary, while CUs with the sizes of 16×16 and 8×8 (defined as smaller blocks) are marked as red boundary. Comparing Fig. 2(a) with Fig. 2(b), we can find that when large size blocks are used by H.264/AVC, the large size blocks will have high probability to be utilized at the same area by HEVC, and vice versa. To analyze this correspondence more accurately, the statistical correlation of the block similarity between H.264/AVC and HEVC for five different sequences is

Table 1
Complexity analyses for each depth and theoretical time saving (inter frame).

Sequences	QP	Complexity of each depth			Depth distributions (%)				Time saving (%)		
		D_0/D_3	D_1/D_3	D_2/D_3	A_0	A_1	A_2	A_3	TS_0	TS_1	TS_2
Basketballpass (416 × 240)	24	30.3	12.2	3.9	18.7	40.5	35.1	5.6	29.2	62.9	75.4
	28	31.4	12.3	3.9	21.2	40.4	34.4	4.1	30.8	64.0	75.7
	32	30.7	11.8	3.7	24.4	41.2	31.7	2.6	33.0	65.6	76.2
	36	30.1	12.0	3.6	27.8	42.7	27.9	1.6	35.2	67.6	76.7
BQMall (832 × 480)	24	36.2	13.6	4.0	20.9	39.2	33.9	5.9	30.6	63.4	75.6
	28	36.5	13.6	3.9	27.7	39.7	28.5	4.1	35.2	66.5	76.5
	32	37.5	13.8	3.9	33.6	39.6	24.3	2.6	39.1	69.0	77.2
	36	38.3	13.6	3.8	38.5	40.1	19.8	1.6	42.3	71.2	77.8
Johnny (1280 × 720)	24	36.4	11.5	3.5	57.2	27.8	14.5	0.5	54.7	74.8	79.4
	28	35.4	10.8	3.4	68.7	21.9	9.2	0.2	62.4	77.6	80.5
	32	35.0	10.3	3.2	76.2	16.7	6.9	0.1	67.3	78.9	81.2
	36	35.4	10.1	3.2	80.9	13.2	5.9	0.08	70.4	79.7	81.5
Kimono1 (1920 × 1080)	24	30.9	12.3	3.7	18.2	56.2	24.4	1.2	28.9	68.3	76.2
	28	34.3	12.7	3.7	26.8	52.7	19.7	0.9	34.6	70.7	77.0
	32	38.3	13.1	3.7	36.4	47.6	15.3	0.6	40.9	73.0	78.0
	36	41.3	13.2	3.7	47.2	41.7	10.6	0.5	48.1	75.5	78.9
Average		34.9	12.3	3.7	39.0	37.6	21.4	2.0	42.67	70.54	77.74

shown in Fig. 3. We find that the average correlation between these two standards ranges from 72% to 90% and reaches around 79% on average. This statistical result demonstrates high correlation in block partitions between these two video coding standards, which we call "Block Partition Similarity" and it motivates our transcoding optimization.

3. Analyses on complexity redundancies in HEVC transcoder

According to the three-level flexible quad-tree structure based block partition in HEVC, three-level of classifiers are assigned for the CU splitting prediction [23], as shown in Fig. 4. Symbols, such as circle, rectangle and triangle, indicate different kinds of classifiers and the arrows around the symbol indicate two choices or class candidates. Three different classifiers are used to determine splitting (+1) or not (−1) for CUs size from 64 × 64, 32 × 32, 16 × 16 to 8 × 8. There is one binary classifier in level one determining whether 64 × 64 or 32 × 32, four binary classifiers in level two determining whether 32 × 32 or 16 × 16, and 16 binary classifiers in level three determining whether 16 × 16 or 8 × 8. According to this CU prediction structure, the potential complexity reduction is statistically analyzed under the assumption that each classifier has 100% CU prediction accuracy.

We statistically analyze the time consumption for each CU depth by transcoding each video stream four times with four different individual depths, so as to calculate upper bound of complexity reduction. The transcoder is cascaded by the original H.264/AVC decoder and original HEVC encoder. D_0 to D_3 indicate the average time of encoding one unit for depth level 0–3, as shown in the left three columns in Table 1. The time of encoding four individual depths contains all the time consumption in the coding process. By normalizing the complexity of each CU, we get the average complexity from D_0 to D_3 as D_0:D_1:D_2:D_3 = 34.9:12.3:3.7:1, which means the encoding complexity of one CU with size from 64 × 64 to 8 × 8 is 34.9:12.3:3.7:1. In addition to the average CU coding complexity, the numbers of CUs in the four depths is also analyzed. The percentages of the CU depth from Depth 0 to Depth 3 are defined as A_0, A_1, A_2, and A_3. The middle four columns in Table 1 show the statistical results for A_0 to A_3. In terms of the CU depth of the video frames, we observe that about 39% and 37.6% image area select 64 × 64 and 32 × 32 CUs respectively. On the other hand, only 2.0% image area select the 8 × 8 as their best CU size.

In Fig. 4, if only Classifier 0 is used in the CU depth prediction in the transcoding, the time saving is defined as TS_0; if Classifier 0 and Classifier 1 are both used in the CU depth prediction, the time

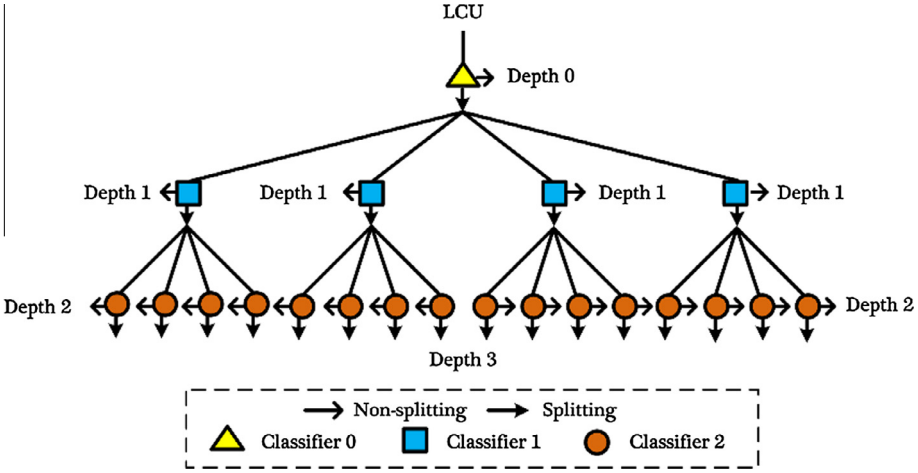


Fig. 4. CU splitting or non-splitting model in HEVC.

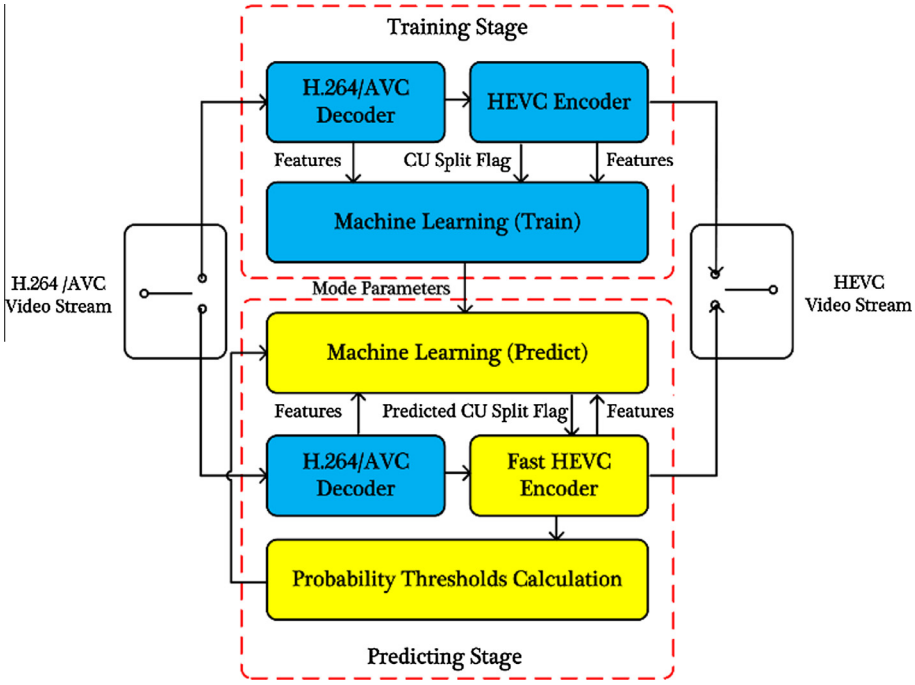


Fig. 5. Framework of the proposed H.264/AVC to HEVC transcoding.

saving is defined as TS_1; if all three classifiers are activated, the time saving is defined as TS_2. Then, TS_0, TS_1 and TS_2 can be calculated by

$$TS_0 = 1 - \frac{A_0D_0 + 4(1-A_0)D_1 + 16(1-A_0)D_2 + 64(1-A_0)D_3}{D_0 + 4D_1 + 16D_2 + 64D_3}, \quad (1)$$

$$TS_1 = 1 - \frac{A_0D_0 + 4A_1D_1 + 16(1-A_0-A_1)D_2 + 64(1-A_0-A_1)D_3}{D_0 + 4D_1 + 16D_2 + 64D_3}, \quad (2)$$

$$TS_2 = 1 - \frac{A_0D_0 + 4A_1D_1 + 16A_2D_2 + 64A_3D_3}{D_0 + 4D_1 + 16D_2 + 64D_3}. \quad (3)$$

The right three columns in Table 1 illustrate the time saving for different sequences. We can observe that TS_0, TS_1 and TS_2 reach 42.67%, 70.54%, and 77.74% on average, respectively, which are the upper bounds of the complexity reduction in CU depth prediction. Also, they indicate that huge complexity reduction can be achieved if the CU depth can be precisely predicted.

4. Proposed machine learning based fast H.264/AVC to HEVC transcoding

4.1. Framework of the proposed transcoder

Fig. 5 shows the framework of the proposed H.264/AVC to HEVC transcoding method, which has two stages, namely the training and predicting stages. During training stage, H.264/AVC decoder and HEVC encoder are connected as a cascaded transcoder, in which CU splitting flags from the original HEVC encoder are collected and regarded as the ground truth. At the same time, feature vectors from the H.264/AVC bit-stream and HEVC coding process are also extracted. Then, these extracted feature vectors and labeled CU splitting flags (ground truth) are used as input to learning machine (train) to generate model parameters of the classifiers. Then, the trained models are recorded and utilized to predict CU splitting flags at the predicting stage. At the predicting stage, CU splitting flags of encoding the current CU are predicted by the machine learning algorithm (predict), then the predicted flags are directly used in HEVC to set the CU partition and skip the unnecessary checking operations, which lowers the complexity of the HEVC encoder.

Switchers are required in this framework to switch between the training and the predicting stages depending on the video content, learning algorithm and parameters. The more frames are encoded at the training stage, the better RD performance may be achieved, since more samples could be used for training. However the coding complexity will be increased as more frames are encoded with the original HEVC encoder. There is a trade-off among the learning robust, RD performance and the coding complexity. The optimization will be discussed in detail in Section 4.4.

The proposed framework in Fig. 5 is inspired by the transcoding architecture in [19]. Compared with [19], the major novelties and differences of this work are listed as follows: (1) Since features is of great importance to the prediction accuracy, optimal feature combinations for the classifiers are selected based on the proposed feature selection algorithm; (2) To have a good trade-off between complexity reduction and RD performance, a probability threshold and an adaptive threshold updating mechanism are proposed.

4.2. SVM based learning algorithm and the decision function

In this paper, the splitting and non-splitting prediction is a binary classification issue. Thus, SVM [24] is utilized due to its robustness and good classification performance in solving numerous realistic classification problems, especially the binary classification issues. The main idea of SVM is to achieve an optimal classification hyper-plane with the maximum margin between two classes. The decision function [24] is:

$$f(\mathbf{x}) = \boldsymbol{\omega}^T \phi(\mathbf{x}) + b, \quad (4)$$

$$y(\mathbf{x}) = \mathrm{sign}(f(\mathbf{x})) = \begin{cases} +1 \\ -1 \end{cases}, \quad (5)$$

where $\boldsymbol{\omega}$ is a normal vector to a hyper-plane of classification, the function $\phi(\mathbf{x})$ maps feature vector $\mathbf{x}$ into a higher dimensional space, and b is the bias term. The feature vector $\mathbf{x}$ is one of key components that affects the prediction accuracy. To improve the

Table 2
Feature candidates and their CC / CV values.

Index	Source	Feature description		Performance evaluation					
				Classifier 0		Classifier 1		Classifier 2	
				CC	CV (%)	CC	CV (%)	CC	CV (%)
1	Video Content	Variance of luminance [19]		0.125	61.8	0.143	57.9	0.014	68.9
2		Number of edge pixel		0.394	63.5	0.131	58.9	0.084	68.9
3		Gradient	Variance of gradient [21]	0.157	61.8	0.159	57.9	0.021	68.9
4			Mean of gradient [21]	0.199	61.8	0.088	57.9	0.023	68.9
5*		SAD between current and co-located blocks		0.446	76.1	0.108	57.9	0.116	68.9
6*	H.264 Bit Stream	Coded Block Pattern (CBP) [19]		0.465	77.0	0.183	57.9	0.146	68.9
7*		MB partition types [19]		0.538	79.1	0.234	57.9	0.274	72.9
8*		DCT Coefficient	Number of non-zero DCT coefficients [20]	0.442	75.8	0.207	57.9	0.205	69.3
9*			DCT coefficients energy [20]	0.358	70.9	0.155	57.9	0.140	69.2
10		Motion Vector	Mean of amplitude [20]	0.065	61.8	0.026	57.9	0.038	68.9
11			Variance of amplitude [20]	0.027	61.8	0.010	57.9	0.060	68.9
12			Variance of phase [20]	0.064	61.8	0.069	57.9	0.051	68.9
13	Context Information of the HEVC	Adjacent Encoded CUs	CBF	0.149	63.5	0.053	57.9	0.066	68.9
14	Encoder		RD Cost	0.146	63.5	0.099	57.9	0.082	68.9
15*			CU Depth [21]	0.622	77.9	0.066	57.9	0.181	74.4
16			Distortion	0.097	61.8	0.125	57.6	0.092	68.9
17			Number of bits	0.204	63.1	0.056	57.9	0.025	68.9
18*			Merge Flag	0.067	61.8	0.249	57.9	0.407	78.9
19*			Skip Flag	0.199	62.8	0.238	57.9	0.371	77.2
20*			Ctx Skip Flag	0.068	61.8	0.215	57.9	0.351	78.4
21*		By-product Information of the Current CU	CBF [21]	0.607	80.8	0.375	65.0	0.311	69.6
22*			RD Cost	0.240	65.3	0.115	57.9	0.136	68.9
23*			Skip Flag	0.839	92.4	0.698	85.4	0.542	73.9
24*			MergeFlag	0.607	80.9	0.479	71.9	0.386	73.5

Note: Yellow shadow and * indicates the selected features.

prediction accuracy, in Section 4.3, we will analyze feature vectors in detail and present a feature selection algorithm. The Radial Basis Function (RBF) kernel is used with kernel parameter and regularization parameters set as 0.5 and 1.

SVM outputs binary values (+1 or −1). However, in some real applications, there usually exists an uncertain situation which is of high false risk for either positive (+1) or negative (−1) prediction. In HEVC CU depth prediction, the positive and negative samples are very close to each other at given features in some cases. Either positive or negative prediction is probably false and causes large RD degradations. Thus, to handle this problem, we map the SVM binary outputs into probabilities with an additional sigmoid function [25]. The probability can be defined as:

$$p(y(\mathbf{x}) = +1|\mathbf{x}) = \frac{1}{1 + e^{Af(\mathbf{x})+B}}, \tag{6}$$

$$p(y(\mathbf{x}) = -1|\mathbf{x}) = 1 - p(y(\mathbf{x}) = +1|\mathbf{x}), \tag{7}$$

where A and B are model parameters in the sigmoid function. By using this probability, we mapped the SVM binary outputs into three outputs, *i.e.* +1, −1 and 0 (uncertain). The decision function is presented as

$$Y(\mathbf{x}) = \begin{cases} +1 & f(\mathbf{x}) > 0 \& p(y(\mathbf{x}) = +1|\mathbf{x}) \geqslant \theta_\varphi \\ 0 & else \\ -1 & f(\mathbf{x}) < 0 \& p(y(\mathbf{x}) = -1|\mathbf{x}) \geqslant \theta_\varphi \end{cases}, \tag{8}$$

where θ_φ is a threshold, and $\varphi \in$ {Classifier 0, Classifier 1, Classifier 2}. As for the positive (+1) prediction, the current CU will be directly split into four sub-CUs and skip checking the current CU depth. For the negative prediction (−1), the current CU will only check the current CU depth and will not be split. As for the uncertain prediction (0), the original CU coding via RD comparison between the RD costs of the current CU depth and sub-CU depth is used to maintain the RD performance, which is the same as the original HEVC encoder. It shall be noted that the threshold θ_φ is a key parameter that not only affects the prediction accuracy but also influences the coding performance. It is analyzed in detail in Section 4.4.

4.3. Feature selection

Features are essential to the classification since good features make classes more distinguishable and bad features may deteriorate the prediction accuracy. Inspired by recent state-of-the-art works on fast video coding and transcoding [19–21], we summarize them and come up with 24 feature candidates for the CU depth prediction, as listed in the left columns in Table 2. These feature candidates are basically extracted from three major sources, including video content, H.264/AVC bit stream and context information of the HEVC encoder.

(1) The video content features include variance of image luminance [19], number of edge pixels from edge detection, and the gradient [21]. These features mainly represent the texture information of the video content. Sum of Absolute Difference (SAD) between the current and temporal co-located blocks is a kind of joint information of motion and texture for the video content.
(2) As for the features from H.264/AVC bit stream, we use the common and prevailing features including the Coded Block Pattern (CBP), MB modes [19], DCT coefficients [19] and MVs [20].
(3) In addition, the context information of the HEVC encoder is also exploited. The context features are mainly summarized into two aspects, one is the spatial and temporal adjacent encoded CUs including the above-left, above, above-right, left and temporal co-located CUs corresponding to the current CU. The other is previously checked or by-product output information of the current CU. Features from adjacent

encoded CUs are CBF, average RD cost, optimal CU depth [21], average distortion, number of bits, merge flags, skip flags and context skip flags. Additionally, the by-product output information of the current CU includes CBF, RD cost, skip flag and merge flag after checking SKIP/Merge. Yet, the rest of PU modes of the current CU are not checked.

Effective features are helpful to discriminate the classes and improve the prediction accuracy. However, inappropriate features may play a negative impact on the prediction accuracy and meanwhile cause additional complexity overhead. Using full set of features may have a good prediction accuracy, but tends to increase time consumption overhead of feature generation and classifier training, as well as predicting complexities. We thus propose a feature selection algorithm to select representative features for the SVM classifiers.

To analyze and select better features for the classifiers, three sequences with different resolutions and motion properties, BasketballPass (416×240), BQMall (832×480) and FourPeople (1280×720), were tested for the feature selection. Four quantization parameters (QPs) with 24, 28, 32 and 36 were used to encode these test sequences. Then, values of the 24 feature candidates and the ground truth CU splitting flags were recorded from the inter frame coding. The outputs of the SVM classifier with different combinations of feature candidates were verified by comparing with the CU splitting flags (ground truth).

Pearson Correlation Coefficient (CC) and Cross Validation (CV) accuracy [24] are employed to evaluate the effectiveness of these feature candidates for the classification. CC is calculated between the feature value and the labeled CU splitting flag (Ground truth). Suppose p and q are the individual feature value and the corresponding ground truth in a sample, the number of samples is N, the value of CC can be calculated as

$$CC = \frac{\sum_{i=1}^{N}\left(p_i - \frac{1}{N}\sum_{j=1}^{N}p_j\right)\left(q_i - \frac{1}{N}\sum_{j=1}^{N}q_j\right)}{\sqrt{\sum_{i=1}^{N}\left(p_i - \frac{1}{N}\sum_{j=1}^{N}p_j\right)^2} \times \sqrt{\sum_{i=1}^{N}\left(q_i - \frac{1}{N}\sum_{j=1}^{N}q_j\right)^2}}. \quad (9)$$

The CV is a common operation in machine learning to measure the stability of current trained classifier. In this paper, 10-fold CV is used to validate the prediction accuracy. 9/10 of total samples are used to train and 1/10 of total samples are used in the test. There are ten iterations in total. At last the average prediction accuracy is output.

The right six columns in Table 2 list the CC and CV for each individual feature candidate, where the higher CC and CV values indicate better performance of using corresponding features. We can find that Feature 23 (Skip Flag) has the highest CC values for three classifiers, which are 0.839, 0.698, and 0.542, respectively. The Skip flag also has the highest CV accuracy for Classifier 0 and Classifier 1 among the candidate features, and the accuracies reach 92.4% and 85.4%, respectively. Whereas, for Classifier 2, Feature 18 (Merge Flag) performs the best on CV accuracy. Consequently, features with high CC and CV accuracy would be utilized to predict CU splitting flag in transcoding. Additionally, we observe that H.264/AVC stream information and previous encoded information are more important for classifiers in the proposed transcoding method, especially, Merge Flag and Skip Flag features from former checked modes. Table 2 shows the CC and CV of using individual feature. However, using one feature is optimal while it does not guarantee to be the best when they are jointly used in the classification. Thus, the performance of feature combinations is investigated. Suppose the total number of candidate features is M, and the selected features is K, feature combination means selecting K features out of M candidate features.

To obtain the best feature combination, we thus propose a feature selection algorithm, which mainly consists of two steps: (1) iteratively calculate the cost (*i.e.* CC or CV) of each feature combination, and (2) select the feature combination with the minimum cost (*i.e.* maximum CC or CV) as the best. This feature selection process is off-line and will not affect the complexity of the designed transcoder. Details of the feature selection algorithm are shown as follow, where *feature[]* denotes input feature candidates, M is the total number of feature candidates in *feature[]*, K is the number of features that will be adopted in the transcoder, *best_features[]* means the selected features.

Algorithm: Feature selection algorithm

Input: features[], M
Initialization: min_cost = MAX_VALUE
Output: best_features[], K
1: **FOR** $1 \leqslant i \leqslant M$ **DO**
2: Calculate the total number of combinations with given M and i, $n = \frac{i!}{i! \times (M-i)!}$
3: **FOR** $0 \leqslant j < n$ **DO**
4: Get a combination of i features, and store it in *temp_feature[][]*
5: Calculate the classification cost (*temp_cost*) of using the temporal feature, *temp_feature*[][]
6: **IF** *temp_cost* < *min_cost* **THEN**
7: *min_cost* = *temp_cost*
8: *best_features*[] = *temp_features*[*i*][*j*]
9: *K* = *j*
10: **END IF**
10: **END FOR**
11: **END FOR**

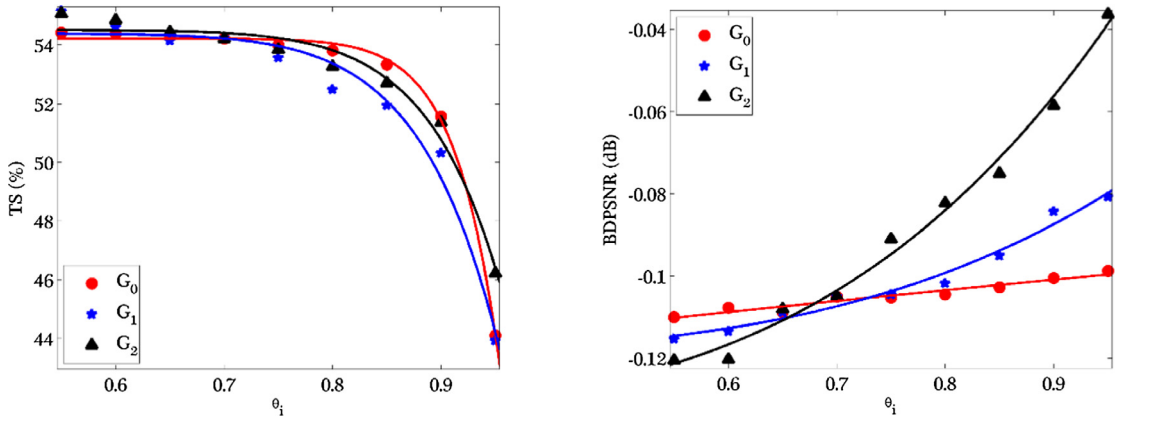
According to the CV value and the feature selection algorithm, we finally select out the 13 features for classification, which are marked with yellow shadow and * in Table 2, since these 13 features provide the maximum CV (92.5%, 89.0%, and 89.8% for Classifier 0, Classifier 1 and Classifier 2 respectively) in the SVM classification. The rest features, such as texture, edge and gradient, which actually have little contribution to CU splitting prediction accuracy, are ignored in this paper.

4.4. Probability threshold determination

In Eq. (8), different probability thresholds θ_φ influence the number of positive (+1) and negative (−1) predictions and have different trade-offs between computational complexity and RD performance. Larger threshold means better RD performance and limited time saving, since more CUs will be selected using the RDO process in Eq. (8). Smaller threshold leads to lower complexity; however it may cause worse RD performance due to false prediction. Thus, an appropriate threshold should be determined. In this subsection, we propose an adaptive threshold scheme to determine this probability threshold.

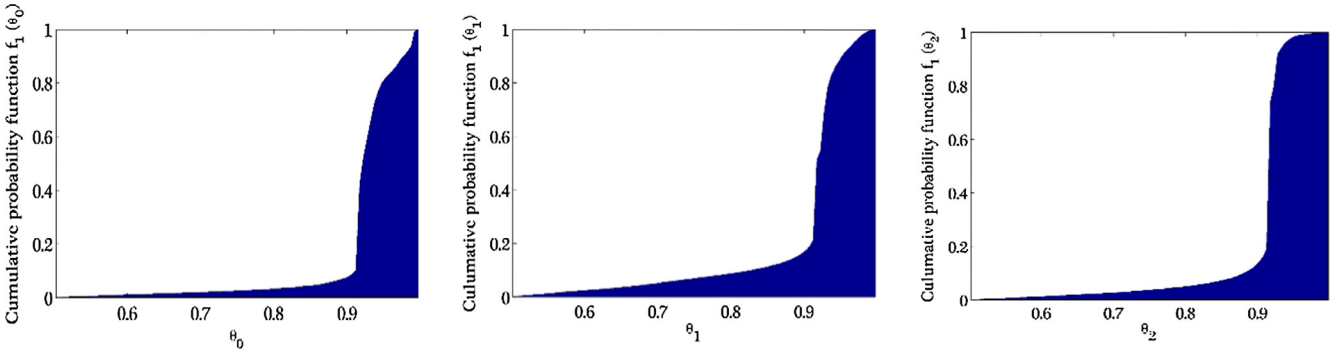
4.4.1. Initial threshold determination

To determine the optimal initial thresholds (denoted as θ_0, θ_1 and θ_2) for the three levels of the classifiers, statistical experiments with different thresholds were performed over four test sequences, including BasketballPass (416×240), BQMall (832×240), FourPeople (1280×720) and PartySence (1920×1080). 25 frames in each sequence were transcoded, where one intra frame was encoded by the original HEVC and the next four inter frames were used for training, the rest 20 frames were transcoded with optimization. Three threshold sets ($\mathbf{G}_i$, $i = \{0,1,2\}$) were used and they



(a) Relationship between time saving and thresholds (b) Relationship between BDPSNR and thresholds

Fig. 6. Time saving ΔT and BDPSNR for different thresholds.



(a) Classifier 0 (b) Classifier 1 (c) Classifier 2

Fig. 7. Decision cumulative probability histogram for different classifiers.

are defined as $\mathbf{G}_0 = \{\theta_0, 0.7, 0.7\}$, $\mathbf{G}_1 = \{0.7, \theta_1, 0.7\}$, $\mathbf{G}_2 = \{0.7, 0.7, \theta_2\}$. In each $\mathbf{G}_i$, θ_i changed from 0.55 to 0.95 with the step of 0.05. Time saving (ΔT) and Bjøntegaard Delta Peak-Signal-to-Noise Ratio (BDPSNR) [26] calculated between the original cascade transcoder and proposed transcoder were used to evaluate the transcoding complexity and the compression efficiency respectively. Time saving ΔT is defined as:

$$\Delta T = \frac{1}{4}\sum_{i=1}^{4}\frac{T_c(QP_i) - T_\psi(QP_i)}{T_c(QP_i)} \times 100\%, \tag{10}$$

where $T_c(QP_i)$ and $T_\psi(QP_i)$ are the transcoding time of the original cascaded transcoder [1] and scheme Ψ with QP_i, here Ψ indicates the proposed method.

Fig. 6 shows ΔT and BDPSNR for different threshold sets $\mathbf{G}_i$. The ΔT and BDPSNR are the average values collected from the four test sequences. Fig. 6(a) shows the relationship between ΔT and threshold, where the dots with different symbols are real collected data and the curves are fitting results. When θ_i in each $\mathbf{G}_i$ is less than 0.9, ΔT are all larger than 50%, which is a significant complexity reduction. Fig. 6(b) shows the relationship between BDPSNR and threshold. We can observe that the $\mathbf{G}_2$ curve is the steepest and $\mathbf{G}_0$ is the flattest, which means that $\mathbf{G}_2$ is sensitive to the changes of θ_i, and $\mathbf{G}_0$ is less sensitive. The BDPSNR degradation reduces as θ_i increases. According to Fig. 6(a) and (b), the larger θ_i in $\mathbf{G}_i$ leads to the better BDPSNR as well as smaller ΔT. To achieve a better trade-off between ΔT and RD performance of the transcoding, we select θ_i in $\mathbf{G}_i$ as the BDPSNR is larger than −0.10 dB and the initial thresholds are set as 0.75, 0.80, and 0.85, respectively.

4.4.2. Adaptive threshold updating scheme

The initial threshold sets are obtained from the off-line statistical data. Though it can have a trade-off between the complexity and RD performance in global aspect, it can hardly get the optimal trade-off in every short time since video content and properties vary along time. To solve this problem, an adaptive threshold scheme is proposed. The transcoding optimization target is to maximize the computational complexity reduction while maintaining the RD performance, thus, the objective can be modeled as

$$\max(\Delta T),\ s.t.\{P \leqslant P_0\}, \tag{11}$$

where P is transcoding RD degradation, *e.g.* BDPSNR, and P_0 is the upper bound of allowable transcoding RD degradation. We find ΔT is inverse proportional to the number of CUs with zero prediction in Eq. (8). The BDPSNR value depends on the prediction accuracy of +1/−1 predictions. Thus, Eq. (10) can be rewritten as

$$\max[1 - f_1(\theta)], s.t.\{f_2(\theta) \geqslant t_0\}, \tag{12}$$

where $f_1(\theta)$ is the percentage of CUs with zero prediction for different θ in Eq. (8), $f_2(\theta)$ is prediction accuracy of +1/−1 predictions, and t_0 is a lower bound of the prediction accuracy. Lagrange multiplier is introduced to solve Eq. (12) and the optimization target is rewritten as

$$\max J(\theta), J(\theta) = [1 - f_1(\theta)] + \lambda \times [f_2(\theta) - t_0], \tag{13}$$

Table 3
CU depth prediction accuracy fitting parameters.

Fitting parameters				
Item		Classifier 0	Classifier 1	Classifier 2
Fitting function		$f_2(\theta) = a\theta^b + c$		
Fitting Parameters	a	0.1108	0.1938	0.1358
	b	9.954	10.04	8.477
	c	0.921	0.8628	0.8912
Fitting accuracy				
SSE		0.00008	0.00027	0.00014
R^2		0.9797	0.9782	0.9805
Adjusted R^2		0.9729	0.9710	0.9740
RMSE		0.00375	0.00676	0.00485

where λ is the Lagrange multiplier. It is easy to prove that Eq. (13) is convex, thus, we can obtain the optimal θ by taking derivative of Eq. (13) and setting it to zero, which is

$$\frac{\partial J}{\partial \theta} = -f_1'(\theta) + \lambda f_2'(\theta) \equiv 0. \tag{14}$$

From Eq. (14), the Lagrange multiplier can be calculated as

$$\lambda = \frac{f_1'(\theta)}{f_2'(\theta)}. \tag{15}$$

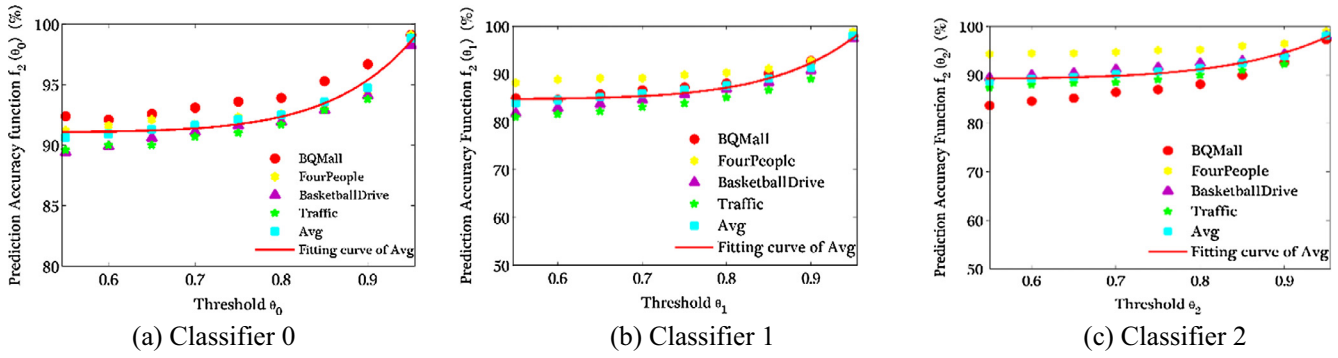
In order to get the optimal threshold θ, the Lagrange multiplier shall be determined firstly and now the problem becomes to calculate the derivatives $f_1'(\theta)$ and $f_2'(\theta)$.For the discrete data, the derivative $f_1'(\theta)$ can be written as

$$f_1'(\theta) = \frac{f_1(\theta+\mu) - f_1(\theta)}{\mu}, \tag{16}$$

where μ is a small interval value for θ. Actually, the Cumulative Probability Function (CPF) $f_1(\theta)$ can be statistically collected in the coding process, it satisfies $f_1(\theta_1) \geqslant f_1(\theta_2)$ when $\theta_1 > \theta_2$, and $0 \leqslant f_1(\theta) \leqslant 1$. Therefore, $f_1(\theta+\mu)$ and $f_1(\theta)$ can be obtained from given μ and θ. Fig. 7 shows an example of CPF $f_1(\theta)$ for three levels of classifiers, which is collected from on-line coding. The horizontal axis is θ and the vertical axis is $f_1(\theta)$.

In addition to the percentage of $f_1(\theta)$ (zero-prediction), the prediction accuracy of +1/−1 predictions, $f_2(\theta)$, is also analyzed. Four test sequences were used in this experiment, including BQMall (832×480), FourPeople (1280×720), BasketballDrive (1920×1080) and Traffic (2560×1600). These H.264/AVC streams were all transcoded to HEVC with QP value 28, and the thresholds for three classifiers are the same and vary from 0.55 to 0.95. 41 frames were transcoded, in which one intra frame was encoded by the original HEVC and four inter frames were used for training, the rest frame were transcoded with low complexity optimization. In the prediction, if the decision value is 0 in Eq. (8), the full RDO process is activated, thus, this CU depth prediction can be regarded as 100% accuracy.

Fig. 8 shows the CU depth prediction accuracy $f_2(\theta)$ for three levels of classifiers, where the dots are real collected data and curves are fitting results. We can observe that $f_2(\theta)$ of three levels of classifiers increase as the thresholds increase. It indicates that better RD performance could be achieved with larger thresholds since the prediction accuracies are improved. Moreover, we also notice that the prediction accuracy of Classifier 0 reaches higher than 91% on average for different thresholds. The reason is the selected features at classifier level 0 have better CC and higher CV accuracy. In Fig. 8, the real collected data is fitted by power function. We can find that the value of $f_2(\theta)$ is similar value among different sequences. Table 3 shows the fitting parameters and fitting accuracy from the average value, which is measured by SSE, R^2, adjusted R^2 and RMSE. The R^2 and adjusted R^2 are higher than



(a) Classifier 0 (b) Classifier 1 (c) Classifier 2

Fig. 8. CU depth prediction accuracies with three classifiers.

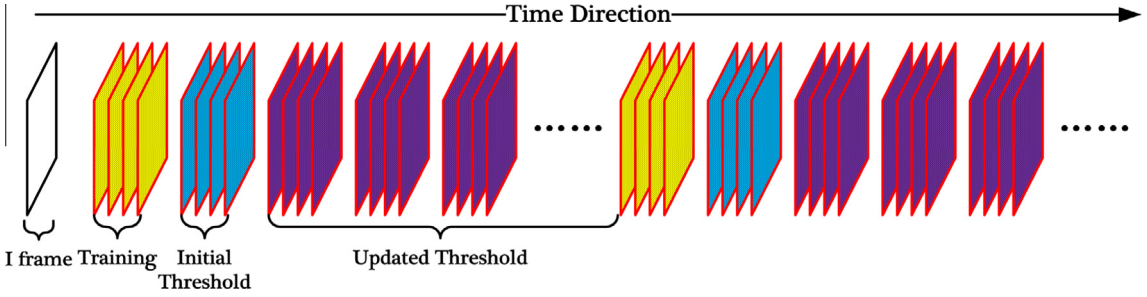


Fig. 9. Adaptive threshold updating mechanism.

Table 4
Complexity and RD performance of three decision levels in Proposed_Adapt.

Set	Class	Sequence	L_0			$L_0 + L_1$			$L_0 + L_1 + L_2$		
			ΔT (%)	BDBR (%)	BDPSNR (dB)	ΔT (%)	BDBR (%)	BDPSNR (dB)	ΔT (%)	BDBR (%)	BDPSNR (dB)
Set1		Traffic	28.4	0.147	−0.008	48.7	0.992	−0.038	51.9	2.257	−0.082
	B	BasketballDrive	21.5	0.460	−0.020	44.5	1.772	−0.07	54.2	3.204	−0.125
	C	BQMall	16.7	0.243	−0.012	31.5	0.863	−0.040	43.7	1.976	−0.090
		PartyScene	10.4	0.061	−0.008	25.6	0.209	−0.013	39.0	1.230	−0.055
	D	BasketballPass	12.3	−0.878	0.032	29.0	−0.440	0.012	40.9	2.189	−0.113
	E	FourPeople	49.3	0.226	−0.011	59.2	0.747	−0.035	65.4	1.638	−0.076
	Average		**23.1**	**0.043**	**−0.005**	**39.8**	**0.691**	**−0.031**	**49.2**	**2.082**	**−0.090**
Set2	A	PeopleOnStreet	13.4	0.155	−0.013	21.9	0.75	−0.043	27.2	2.753	−0.145
	B	BQTerrace	24.8	−0.393	−0.004	42.0	0.090	−0.012	46.4	1.158	−0.037
		Cactus	25.0	−0.023	−0.002	41.8	0.967	−0.041	51.2	2.260	−0.091
		Kimono1	15.2	0.401	−0.017	47.6	2.408	−0.106	55.5	2.732	−0.120
		ParkScene	19.0	0.075	−0.007	39.5	0.805	−0.035	48.8	1.821	−0.073
	C	RaceHorsesC	11.0	0.053	−0.006	31.4	0.607	−0.028	45.0	1.569	−0.072
		BasketballDrill	21.6	0.456	−0.022	39.8	1.167	−0.050	51.9	2.918	−0.121
	D	Blowing.Bubbles	7.0	−0.473	0.007	27.6	−0.152	−0.009	41.6	2.314	−0.105
		BQSquare	5.4	0.195	−0.006	29.3	0.336	−0.010	35.0	0.754	−0.025
		RaceHorses	0.8	−0.345	0.004	24.6	0.304	−0.023	37.4	1.722	−0.091
	E	Johnny	48.1	0.505	−0.019	59.3	0.433	−0.017	64.8	1.340	−0.049
		KristenAndSara	40.1	−0.091	0.003	50.6	0.621	−0.027	61.7	1.245	−0.053
		Video 1	45.1	0.053	−0.003	59.8	0.941	−0.038	65.7	1.745	−0.070
		Video 3	48.8	0.941	−0.044	59.4	1.869	−0.084	65.6	2.621	−0.116
		Video 4	40.0	0.428	−0.017	53.7	1.443	−0.056	61.3	2.270	−0.088
	Average		**24.4**	**0.129**	**−0.010**	**41.9**	**0.839**	**−0.039**	**50.6**	**1.948**	**−0.084**
Average of all sequences			23.9	0.105	−0.008	41.3	0.796	−0.036	50.2	1.986	−0.085

0.97, and SSE and RMSE approach zero. Thus, we can conclude the fitting accuracy is high enough and $f_2(\theta)$ can be approximated as a power function of θ, noted as

$$f_2(\theta) = a\theta^b + c, \tag{17}$$

where a, b and c are fitting parameters. According to Fig. 8, we also find that the prediction accuracy of different sequences are similar, therefore, we can use the average value to predict the prediction accuracies for different sequences, instead of on-line prediction for each sequence.

Substitute Eqs. (16) and (17) to Eq. (15), the Lagrange multiplier can be presented as

$$\lambda = \frac{f_1(\theta+\mu) - f_1(\theta)}{ab\mu\theta^{b-1}}. \tag{18}$$

Applying Eq. (18) to Eq. (13), we can get the optimal θ by

$$\theta^* = \arg\max\left\{[1 - f_1(\theta)] + \frac{f_1(\theta+\mu) - f_1(\theta)}{ab\mu\theta^{b-1}}[a\theta^b + c - t_0]\right\}. \tag{19}$$

Since $f_1(\theta)$ is collected from the transcoding process, the optimal θ is obtained in the on-line transcoding using Eq. (19). Fig. 9 shows the mechanism of adaptive threshold updating and calculation. There are four kinds of frames in video sequences. One is intra frame (white rectangle). The second is a group of training frames (yellow rectangles), which are encoded with the original HEVC. From these frames, the ground truth data, such as CU depth, features, are collected and the SVM classifiers are trained over these ground truth data. The third is the inter frames optimized with initial thresholds, shown as the light blue rectangles. These frames are optimized with CU depth prediction by using the initial thresholds, and meanwhile, some statistical information, including $f_1(\theta)$, *etc.* are collected for optimal θ determination of the following frames. Finally, as the fourth kind of frames, the optimization frames (purple color) are encoded with fast coding optimization and the optimal θ is updated as well. The four kinds of frames are encoded sequentially and will be repeated periodically.

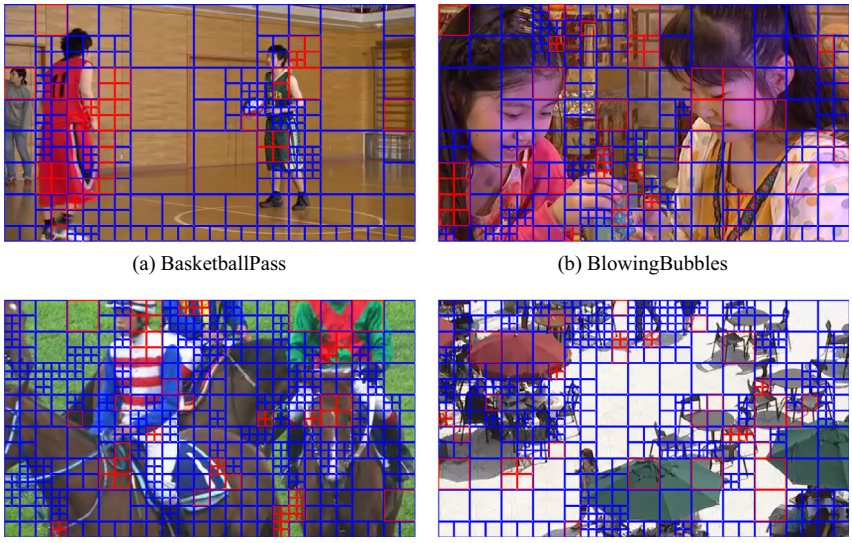
The yellow frames are used for model training, they are unoptimized frames. Thus, the complexity reduction will be reduced as the percentage of yellow frames increase. Meanwhile, if the number of these yellow training frames is too small, it will make the training model unstable. In this paper, the number of yellow frames is set as 4 in one cycle. The third kind of frames are optimized frames and used to calculate $f_1(\theta)$. Meanwhile, $f_1(\theta)$ is frequently updated during encoding the fourth kind of frames. Therefore, we set the number of light blue frames as 4 as well.

Algorithm: Optimal threshold decision algorithm for θ

Input: n, N, θ', ω, a, b, c, t_0
Initialization: $m[\,] = 0$
Output: θ^*
FOR $1 \leqslant i \leqslant N$ **DO**
  CALCULATE $p(\mathbf{x}_i)$ by Eqs. (6) and (7)
  FOR $1 \leqslant k \leqslant n$ **DO**
    IF $p(\mathbf{x}_i) < 0.5 + \frac{0.5}{n} \times k$ **THEN**
      $m[k-1] = m[k-1] + \frac{1}{N}$
    END IF
  END FOR
END FOR
SOLVE

$$k^* = arg \max_k \left(1 - m[k-1] + 2n \times \frac{m[k] - m[k-1]}{ab \times \left(\frac{1}{2} + \frac{1}{2n} \times (k-1)\right)^{b-1}} \times \left(a\left(\frac{1}{2} + \frac{1}{2n} \times (k-1)\right)^b + c - t_0\right)\right)$$

CALCULATE $\theta = 0.5 + \frac{0.5}{n} \times (k^* - 1)$
CALCULATE the final threshold $\theta^* = \theta \times \omega + \theta' \times (1 - \omega)$



(a) BasketballPass (b) BlowingBubbles

(c) RaceHorses (d) BQSquare

Fig. 10. CU partition prediction by the proposed algorithm.

Table 5
Complexity and RD performance comparison between the proposed algorithm and benchmarks (LP).

Set	Class	Sequence	ICIP2012 [12]			CSVT2014 [20]			JM Decoder + TMM2013 [30]			Proposed_Adpt		
			ΔT (%)	BD BR (%)	BD PSNR (dB)	ΔT (%)	BD BR (%)	BD PSNR (dB)	ΔT (%)	BD BR (%)	BD PSNR (dB)	ΔT (%)	BD BR (%)	BD PSNR (dB)
Set1	A	Traffic	40.4	2.595	−0.094	43.0	2.021	−0.073	33.7	1.635	−0.059	51.9	2.257	−0.082
	B	BasketballDrive	41.2	8.104	−0.315	41.5	1.823	−0.071	20.8	1.906	−0.075	54.2	3.204	−0.125
	C	BQMall	40.1	3.616	−0.170	46.5	3.143	−0.149	27.1	4.909	−0.234	43.7	1.976	−0.090
		PartyScene	38.1	1.833	−0.078	39.4	1.832	−0.079	26.3	1.164	−0.048	39.0	1.230	−0.055
	D	BasketballPass	35.1	1.968	−0.099	47.9	3.199	−0.161	26.6	4.091	−0.202	40.9	2.189	−0.113
	E	FourPeople	39.4	3.004	−0.139	54.7	2.262	−0.105	52.1	4.391	−0.204	65.4	1.638	−0.076
	Average		**39.1**	**3.520**	**−0.149**	**45.5**	**2.380**	**−0.106**	**31.1**	**3.016**	**−0.137**	**49.2**	**2.082**	**−0.090**
Set2	A	PeopleOnStreet*	43.2	3.563	−0.186	47.7	2.258	−0.119	20.8	2.176	−0.113	27.2	2.753	−0.145
	B	BQTerrace	39.2	1.634	−0.048	43.1	1.759	−0.048	33.3	1.440	−0.040	46.4	1.158	−0.037
		Cactus	40.2	3.530	−0.142	42.7	1.643	−0.066	33.6	3.078	−0.125	51.2	2.260	−0.091
		Kimono1	40.5	4.852	−0.211	44.9	1.308	−0.057	15.2	0.552	−0.024	55.5	2.732	−0.120
		ParkScene	39.9	2.091	−0.083	44.8	1.368	−0.054	30.2	1.304	−0.051	48.8	1.821	−0.073
	C	RaceHorsesC	40.9	3.301	−0.150	44.8	2.498	−0.115	20.3	2.646	−0.118	45.0	1.569	−0.072
		BasketballDrill	41.8	4.115	−0.170	43.9	1.977	−0.081	26.3	3.116	−0.128	51.9	2.918	−0.121
	D	Blowing.Bubbles*	37.3	2.313	−0.098	42.3	1.825	−0.076	20.4	1.242	−0.053	41.6	2.314	−0.105
		BQSquare	40.7	1.335	−0.049	42.9	3.100	−0.113	33.2	3.461	−0.126	35.0	0.754	−0.025
		RaceHorses	38.7	2.787	−0.139	43.4	2.894	−0.145	20.1	1.518	−0.079	37.4	1.722	−0.091
	E	Johnny	39.1	3.080	−0.112	48.4	2.451	−0.089	51.9	3.303	−0.120	64.8	1.340	−0.049
		KristenAndSara	38.6	2.892	−0.122	53.2	2.593	−0.109	47.6	4.847	−0.205	61.7	1.245	−0.053
		Vidyo 1	38.2	2.287	−0.091	53.4	1.707	−0.068	51.2	3.177	−0.128	65.7	1.745	−0.070
		Vidyo 3	39.9	3.939	−0.174	54.4	4.013	−0.176	50.4	3.140	−0.141	65.6	2.621	−0.116
		Vidyo 4	40.4	3.481	−0.137	50.3	1.764	−0.070	45.4	4.361	−0.173	61.3	2.270	−0.088
	Average		**39.9**	**3.013**	**−0.127**	**46.7**	**2.211**	**−0.092**	**33.3**	**2.624**	**−0.108**	**50.6**	**1.948**	**−0.084**
Average of all sequences			39.7	3.158	−0.134	46.3	2.259	−0.096	32.7	2.736	−0.116	50.2	1.986	−0.085

The optimal threshold decision algorithm for θ is presented as above. The input n is the number of patches of the cumulative probability histogram, which represents the fidelity of θ. N is the total number of CU of the light blue frames (the third type). $m[\,]$ is the value of $f_1(\theta)$, θ' is the used θ of previous frames. $\omega \in [0,1]$ is a weighting factor for the updating. Smaller ω leads to stable θ while larger ω may make θ sensitive to the video content. In this paper, ω is set as 0.85. The proposed scheme using the adaptive threshold is denoted as **Proposed_Adapt** for short.

Table 6
Complexity and RD performance comparison between the proposed algorithm and benchmarks (RA).

Set	Class	Sequence	ICIP2012 [12]			CSVT2014 [20]			JM Decoder + TMM2013 [30]			Proposed_Adpt		
			ΔT (%)	BD BR (%)	BD PSNR (dB)	ΔT (%)	BD BR (%)	BD PSNR (dB)	ΔT (%)	BD BR (%)	BD PSNR (dB)	ΔT (%)	BD BR (%)	BD PSNR (dB)
Set1	A	Traffic	43.2	1.920	−0.071	32.5	1.733	−0.069	43.1	1.266	−0.044	51.8	2.510	−0.110
	B	BasketballDrive	43.3	7.657	−0.246	37.6	3.703	−0.131	34.3	4.607	−0.170	42.0	4.336	−0.148
	C	BQMall	42.9	3.539	−0.149	38.7	2.945	−0.144	36.9	4.184	−0.207	43.4	2.241	−0.105
		PartyScene	41.9	1.520	−0.059	43.2	1.897	−0.078	37.4	1.514	−0.068	39.8	2.509	−0.078
	D	BasketballPass	36.9	2.432	−0.104	43.6	4.173	−0.200	35.5	3.985	−0.183	41.6	2.338	−0.094
	E	FourPeople	42.4	1.387	−0.074	53.3	2.048	−0.088	58.8	1.904	−0.102	62.1	0.982	−0.063
	Average		**41.8**	**3.076**	**−0.117**	**41.5**	**2.750**	**−0.118**	**41.0**	**2.910**	**−0.129**	**46.8**	**2.486**	**−0.100**
Set2	A	PeopleOnStreet	45.7	3.876	−0.167	50.7	3.650	−0.164	29.9	2.169	−0.113	29.1	2.443	−0.140
	B	BQTerrace	42.8	1.184	−0.034	42.0	2.334	−0.075	43.5	1.279	−0.034	56.0	1.746	−0.052
		Cactus	44.0	3.429	−0.125	38.9	2.041	−0.077	45.9	3.417	−0.144	48.8	2.572	−0.115
		Kimono1	43.5	4.003	−0.156	40.0	3.276	−0.132	29.7	0.881	−0.034	58.0	2.776	−0.116
		ParkScene	42.1	1.835	−0.066	40.0	3.152	−0.126	40.5	1.310	−0.048	50.8	2.219	−0.085
	C	RaceHorsesC	44.0	3.832	−0.139	43.4	3.660	−0.164	29.9	3.214	−0.141	37.9	2.302	−0.079
		BasketballDrill	45.0	3.707	−0.136	39.0	3.787	−0.165	34.9	3.254	−0.137	47.0	2.791	−0.114
	D	Blowing.Bubbles	39.4	2.216	−0.086	36.1	2.095	−0.105	32.3	1.261	−0.048	42.6	3.098	−0.114
		BQSquare	41.9	0.795	−0.026	42.2	2.974	−0.118	43.0	2.499	−0.077	40.5	2.141	−0.075
		RaceHorses	40.8	3.727	−0.151	39.3	3.791	−0.165	31.5	1.997	−0.090	38.1	2.562	−0.100
	E	Johnny	42.2	1.538	−0.065	40.8	1.873	−0.077	58.3	1.481	−0.059	60.2	2.427	−0.078
		KristenAndSara	41.7	1.520	−0.076	50.6	2.368	−0.090	56.0	2.299	−0.110	61.6	1.598	−0.072
		Vidyo 1	41.6	1.582	−0.064	56.2	3.181	−0.130	56.6	1.631	−0.086	62.6	1.649	−0.068
		Vidyo 3	43.0	2.513	−0.112	44.7	2.843	−0.120	55.6	1.873	−0.098	61.1	1.716	−0.079
		Vidyo 4	43.4	4.984	−0.139	48.8	4.169	−0.103	52.0	3.242	−0.136	57.6	3.489	−0.127
	Average		**42.7**	**2.716**	**−0.103**	**43.5**	**3.013**	**−0.121**	**42.6**	**2.120**	**−0.090**	**50.1**	**2.369**	**−0.094**
Average of all sequences			42.5	2.819	−0.107	42.9	2.938	−0.120	42.2	2.346	−0.101	49.2	2.402	−0.096

4.5. Overall algorithm of the proposed transcoder

The overall algorithm of the proposed transcoding process has two sub-steps, on-line model training and predicting. After model training, the switcher in Fig. 5 is turned to the predicting stage to encode rest frames. The switcher can be turned to model training for model updating if necessary. Detailed steps of the model training and predicting are shown as follows.

Model Training:
Step (1): Encode yellow frames and collect ground truth CU depth data;
Step (2): Train SVM model over the ground truth CU depth and feature.
Predicting:
Step (1): Extract feature vectors of the current CU;
Step (2): Predict CU split flag with trained SVM mode parameters;
Step (3): Calculate decision probability according to decision value by Eqs. (6) and (7);
Step (4): If decision probability is smaller than threshold, encode the current CU and split the current CU into four sub-CUs, go to Step 7;
Step (5): If decision value is larger than 0, split the current CU into four sub-CUs, go to Step 7;
Step (6): Encode the current CU, go to Step 8;
Step (7): Go to Step 1for four sub-CUs;
Step (8): Go to Step 1 for next CU.

5. Experimental results and analyses

To testify the effectiveness of the proposed fast transcoding method, we implemented it on the platform an original cascaded transcoder consists of the JM 18.4 [27] and HM 14.0 [28]. 21 different test sequences with various resolutions and properties, from Class A to Class E [29], were utilized. And these test sequences are divided into two sets, Set 1 is the group of sequences which have been utilized in feature selection and model training, and Set 2 is the rest of test sequences. Firstly, they were encoded with H.264/AVC when QP was set as 28 to generate bit streams, *i.e.*, one bit stream for each sequence. Coding structure is IPPP. Then, the H.264/AVC bit streams were put into the fast H.264/AVC to HEVC transcoder, which transcoded the bit stream with four different QPs of 22, 27, 32 and 37 under HEVC common test conditions [29]. The HEVC transcoder was set as LP and random access (RA) configurations, respectively. Fast encoder decision and fast decision for Merge RD cost were enabled for HM 14.0. All the video transcoding experiments were performed on the computer with CPU AMD Athlon IIX2 B24, 2.99 GHz, 2 GB memory, Window XP operating system. BDPSNR and Bjønteggard Delta Bit Rate (BDBR) [26] were used to evaluate the RD performance of different schemes compared with the original cascaded transcoder. Additionally, processing time of transcoding was recorded, which includes time of decoding, model training, encoding and threshold updating. ΔT in Eq. (10) was also calculated to measure the complexity reduction where $\Psi \in$ {ICIP2012 [12], CSVT2014 [20], TMM2013 [30], Proposed_Adapt}. In the optimal threshold decision algorithm, n was set as 10.

5.1. Transcoding performance for three levels of CU depth prediction

There are three levels of CU decision from CU depth 0–3, which are D_0/D_1, D_1/D_2 and D_2/D_3. Thus, the coding performances of three-level of CU depth prediction are analyzed individually. Here, we denote the three levels of CU depth prediction by using classifier 0–2 as L_0 to L_2 respectively.

Table 4 presents the RD performance of three decision levels. It shows that 23.9% transcoding time is saved on average for L_0, in which only CU depth D_0/D_1 are predicted by the Classifier 0. Meanwhile, the average BDPSNR and BDBR values are −0.008 dB and

Table 7
Time consuming analysis for each part and complexity overheads in the proposed transcoder.

Sequence	QP	Time consuming (s)				Complexity of the overall transcoder (%)		
		T_{DEC}	T_{MT}	T_{ENC}	T_{TOT}	T_{DEC}/T_{TOT}	T_{MT}/T_{TOT}	T_{ENC}/T_{TOT}
BasketBallPass	24	0.61	0.43	280.68	281.72	0.22	0.15	99.63
(416 × 240)	28	0.45	0.35	231.77	232.56	0.19	0.15	99.66
	32	0.45	0.24	195.28	195.97	0.24	0.12	99.64
	36	0.45	0.29	164.72	165.45	0.26	0.17	99.57
BQMall	24	3.51	1.72	1152.36	1157.59	0.30	0.14	99.56
(832 × 480)	28	2.07	1.29	971.68	975.03	0.21	0.13	99.66
	32	2.02	1.20	855.56	858.78	0.23	0.14	99.63
	36	1.93	1.17	719.63	722.73	0.26	0.16	99.58
FourPeople	24	4.12	4.47	876.05	884.64	0.46	0.50	99.04
(1280 × 720)	28	2.96	3.50	811.75	818.20	0.36	0.42	99.22
	32	2.89	3.31	735.95	742.16	0.38	0.44	99.18
	36	2.82	3.11	618.21	624.14	0.45	0.49	99.06
BasketBallDirve	24	16.24	21.31	5769.87	5807.41	0.27	0.36	99.37
(1920 × 1080)	28	11.44	12.70	4512.14	4536.28	0.25	0.27	99.48
	32	10.54	11.15	3501.86	3523.55	0.29	0.31	99.40
	36	10.49	10.00	2892.06	2912.55	0.36	0.34	99.30
Traffic	24	23.20	111.14	9034.18	9168.52	0.24	1.21	98.55
(2560 × 1600)	28	23.30	52.09	6317.31	6392.70	0.36	0.81	98.83
	32	23.40	28.35	4728.50	4780.25	0.48	0.59	98.93
	36	23.20	20.73	3776.74	3820.67	0.60	0.54	98.86
Average						0.32	0.37	99.31

0.105%, respectively. The RD degradation is very small and negligible since the CU depth prediction accuracy is higher than 91% according to Fig. 8(a). For $L_0 + L_1$ where two levels of classifiers are applied, 41.3% transcoding time is saved, meanwhile, average BDPSNR and BDBR values are −0.036 dB and 0.796%, respectively. As one more level is optimized, more complexity reduction could be achieved at the cost of a little RD degradation. For $L_0 + L_1 + L_2$ in which all three levels are optimized, the complexity reduction is 50.2% on average. The BDPSNR and BDBR values are −0.085 dB and 1.986%, respectively. The three levels of CU depth prediction, $L_0 + L_1 + L_2$ has the lowest complexity and relative larger RD degradation. L_0 has negligible RD degradation and the complexity reduction is relatively small. L_0 to L_2 can be used individually or jointly, which can be flexibly activated according to user preference or system requirements. The complexity reductions in different settings are 23.9%, 41.3% and 50.2%, which are approaching the maximum potential complexities, 42.67%, 70.54% and 77.74%, as shown in Table 1. However, there is still a gap since the decoder and the encoder of encoding the training frames are not optimized; meanwhile, feature extraction, and on-line model training will bring a few additional complexity overheads. Fig. 10 shows the CU size prediction with the $L_0 + L_1 + L_2$ optimization, where the grid is the predicted CU size. The blue rectangles are correctly predicted CU and the red rectangles are falsely predicted CU. We can observe that the size of most CU can be correctly predicted by the proposed H.264/AVC to HEVC transcoder.

5.2. Transcoding performance comparison with the state-of-the-art benchmarks

Tables 5 and 6 show the complexity and RD performance comparisons with three state-of-the-art transcoding schemes, including ICIP2012 [12], CSVT2014 [20] and a fast CU depth decision algorithm in HEVC cascaded with JM decoder, denoted as "JM Decoder + TMM2013 [30]". Compared with the original cascaded transcoder that consists of the original H.264/AVC decoder and the original HEVC encoder, ICIP2012 under LP configuration can reduce complexity from 35.1% to 43.2%, 39.7% on average. Meanwhile, BDPSNR is from −0.049 dB to −0.315 dB, −0.134 dB on average. And the BDBR increases from 1.335% to 8.104%, 3.158% on average. For RA configuration, ICIP2012 can reduce complexity 42.5% on average with BDPSNR degradation as −0.107 dB. Though significant complexity reduction is achieved, the RD degradation is large. The CSVT2014 under LP configuration reduces the complexity from 39.4% to 54.7%, 46.3% on average. The BDPSNR degradation is from −0.054 dB to −0.176 dB, −0.096 dB on average. The BDBR increases from 1.308% to 4.013%, 2.259% on average. For RA configuration, CSVT2014 reduces complexity 42.9% on average while BDBR increase 2.938%. It achieves more complexity reduction and has better RD performance when compared with ICIP2012. As for the "JM Decoder + TMM2013" under LP configuration, it reduces complexity from 15.2% to 52.1%, which is 32.7% on average. Meanwhile, the BDBR and BDPSNR are 2.736% and −0.116 dB on average, respectively. For RA configuration, "JM Decoder + TMM2013" can reduce complexity 42.2% on average, which is more than that of the LP configuration.

As illustrated in Tables 5 and 6, the Proposed_Adapt reduces complexity 50.2% and 49.2% on average, meanwhile the average BDPSNR are −0.085 dB and −0.096 dB under LP and RA configurations, respectively. The proposed scheme achieves more complexity reduction and less RD degradation when compared with the benchmarks, which has proved the effectiveness of the proposed scheme. For some special cases, such as "PeopleOnStreet" and "BlowingBubbles", the proposed method is not superior when compared with CSVT2014 because of the initial probability threshold and the constant weighting factor ω. However, the proposed algorithm is the best in terms of complexity reduction and RD performance for most of sequences.

Besides the RD performance and the overall complexity, we also analyze the coding complexity of each part of the Proposed_Adapt algorithm, as shown in Table 7. The Proposed_Adapt consists of three parts, including H.264/AVC decoder (DEC), SVM model training plus threshold calculation (MT), and the HEVC encoder (ENC). The processing time of each part is denoted as T_{DEC}, T_{MT}, and T_{ENC}, Let T_{TOT} be the overall processing time of the transcoder. In the transcoder, we can observe that the T_{DEC}/T_{TOT} and T_{MT}/T_{TOT} are only 0.32% and 0.37% on average, and T_{ENC}/T_{TOT} is major part of the complexity, which has 99.31%. Therefore, the overhead complexity of the model training plus threshold calculation is little and negligible compared to the HEVC encoder. Because the training data (features

and ground truths) are extracted from the first four frames of each sequence, the sequences with large resolution would have more training data than the sequences with small resolution. More training data means higher on-line training complexity. Therefore, the time consuming of T_{MT} is a little more for the sequences with large resolution. But the value of T_{MT}/T_{TOT} is very small for all the sequences. The proposed transcoder focuses on the complexity reduction of T_{ENC}, which can reach about 50%. Although it is not real time, the complexity reduction is beneficial to real time applications of the H.264/AVC to HEVC transcoding. Details of the source code, trained model and testing results can be referred in the website.[1]

6. Conclusions

In this paper, we exploit the block partition similarity between coding standards and present a fast H.264/AVC to HEVC transcoding method based on SVM, so as to reduce transcoding complexity. In the proposed method, reprehensive SVM features are selected by the proposed feature selection algorithm, and the prediction accuracy of the CU depth prediction is improved by the proposed adaptive parameter determination algorithm. Extensive experiments reveal that the proposed fast transcoding method can reduce 50.2% and 49.2% complexity on average under LP and RA configurations, respectively, while the rate-distortion degradation is negligible. The proposed scheme, which is more efficient and outperforms the state-of-the-art benchmarks, will be useful for high-quality transcoding applications and services.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grants 61471348, U1301257 and 61272289, in part by Shenzhen Overseas High-Caliber Personnel Innovation and Entrepreneurship Project under Grant KQCX20140520154115027, and in part by Guangdong Special Support Program for Youth Science and Technology Innovation Talents under Grant 2014TQ01X345, in part by the National High-tech R&D Program of China under Grant 2015AA015901, and by Zhejiang Provincial Natural Science Foundation of China under Grant LY15F010005.

References

[1] I. Ahmad, X. Wei, Y. Sun, Y. Zhang, Video transcoding: an overview of various technique and research issues, IEEE Trans. Multimedia 7 (5) (2005) 793–804.
[2] J.D. Cock, S. Notebaert, P. Lambert, R.V. Walle, Motion-refined rewriting of H.264/AVC-coded video to SVC streams, J. Vis. Commun. Image R 22 (2011) 391–400.
[3] H. Kalva, The H.264 video coding standard, IEEE Multimedia 13 (4) (2006) 86–90.
[4] G. Sullivan, J. Ohm, W. Han, T. Wiegand, Overview of the High Efficiency Video Coding (HEVC) standard, IEEE Trans. Circ. Syst. Video Technol. 22 (12) (2012) 1649–1668.
[5] F. Zhang, E. Steinbach, P. Zhang, MDVQM: a novel multidimensional no-reference video quality metric for video transcoding, J. Vis. Commun. Image R 25 (2014) 542–554.
[6] Y. Liu, S. Ci, H. Tang, Y. Ye, J. Liu, QoE-oriented 3D video transcoding for mobile streaming, ACM Trans. Multimedia Comput. Commun. Appl. 8 (3s) (2012). Article 42:1-20.
[7] X. Liu, K.Y. Yoo, Fast interframe mode decision algorithm based on mode mapping and MB activity for MPEG-2 to H.264/AVC transcoding, J. Vis. Commun. Image R 21 (2010) 155–166.
[8] H. Shu, L.P. Chau, Intra/inter macroblock mode decision for error-resilient transcoding, IEEE Trans. Multimedia 10 (1) (2008) 97–104.
[9] S. Kim, J. Han, J. Kim, Efficient motion estimation algorithm for MEPG-2 to H.264 transcoder, in: Proc. IEEE Int'l Conf. Image Process., Genoa, Italy, 2005. pp. III-656-9.
[10] K. Fung, W. Siu, Low complexity H.263 to H.264 video transcoding using motion vector decomposition, Proc. IEEE Int'l Symp. Circuits Syst., vol. 2, 2005, pp. 908–911.
[11] B. Petljanski, H. Kalva, DCT domain intra MB mode decision for MPEG-2 to H.264 transcoding, in: Proc. IEEE Int'l Conf. Consumer Electron., Las Vegas, NV, USA, 2006, pp. 419–420.
[12] E. Peixoto, E. Izquierdo, A complexity-scalable transcoder from H.264/AVC to the new HEVC codec, in: Proc. IEEE Int. Conf. Image Process., Orlando, FL, USA, 2012, pp. 737–740.
[13] D. Zhang, B. Li, J. Xu, H. Li, Fast transcoding from H.264/AVC to high efficiency video coding, in: Proc. IEEE Int'l Conf. Multimedia Expo., Melbourne, VIC, 2012, pp. 651–656.
[14] T. Shen, Y. Lu, Z. Wen, L. Zou, Y. Chen, J. Wen, Ultra-fast H.264/AVC to HEVC transcoder, in: Data Compress. Conf., Snowbird, UT, March 20–22, 2013, pp. 241–250.
[15] W. Jiang, Y.W. Chen, Low-complexity transcoding from H.264 to HEVC based on motion vector clustering, Electron. Lett. 49 (19) (2013) 1224–1226.
[16] Y. Chen, Z. Wen, J. Wen, M. Tang, P. Tao, Efficient software H.264/AVC to HEVC transcoding on distributed multi-core processors, IEEE Trans. Circ. Syst. Video Technol. 25 (8) (2015) 1423–1434.
[17] G. Fernandez-Escribano, H. Kalva, J.L. Martinez, P. Cuenca, L. Orozco-Barbosa, A. Garrido, An MPEG-2 to H.264 video transcoder in the baseline profile, IEEE Trans. Circ. Syst. Video Technol. 20 (5) (2010) 763–768.
[18] C. Chiang, W. Pan, S. Zhuang, S. Lai, Fast H.264 encoding based on statistical learning, IEEE Trans. Circ. Syst. Video Technol. 21 (9) (2011) 1304–1315.
[19] T. Shanableh, E. Peixoto, E. Izquierdo, MPEG-2 to HEVC video transcoding with content-based modeling, IEEE Trans. Circ. Syst. Video Technol. 23 (7) (2013) 1191–1196.
[20] E. Peixoto, T. Shanableh, E. Izquierdo, H.264/AVC to HEVC video transcoder based on dynamic thresholding and content modeling, IEEE Trans. Circ. Syst. Video Technol. 24 (1) (2014) 99–112.
[21] X. Shen, L. Yu, CU splitting early termination based on weighted SVM, EURASIP J. Image Video Process. 4 (2013) 1–11.
[22] J. Xiong, H. Li, F. Meng, S. Zhu, Q. Wu, B. Zeng, MRF-based fast HEVC inter CU decision with the variance of absolute differences, IEEE Trans. Multimedia 16 (8) (2014) 2141–2153.
[23] Y. Zhang, S. Kwong, X. Wang, H. Yuan, Z. Pan, L. Xu, Machine learning based coding unit depth decisions for flexible complexity allocation in high efficiency video coding, IEEE Trans. Image Process. 24 (7) (2015) 2225–2238.
[24] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machine, ACM Trans. Intell. Syst. Technol. 2 (3) (2011) 1–27.
[25] V. Franc, A. Zien, B. Scholkopf, Support vector machines as probabilistic models, in: Proc. IEEE Int'l. Conf. Machine Learning, Bellevue, WA, USA, 2011.
[26] G. Bjøntegaard, Calculation of Average PSNR Differences between RD-Curves, ITU-T Video Coding Experts Group (VCEG), doc. M33, Austin, TX, 2001.
[27] ITU-T, Joint Model (JM), H.264/AVC Reference Software, JM 18.6. <http://iphome.hhi.de/suehring/tml/download/>.
[28] K. McCann, B. Bross, W.-J. Han, I.K. Kim, K. Sugimoto, G.J. Sullivan, High Efficiency Video Coding (HEVC) Test Model 14 (HM 14) Encoder Description, JCT-VC of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Tech. Rep. Doc. JCTVC-P1002, San Jose, CA, January 9–17, 2014. <http://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-14.0/>.
[29] F. Bossen, Common Test Conditions and Software Reference Configurations, JCTVC of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Tech. Rep. Doc. JCTVC-J1100, Stockholm, SE, July 11–20, 2012.
[30] L. Shen, Z. Liu, X. Zhang, W. Zhao, Z. Zhang, An effective CU size decision method for HEVC encoders, IEEE Trans. Multimedia 15 (2) (2013) 465–470.

[1] http://codec.siat.ac.cn/downloadcenter/machine-learning-based-fast-transcoding.rar.