Machine Learning Based VP9-to-HEVC Video Transcoding

Xiangyu Li [1], Rong Xie [1,2], Li Song [1,2], Liang Zhang [1,2]
[1] Institute of Image Communication and Network Engineering, Shanghai Jiao Tong University
[2] Cooperative Medianet Innovation Center,
Shanghai, China
{dianmu, xierong, song_li}@sjtu.edu.cn, liang.zhang@ieee.org

***Abstract*—As more and more coding standards are developed for videos with higher resolution and higher frame rates, the conversion between them is in demand. This paper focuses on the development of fast VP9-to-HEVC transcoding algorithm. Since the decoding-and-full-re-encoding process is time consuming, we explore Naïve-Bayes based machine learning algorithm to accelerate VP9-to-HEVC transcoding. The machine learning process can bridge VP9 decoding information and HEVC splitting decision throughout training results, so the HEVC CU searching process can be simplified for faster transcoding. Two new features closely related to VP9, sub-block counting and depth map, are added for VP9-to-HEVC video transcoder. Using coding features and Naïve-Bayes algorithm, several models are built for different QPs and block sizes to reduce transcoding time. Experiments show that a maximum time reduction of 53% and an average reduction of 44% can be achieved and the loss of decoded image quality in BD-rate is almost ignorable.**

Keywords—VP9; HEVC; video transcoding; machine learning algorithm

I. INTRODUCTION

Videos with higher resolution and higher frame rate gradually become the preferences in our daily life. Behind it, the development in semiconductor and communication systems have enlarged the storage capacity and network bandwidth for multimedia transmission and delivery. As people pursuit for better viewing experiences, different organizations come up with new coding standards and codecs to fulfill requirements for better compression efficiencies, two of which are HEVC and VP9. Also, the wide applications of videos compressed by different standards brings the needs of transcoding between them.

HEVC is formally called ITU-T H.265 [1] or ISO/IEC 23008-2 [2], which is designed by ISO/IEC MPEG and ITU-T VCEG groups. As the recent successor of the mainstream H.264/AVC [3], HEVC is the nowadays popular compression standard. It has been applied by multiple media services as it is better capable of dealing with high-resolution videos such as 2K or 4K video. Compared to H.264/AVC, HEVC can reduce the bitrate by about 50% while maintaining video quality similar to H.264/AVC. The cost for better compression efficiency is the increment in computationally complexity due to flexible block partitions and more precise prediction.

VP9 [4], on the other hand, is the successor of VP8 [5]. It is an open source royalty-free codec developed by Google who joins in the competition for standards making. Thanks to storage capacity and network bandwidth improvement, high-resolution videos occupy a larger portion of web contents for better viewing experience. VP9 is mainly designed for these web video applications and scenarios. Due to its good performance and royalty free characteristics, VP9 is now supported by multiple browsers with applications for practical consumer use. As for compression algorithms, VP9 shares with HEVC a similar hybrid coding framework. Under this framework, VP9 uses techniques and features of its own to compose each coding module to maintain the compression efficiency as well as the video quality. Nowadays VP9 has been integrated in multiple Google products, and is widely used in video-sharing websites like Youtube. With its continuous upgrading and promotion, even facing to the next generation, we can expect more and more web video contents being encoded by VP9. There will be needs to exploit transcoding algorithms where VP9 is involved.

Transcoding between HEVC and VP9 is of great importance as more and more competing codecs are developed and applied in the market. A VP9-to-HEVC transcoder is able to solve the rising adaptability issue where the original video content is encoded by VP9 and the playback environment is HEVC.

Most previous transcoding works are done on H.264-to-HEVC transcoder. Since block partitions in HEVC are preserved using quad-tree coding structure, the encoding process needs to recursively check each coding unit (CU) partition including calculation of rate-distortion costs and final decision-making. This process is time consuming. To accelerate the H.264-to-HEVC transcoding process, several studies have focused on simplifying this CU partition searching process. Peixoto and Izquierdo [6] first proposed an algorithm to simplify CU size decision process in HEVC by re-utilizing motion vector (MV) information derived from the H.264/AVC decoder. Two improved mapping scheme related to machine learning algorithms (one offline training and the other online training) were proposed in [7] for better fitting the macro-blocks (MBs) from H.264/AVC from H.264/AVC to HEVC CUs. Peixoto et al. [8] further proposed a two-step CU splitting way in which MV variance distance was first compared to

thresholds (low and high) to decide whether to split a CU. A linear discriminant functions (LDF) method was then introduced to decide whether the not-splitting CU should be further partitioned or not. Diaz-Honrubia et al. [9] proposed a fast quad-tree level decision (FQLD) method, which uses Naïve-Bayes based machine learning algorithms to make the CU partition decisions. An improved adaptive fast quad-tree level decision (AFQLD) algorithm [10] was proposed by the same author and achieves maximum 2.52× times speed-up with 4.52% BD-rate penalty. Dongdong Zhang [11] proposed a fast CU partition algorithm based on Fisher Discriminant Analysis, in which the author used online machine learning strategy to update weight vectors as well as thresholds for training sets to ensure the accuracy of trained models, and the algorithm obtains average 1.92× speed-up with 2.75% BD-rate penalty. Besides those H.264-to-HEVC transcoders, to the best of our knowledge, only one research has explored the transcoding from HEVC to VP9, which is conducted by Enrique and Rafael [12]. The algorithm took into account the mechanism similarity of HEVC and VP9 and replaced some of Inter predictions within blocks with Intra predictions according to the information of the corresponding HEVC blocks. Furthermore, it simplified the reference frame selecting process within VP9, forced VP9 encoder to utilize the reference frame that HEVC already uses, and discarded further searching.

Based on above mentioned contributions and studies on H.264-to-HEVC transcoders, we propose a machine learning based VP9-to-HEVC video transcoder. Naïve-Bayes based machine learning algorithm will bridge VP9 decoding information and HEVC splitting decision throughout training results and simplifies HEVC CU searching process. Two new features closely related to VP9, sub-block counting and depth map, are additionally proposed for VP9-to-HEVC video transcoder. Using coding features and Naïve-Bayes algorithm, several models are built for different QPs and block sizes to reduce transcoding time.

The rest of this paper is organized as follows. Section II briefly reviews the VP9-to-HEVC transcoding framework. Section III presents VP9 and HEVC feature selection. Section IV describes model training and transcoding. Section V presents the experimental results and analysis. The final conclusion is in Section VI.

II. THE PROPOSED TRANSCODER

A trivial VP9-to-HEVC transcoder is simply composed of a VP9 decoder followed by a HEVC encoder, without feature extraction and data processing in between. For such trivial transcoder, the full re-encoding process is computationally complex and time consuming. To accelerate this transcoding process, a common idea is to filter and select certain effective features from VP9 decoder, process it, then re-utilize processed features to simplify the HEVC re-encoding process in a reasonable manner. The extracted features can help to bridge the decoder and encoder to save the VP9-to-HEVC transcoding time.

The RDO process in HEVC tends to recursively check all possible CU sizes from 64×64 down to 8×8. This is a computationally complex and time consuming process. So the

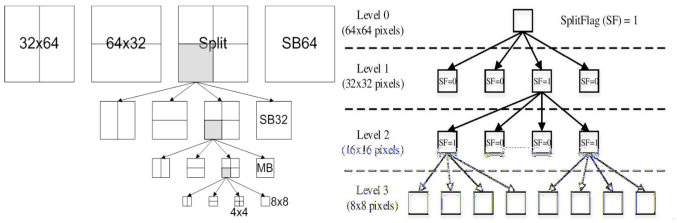

Fig. 1. VP9 splitting structure (left) and HEVC partitioning structure (right)

presented VP9-to-HEVC transcoder aims to early terminate this searching process to skip certain recursions at different depth levels. The pattern between VP9 extracted information and HEVC encoding decisions are learned by data driven mechanism using Naïve-Bayes based machine learning algorithms. Before the transcoding, several models are built by offline training to deal with different QP-values and different block sizes. The models select effective features and show the learned patterns in a probabilistic way. Then the models are applied to instruct the transcoding process. In the training and transcoding process, two new features, sub-block counting and depth map, closely related to VP9 are proposed for better transcoding performance. The detailed feature selection and speedup process is described below.

III. VP9 AND HEVC FEATURE SELECTION

The proposed VP9 and HEVC feature section will fully exploit high similarity of coding unit splitting structure between VP9 and HEVC encoding process. Different from H.264 splitting structure, VP9 provides with a more specific and complex quad-tree structure, which is similar to that of HEVC, as shown in Fig. 1. HEVC splits 64×64 coding unit down to 8×8 blocks, whereas VP9 divides large coding units of 64×64 all the way down into 4×4 blocks. Therefore, VP9 provides us with more information and new possibilities in feature selection for information reusing. Other than traditional MV information and costs information, two new features, sub-block counting and depth map, are additionally collected and trained based on this VP9 splitting structure.

A. Motion Vector related features

Traditional features like VP9 motion vectors information are collected for reference. As motion vectors carries the block moving information within frames, they are of high value to be considered as features and fed into the training process to build the models. In VP9 Inter Prediction, an inter block can be predicted using either a single reference frame or a combination of two reference frames, the latter of which is called compound prediction where two motion vectors and two reference frames are specified for an inter block. In the case of compound prediction, the prediction is first formed from each reference frame, and then the final prediction is produced as an averaged combination of these two [13].

As video sequences may be composed of pictures with static and motion objects, motion vectors could be long or short and point to different reference pictures. A pre-processor, such as scaling to the nearest reference picture, may be required to deal with multiple-reference-picture scenarios before they are fed into the training process. In this case, we first scale the

MVs down to the same reference frame according to the distance from current frame to the corresponding reference frames. Then, the MVs are averaged into one MV, and its data is collected as features during the VP9 decoding process.

The depth information of VP9 coding unit splitting structure is extracted together with motion vector data of its x- and y-direction value. The depth information is to suggest the size of PU that contains this motion vector. Note that during the process there will be not only Inter-coded blocks but also Intra-coded blocks in a VP9 frame. To deal with these Intra-coded blocks, we mark the values of motion vector to be zero while still preserving the depth value. This helps us to re-create the block structure with motion vectors in an easier way, since zero (represent Intra-coded) MVs will not influence the overall value.

After collecting the MV information, we re-organize the information in two forms before they are sent to train the models. First, the Z-scan ordered continuous-preserved values are separated and then merged into unit of two block sizes, 64×64 and 32×32. Within each unit size, we then calculate several features from the original MV value. In this paper, altogether six features are calculated and formed. They are: the sum of MV values within the unit size; the average MV value of x-direction and y-direction separately; the variance value of x-direction and y-direction separately; the variance value of all MVs in the block. These six features are regarded as motion vector related features which are collected in the VP9 decoding process.

B. *HEVC coding modes features*

HEVC encoding process recursively split and check the coding units. Two costs are calculated before other partition modes are checked, they are 2N×2N cost and SKIP cost. These two Lagrangian costs give us partial information for references. In this manner, the costs of 2N×2N mode and SKIP mode are chosen to train transcoding probabilistic models by Naïve-Bayes based machine learning algorithm. Similar to the motion vector processing steps, the costs of 2N×2N mode and SKIP mode are also collected in terms of block sizes (64×64 size and 32×32 size), so that we are able to train difference models for different block sizes.

C. *New VP9 coding features*

The first new coding feature related to the coding structure is the sub-block counting, which calculates the number of sub-blocks within each 64×64 Superblock in VP9, as shown in Fig. 2 (left). In this example, altogether 22 sub-blocks are counted in Zigzag order. Both HEVC and VP9 tend to split CUs into smaller sizes where the details of a picture are presented. If the sub-block count is large in VP9, HEVC will also tend to split the CU into smaller coding units. Like all other features, the sub-block counting data is also extracted in terms of two block sizes (64×64 and 32×32). Each block size records the sum of sub-block counting lower than or equal to that block size. For example, in Fig. 2 (left), there are four 32×32 blocks, and for each 32×32 block, the sub-block counting is {10,4,7,1} respectively in a zig-zip order. Similarly, in Fig. 2 (left), there is one 64×64 block, and the sub-block counting is 22 which is the sum of four 32×32 blocks' counts. For a 64×64 Superblock that is not split, the sub-block counts for all 32×32 blocks are set to {0,0,0,0}.

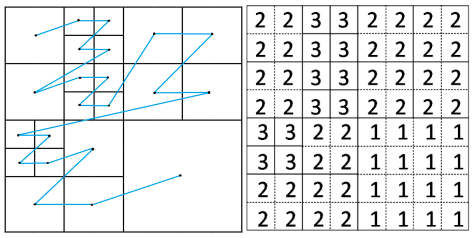

Fig. 2. VP9 sub-block count (a 64×64 Superblock splitted down to 8×8) (left) and depth map (right)

The second feature is the depth map. Different from previous H.264/AVC to HEVC transcoder, this Quad-tree structure feature is recorded using depth map to preserve the VP9 splitting decisions for each Superblock. The depth map information preserved, as shown in Fig. 2 (right), is extracted in unit of 8×8, which merges four 4×4 blocks together since HEVC has no 4×4 CUs. Note that RDO mechanism and the Inter/Intra prediction thresholds used within VP9 are different from those of HEVC. Directly copying the resulted VP9 splitting structure to HEVC is not logically appreciated and might cause large distortion. To this end, a weight from the offline machine learning training process is assigned to the depth level as a factor to influence the HEVC CU splitting decisions at two levels (64×64 and 32×32). In the HEVC re-encoding process, the depth at the same positon is collected from corresponding VP9 depth map and the weight for this depth is given by the trained model. In this way, the depth map helps to HEVC CU splitting decisions. Besides, a simplified and easier way to use depth map is to only focus on the maximum depth within each 64×64 Superblock or 32×32 sized blocks. This maximum depth is also collected as a feature. The weight parameter is the result of the offline machine learning training process.

All ten features extracted or calculated for the model training and transcoding are summarized as follows:

1. motion vector related features
   - sum of MV values
   - average MV value of x-direction
   - average MV value of y-direction
   - variance MV value of x-direction
   - variance MV value of y-direction
   - variance of all MVs within
2. HEVC encoding features
   - 2N×2N mode cost
   - SKIP mode cost
3. VP9 new features
   - sub-block counting
   - depth map or its simplified version

Other than the depth map, the rest features are split into 64×64 size and 32×32 size for model training and transcoding.

IV. Model Training and Transcoding

After the coding features are extracted from VP9 decoding information, the task is to utilize the information to build the

probabilistic models. To this end, the Naïve-Bayes based machine learning algorithm will be briefly reviewed, the models building process and transcoding strategy are then explained.

A. Naïve-Bayes based machine learning algorithm

In the offline learning process, the data driven machine learning algorithm, which is used to find the patterns between VP9 coding features and HEVC CU splitting decisions, is based on Naïve-Bayes theories. Suppose in the classification problem that we have gathered N features with their values being $\{v_1, v_2, \dots, v_N\}$ and the possible result set is $\{R_1, \dots, R_K\}$. By calculating the conditional probability $P(R|v_1, v_2, \dots, v_N)$, we choose the one with maximum probability to be the final result r, which is formulated as follows:

$$r = argmax\ P(R|v_1, v_2, \dots, v_N) \quad where\ R \in \{R_1, \dots, R_K\} \tag{1}$$

Based on the Bayes theory, the formula above is re-written as

$$r = argmax \frac{P(v_1, v_2, \dots, v_N|R)P(R)}{P(v_1, v_2, \dots, v_N)} \tag{2}$$

where $P(v_1, v_2, \dots, v_N)$ is independent of the result set and can be ignorable from the formula. Then, the formula is re-written as

$$r = argmax\ P(v_1, v_2, \dots, v_N|R)P(R) \quad where\ R \in \{R_1, \dots, R_K\} \tag{3}$$

here $P(v_1, v_2, \dots, v_N|R)$ is the joint conditional probability to be computed. Calculating this item is a hard work due to data amount involved (exponential) that is too large to be dealt with. Further simplification will be done through utilizing the independency character. As assumed often in Naïve-Bayes classifiers, the feature involved in the algorithm should be conditionally independent. The formula is then further simplified as

$$r = argmax\ P(R) \prod_{i=1}^{N} P(v_i|R) \quad R \in \{R_1, \dots, R_K\} \tag{4}$$

where the probability $P(R)$ is computed as

$$P(R_i) = \frac{freq(R_i)}{N} \tag{5}$$

Since the feature value v_i is a discrete variable, the conditional probability, $P(v_i|R)$, is equal to frequency statistics from the collected data. Here, $K = 2$ since the result set $\boldsymbol{R}$ only includes two variables $\{R_{split}, R_{not\ split}\}$, representing HEVC encoding split or not-split decisions.

The Naïve-Bayes classifier [14] [15] is a basic and simple one in compared to many other probabilistic solutions. Due to its simplicity, the data mining process does not need complex computation. Its practical performance shows a competitive result with respect to more complex ML algorithms [10]. Furthermore, it is efficient for scarce data to ensure the effectiveness of the trained models and also robust against over-fitting problem because of the low-ordered parameters [10] it requires.

TABLE I. HIT-RATE OF TRAINING VIDEOS AT DEPTH 0

Sequences	Hit Rate (%) at Depth 0				
	22	27	32	37	Average
PeopleOnStreet	93.12	91.02	86.91	84.43	88.87
ParkScene	87.70	88.28	87.36	83.00	86.59
PartyScene	90.67	89.87	86.29	79.25	86.52
BQSquare	90.11	90.75	87.60	85.24	88.43
Average	90.40	89.98	87.04	82.98	87.6

TABLE II. HIT-RATE OF TRAINING VIDEOS AT DEPTH 1

Sequences	Hit Rate (%) at Depth 1				
	22	27	32	37	Average
PeopleOnStreet	84.22	82.99	82.22	79.58	82.25
ParkScene	84.49	84.13	82.78	80.33	82.93
PartyScene	86.78	88.96	80.35	75.88	82.99
BQSquare	89.31	87.20	80.29	75.41	83.05
Average	86.20	85.82	81.41	77.80	82.81

B. Training sequences and models building

The features and HEVC decisions are fed into the classifier to generate corresponding models where data mining is applied. Note that such supervised model learning process is implemented offline, while the result is used in the transcoding acceleration.

Four sequences with different resolutions and motion characteristics are chosen from Class A, B, C, and D for model training. They are

- Class A: PeopleOnStreet, 2560×1600.
- Class B: ParkScene, 1920×1080.
- Class C: PartyScene, 832×480.
- Class D: BQSquare, 416×240.

Since the transcoding will possibly change the CU splitting structure in HEVC re-encoding process, the hit rate is used to present the accuracy of the classification and defined as the percentage of correctly classified splitting over total ground-truth splitting. Table I and Table II show the achieved hit rates of these four training video sequences at two different depths by the ML algorithms as mentioned in Sub-Section IV-A. For depth 0 (64×64 level) and depth 1 (32×32 level), the hit-rate of four QP-values {22, 27, 32, 37} are listed for reference.

The trained models used in actual transcoding process are generated from these four sequences. Our goal is to early-terminate the RD searching process, more specifically, to early-decide whether to further split the current CU at each CU level (64×64 level and 32×32 level). To this end, both depth-level and QP-value need to be considered. One way is for each depth to build four models corresponding to four QP values. In this way, total eight models of two different depths are trained in the offline learning process. However, by observing the average training accuracy in Tables I and II, we can find that data of QP-22 and QP-27 are above the average, while QP-32 and QP-37 are below the average. Therefore, we choose QP-30

TABLE III. HIT-RATE OF DIFFERENT TRAINED MODELS

Model I	Model II	Model III	Model IV
90.19	85.01	86.01	79.60

as the threshold for merging the models. The final defined four models are listed below

- Model I: Depth 0, QP<30 (merge QP 22 and 27).
- Model II: Depth 0, QP>30 (merge QP 32 and 37).
- Model III: Depth 1, QP<30 (merge QP 22 and 27).
- Model IV: Depth 1, QP>30 (merge QP 32 and 37).

The hit rate (training accuracy) of four models are presented in Table III. From Tables I – III, we can find that the ML algorithm is more accurate for smaller QP and larger block size.

C. Early termination

The trained models are presented in a probabilistic form, which gives the value intervals of a chosen effective feature and the probability of classification decisions (split or not split) corresponding to its value interval. In the transcoding stage, the value of feature is obtained from VP9 decoder. Based on the feature intervals, we have the probability of classification decisions from the trained models to make a decision whether or not to further split the current CU.

The model-generated thresholds are used for early-termination in the transcoding stage. First, costs for SKIP/MERGE and 2N×2N mode are collected as we mentioned in Sub-Section III-B. Then the early-termination decision is made based on the model we built. According to the built model, if the decision is split, the search for this CU depth is early-ended, discarding all remaining unchecked possible modes (N×N, 2N×N, N×2N etc.). Then the search directly skips to the next depth, where CU is split into 4 subparts if allowed. Otherwise, if the decision is not split, then at this level we check all remaining unchecked possible modes, stop splitting the CU into 4 subparts and search for this CU is ended. In this case, all subparts recursions and estimations calculations are discarded. Combine the above two early termination strategy we can speed up the HEVC re-encoding process.

V. EXPERIMENRT RESULTS

The experiments are conducted to evaluate the performance of the proposed VP9-to-HEVC transcoder. The anchor is the VP9-to-HEVC full re-encoding transcoder with corresponding software being libvpx-v1.5.0 for VP9 and HM 16.0 for HEVC. The experiments condition includes a four-core CPU running at 2.7 GHz and 8 GB main memory. Six sequences with different characteristics are tested at different QPs {22,27,32,37} with low-delay P configuration used in HEVC.

- Class B: BasketballDrive, BQTerrace.
- Class C: BQMall, RaceHorsesC.
- Class D: BlowingBubbles, RaceHorsesD.

The performance of the transcoder is measured by speed-up, time reduction, and BD-rate measure [16] which are defined as

$$Speed\text{-}up = \frac{Time_{anchor}}{Time_{proposed}} \quad (6)$$

$$Time\text{-}Saving(\%) = \frac{Time_{anchor}\text{-}Time_{proposed}}{Time_{anchor}} \times 100\% \quad (7)$$

$$BD\text{-}rate(\%) = \frac{4\times BD_y + BD_u + BD_v}{6} \quad (8)$$

The experimental results of the proposed VP9-to-HEVC transcoder are shown in Table IV in terms of BD-rate and speed-up. Note that the time-saving performance is equal to the speed-up.

A. Performace Analysis

Table IV shows the performances of the proposed transcoding algorithm for Depth 0, where the algorithm is only applied at Depth 0 (64×64 sized CUs), and Depth 1, where only 32×32 sized CUs is accelerated, as well as Depth 0&1, which includes both 64×64 and 32×32 sized CUs.

Analyzing the BD-rate penalty data from Table IV yields that early-termination process is mainly functional in 32×32 sized CU. For each sequence, the BD-rate penalty varies from a minimum 1.2% for *RaceHorsesD* to a maximum 6.3% for BQTerrace. An averaged BD-rate penalty for six sequences is 2.8% for the Depth 0&1. The averaged BD-rate performance of only 0.3% is pretty good in Depth 0, while the number drops to 2.6% for the Depth 1.

The proposed method achieves a great time-saving performance. Table IV shows that a minimum of 35% is achieved for *RaceHorsesD* while a maximum time reduction of 53% is achieved for sequence *BasketballDrive*. An average time reduction of 18% can be reached in Depth 0, 34% in Depth 1, and 44% in Depth 0&1. These data also suggest that the proposed algorithm is especially functional for 32×32 sized CU.

Table V and VI show the hit-rate, which is defined in Sub-Section IV-B, of these six testing sequences with Depth 0 and Depth 0&1 condition. The hit-rate for different QP-values are listed for reference. From the tables we can see that, on one hand, the overall hit-rate at Depth 0 is higher than Depth 0&1, on the other hand, lower QP-value tends to acquire higher hit-rate. A minimum hit-rate is found to be 77.41% at QP-37 Depth 0&1 and the maximum hit-rate is found to be 92.19% at QP-22 Depth 0. The overall hit-rate for all cases is about 84.5%.

TABLE IV. EXPERIMENTAL RESULTS FOR PROPOSED TRANSCODER

		BD-Rate (%)												Speed Up		
		Depth 0				Depth 1				Depth 0&1				D0	D1	D0&1
		Y	U	V	YUV	Y	U	V	YUV	Y	U	V	YUV			
Class B	BQTerrace	0.5	0.1	0.2	0.4	7.0	2.1	2.5	5.4	8.4	2.1	2.2	6.3	1.24	1.59	1.89
	BasketballDrive	0.3	0.1	0.3	0.3	5.7	0.9	1.3	4.2	4.9	3.1	2.3	4.2	1.36	1.66	2.14
Class C	BQMall	0.9	0.0	0.3	0.7	2.8	0.4	1.3	2.2	3.5	0.7	0.0	2.5	1.19	1.47	1.69
	RaceHorsesC	0.1	-0.1	0.2	0.0	0.8	0.7	0.9	0.8	1.6	0.2	0.2	1.5	1.22	1.43	1.71
Class D	RaceHorsesD	0.5	0.2	0.0	0.4	1.7	0.1	0.1	1.2	1.7	0.2	0.3	1.2	1.17	1.44	1.55
	BlowingBubbles	0.1	0.2	0.0	0.1	1.9	1.2	0.9	1.6	2.1	0.7	-0.3	1.5	1.14	1.49	1.63
Average		0.4	0.1	0.2	0.3	3.3	0.9	1.2	2.6	3.7	1.2	0.8	2.8	1.22	1.51	1.77

TABLE V. HIT RATE OF TESTING VIDEOS AT DEPTH 0

	Hit-Rate (%) at Depth 0				
	22	**27**	**32**	**37**	**Average**
BQTerrace	89.20	89.4	84.35	86.13	87.27
BasketballDrive	91.03	87.9	88.56	85.90	88.35
BQMall	92.19	82.45	83.69	84.66	85.75
RaceHorsesC	90.11	90.75	87.27	87.11	88.81
RaceHorsesD	88.92	90.29	84.78	82.96	86.74
BlowingBubbles	89.77	86.99	85.35	91.91	88.51
Average	90.20	87.96	85.67	86.45	87.57

TABLE VI. HIT RATE OF TESTING VIDEOS AT DEPTH 0&1

	Hit-Rate (%) at Depth 0&1				
	22	**27**	**32**	**37**	**Average**
BQTerrace	81.02	83.56	83.14	80.27	82.00
BasketballDrive	83.22	80.75	83.09	78.49	81.39
BQMall	83.12	84.62	82.17	77.41	81.83
RaceHorsesC	82.96	81.70	78.50	79.84	80.75
RaceHorsesD	80.48	83.90	80.55	81.32	81.56
BlowingBubbles	83.53	79.29	81.11	81.98	81.48
Average	82.39	82.30	81.42	79.89	81.50

B. Performance Comparison

Table VII presents the comparison results of proposed machine learning algorithm and a direct-mapping, non-machine learning based, VP9-to-HEVC transcoder. The direct-mapping scheme only uses sub-block counting as the key value for mapping. The sub-block count is compared to full-splitting numbers at each depth, which is {1, 4, 16, 64}, to decide the CU level in transcoding process. For example, in Fig. 2 (left), the sub-block counting for 64×64 block is 22. Comparing 22 with {1, 4, 16, 64}, we will find that 22 is closest to 16, so the direct-mapping scheme chooses to stop splitting at the 16×16 level. In the algorithm design process, we are able to implement direct-mapping when there is only one feature scenario. In case of multiple features scenarios, such as ten features in this paper, it is almost impossible to manually design reliable mapping methods. However, data-driven machine learning algorithm can easily provide us the mapping relationship with a great convenience while still maintaining good BD-rate performance. From Table VII we can see that direct-mapping scheme is poor in quality compared to the proposed ML algorithm, although they are similar in speed.

TABLE VII. COMPARISON RESULTS

	Direct-Mapping		Proposed ML	
	BD-rate	**Speed**	**BD-rate**	**Speed**
BQTerrace	9.4%	1.94	6.3%	1.89
BasketballDrive	6.7%	2.27	4.2%	2.14
BQMall	3.7%	1.65	2.5%	1.69
RaceHorsesC	2.3%	1.76	1.5%	1.71
RaceHorsesD	1.1%	1.49	1.2%	1.55
BlowingBubbles	1.8%	1.59	1.5%	1.63
Average	4.2%	1.78	2.8%	1.77

VI. CONCLUSION

This paper presented an effective and fast machine learning based VP9-to-HEVC video transcoder. Two additional VP9 features were introduced to enhance the VP9-to-HEVC mapping and models building. The experimental results confirmed that the time reduction reaches to a maximum of 53% and an average of 44% compared to the full re-encoding transcoder. All these experimental results demonstrated the effectiveness of the proposed transcoder. For the future work, more advanced ML algorithms could be used for model building. VP9 intra decisions could also be taken into consideration to speed up the HEVC intra coding process.

ACKNOWLEDGMENT

This work was supported by NSFC (61671296 and 61521062), the 111 Project (B07022 and Sheitc No.150633) and the Shanghai Key Laboratory of Digital Media Processing and Transmissions.

REFERENCES

[1] ITU-T, "H.265: High Efficiency Video Coding." ITU-T, Tech. Rep., June 2013.

[2] ISO/IEC, "Iso/iec 23008-2:2013," ISO/IEC, Tech. Rep., November 2013.

[3] ITU–T, "Advanced video coding for generic audiovisual services," Recommendation ITU-T H.264, Feb. 2014.

[4] A. Grange and H. Alvestrand, "A VP9 Bitstream Overview," Internet draft, Feb. 2013

[5] J. Bankoski, P. Wilkins, and Y. Xu, "VP8 Data Format and Decoding Guide," Internet draft, Nov. 2011.

[6] E. Peixoto and E. Izquierdo, "A Complexity-Scalable Transcoder from the H264/AVC to the new HEVC codec," in Proc. 2012 IEEE International Conference on Image Processing, Orland, Florida, USA, pp. 737-740, Sep. 2012.

[7] E. Peixoto, B. Macchiavello, E. M. Hung, A. Zaghetto, T. Shanableh, and E. Izquierdo, "An H.264/AVC to HEVC video transcoder based on mode mapping," in Proc. IEEE Int. Conf. Image Process. (ICIP), Melbourne, Vic., Australia, Sep. 2013, pp. 1972–1976.

[8] E. Peixoto, T. Shanableh, and E. Izquierdo, "H.264/AVC to HEVC Video Transcoder based on Dynamic Thresholding and Content Modeling," IEEE Trans. CSVT, vol. 24, pp. 99-112, Jan. 2014.

[9] A. J. Diaz-Honrubia, J. L. Martinez, J. M. Puerta, J. A. Gamez, J. de Cock, and P. Cuenca, "Fast Quadtree Level Decision Algorithm for H.264/HEVC Transcoder," in Proc. ICIP, 2014, pp. 2497-2501.

[10] A. J. Diaz-Honrubia, J. L. Martinez, P. Cuenca, J. A. Gamez, and Jose M. Puerta. "Adaptive Fast Quadtree Level Decision Algorithm for H.264/HEVC Video Transcoding," IEEE Trans. CSVT, vol. 26, pp. 154-168, Jan. 2016.

[11] Dongdong Zhang, Jie Tong, and Di Zang, "Fast CU Partition for H.264/AVC to HEVC Transcoding Based on Fisher Discriminant Analysis," in Proc. IEEE Int. Conf. Image Process. (ICIP), Chengdu, China, Nov. 2016

[12] E. De La Torre, R. Rodriguez-Sanchez and J. L. Martinez, "Fast Video Transcoding from HEVC to VP9," IEEE Trans. Consumer Electronics, vol. 61, pp.336-343, August 2015

[13] Adrian Grange, Peter de Rivaz and Jonathan Hunt, "VP9 Bitstream and Decoding Process Specification, " unpublished.

[14] M. Minsky, "Steps toward artificial intelligence," Trans. Inst. Radio Eng., vol. 4, pp. 8–30, Jan. 1961.

[15] P. Domingos and M. Pazzani, "Beyond independence: Conditions for the optimality of the simple Bayesian classifier," in Proc. Int. Conf. Mach. Learn., Jul. 1996, pp. 105–112.

[16] G. Bjontegaard, "Improvements of the BD-PSNR model," ITU-T Telecommunications Standardization Sector, Video Coding Experts Group (VCEG), Tech. Rep. VCEG-AI11, July 2008.