Machine Learning Based Fast Intra Mode Decision for HEVC Screen Content Coding Via Decision Trees

Wei Kuang, *Student Member*, *IEEE*, Yui-Lam Chan, *Member*, *IEEE*, Sik-Ho Tsang, *Member*, *IEEE*, and Wan-Chi Siu, *Life Fellow*, *IEEE*

***Abstract*—The Screen Content Coding (SCC) extension of High Efficiency Video Coding (HEVC) improves coding gain for screen content videos by introducing two new coding modes, intra block copy (IBC) and palette (PLT) modes. However, the coding gain is achieved at the increased cost of computational complexity. In this paper, we propose a decision tree based framework for fast intra mode decision by investigating various features in training sets. To avoid the exhaustive mode searching process, a sequential arrangement of decision trees is proposed to check each mode separately by inserting a classifier before checking a mode. As compared with the previous approaches that both IBC and PLT modes are checked for screen content blocks (SCBs), the proposed coding framework is more flexible which facilitates either IBC or PLT mode to be checked for SCBs such that computational complexity is further reduced. To enhance the accuracy of decision trees, dynamic features are introduced which reveal the unique intermediate coding information of a coding unit (CU). Then, if all modes are decided to be skipped for a CU at the last depth level, at least one possible mode is assigned by a CU type decision tree. Furthermore, a decision tree constraint technique is developed to reduce the rate-distortion performance loss. Compared with the HEVC-SCC reference software SCM-8.3, the proposed algorithm reduces computational complexity by 47.62% on average with a negligible Bjøntegaard delta bitrate (BDBR) increase of 1.42% under all-intra (AI) configuration, which outperforms all state-of-the-art algorithms in the literature.**

***Index Terms*—Screen Content Coding (SCC), High Efficiency Video Coding (HEVC), fast algorithm, machine learning, decision tree.**

I. Introduction

SCREEN content video is an emerging video type due to the fast development of the Internet and wireless communication, and it has been applied to many applications, such as online education, remote desktop, and web conferencing [1]. Screen content videos often show a mixed content with both nature image blocks (NIBs) and computer-generated screen content blocks (SCBs) in a single frame, as shown in Fig. 1.

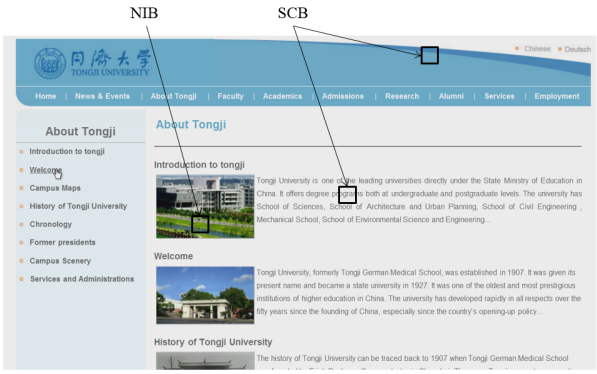


Fig 1. NIB and SCB in the first frame of "WebBrowsing".

Compared with NIBs, SCBs exhibit different characteristics, including no sensor noise, large flat areas with a single color, repeated patterns and limited colors. While NIBs can be well compressed by the conventional intra (Intra) mode in High Efficiency Video Coding (HEVC) [2], new techniques are necessary for SCBs. Therefore, the Joint Collaborative Team on Video Coding (JCT-VC) has developed Screen Content Coding (SCC) extension [3] on top of HEVC to explore new encoding tools for screen content videos since January 2014, and it was finalized in 2016.

In SCC, two new intra coding tools, intra block copy (IBC) mode [4] and palette (PLT) mode [5], [6] are particularly effective in addressing the blocks with repeated patterns and limited colors, respectively. However, they bring significant burden to a SCC encoder, which take up over 50% encoding time of the mode searching process.

To simplify the encoding process of HEVC, a fast CU partitioning algorithm was proposed in [7] by using Bayesian decision rule. CU partitioning process is early terminated by using joint online and offline learning. In [8], a fast mode decision algorithm was proposed to predict the RD cost and bit cost of a CU based on the statistical analysis. Then unnecessary modes are skipped according to the prediction. In [9], both the mode searching process and the CU partitioning process are terminated adaptively by analyzing the RD cost of the current CU. Although they work well for computational complexity reduction of HEVC, they are not suitable for SCC in which new coding modes such as IBC and PLT have been adopted.

To reduce the computational complexity of SCC, fast mode searching algorithms were designed in [10]–[12], and fast CU

Manuscript received May 8, 2018; revised October 24, 2018; accepted February 20, 2019. This work was supported by the Center for Signal Processing, Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, the research studentship provided by the University, and a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Grant No. PolyU 152112/17E).

The authors are with the Center for Signal Processing, Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Kowloon, Hong Kong (e-mail: wei.kuang@connect.polyu.hk; enylchan@polyu.edu.hk;sik-ho.tsang@polyu.edu.hk;enwcsiu@polyu.edu.hk).

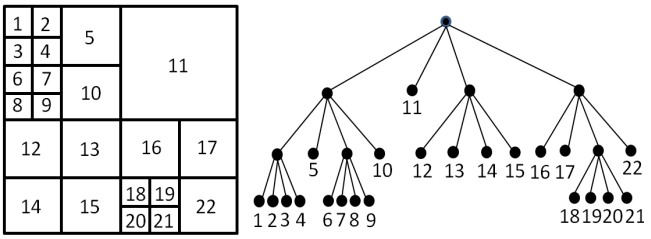


Fig. 2. A CTU partition and its corresponding partitioning structure.

size decision algorithms were suggested in [13]–[15]. Then, various algorithms were integrated to make both fast mode decision and CU size decision in [16]–[18]. In [10], a new mode was proposed to fill a noiseless smooth CU by its boundary samples. In [11], a hash value is calculated to adaptively skip the local search process in IBC mode. In [12], IBC mode is skipped for zero activity CUs and low gradient CUs. In [13], a neural network based fast algorithm was proposed to make fast CU size decision by utilizing features that describe CU statistics and sub-CU homogeneity. However, high RD performance loss is induced by this approach. In [14], for static regions, collocated CU depth and mode information is utilized to predict the current CU size. Besides, an approach with adaptive searching step was proposed to simplify the block matching process of IBC mode. However, this algorithm is not suitable for screen content videos with many dynamic regions. In [15], a fast CU size decision algorithm based on entropy was proposed. Some rules are firstly set based on entropy to make CU partitioning decision, and then the coding bits are used to improve the decision accuracy. The algorithms in [16]–[18] are mainly based on the assumption that NIBs select Intra mode while SCBs select IBC and PLT modes. They then classify CUs into NIBs and SCBs to make fast mode decision. In [16], a decision tree based classifier was firstly designed to classify CUs into NIBs and SCBs, so that NIBs only check Intra mode while SCBs check IBC and PLT modes. Besides, to speed up the encoding of NIBs, two classifiers were designed to predict the Intra mode direction from 35 prediction modes and early terminate the partitions of NIBs, respectively. In [17], a CU type classification is performed by CU content analysis. While IBC and PLT modes are skipped for some smooth NIBs, all modes are checked for SCBs and non-smooth NIBs. Then the depth information of temporal and spatial neighbor CUs as well as coding bits are utilized to make fast CU size decision. In [18], Intra mode is firstly checked for all CUs with 2N×2N prediction units (PUs), and then an early CU partitioning decision is made. If a CU is classified as a partitioning CU, it directly goes to the next depth level. Otherwise, it is further classified as a SCB or NIB. If it is a SCB, both IBC and PLT modes are checked. If it is a NIB, only Intra mode for N×N PUs in the depth level of 3 is tested. Although the methods in [16]–[18] provide better performance compared with the previous works, they mainly focus on the fast encoding of NIBs. For SCBs, either both IBC and PLT modes or all modes need to be checked. Therefore, it is desired that mode candidates can be further reduced for SCBs.

In this paper, we propose a machine learning based fast intra mode decision algorithm for SCC. By extracting some features from the original encoding process, fast mode decision is modeled as a data classification problem which is applied to predict whether a certain mode is checked or not. The classification is efficiently solved by using decision tree based classifiers. Specifically, decision tree based classifiers were also adopted in [16] and [18]. However, they treat the decisions for IBC and PLT modes the same, and they simply designed CU type classifiers rather than mode classifiers. As a result, a SCB needs to check both IBC and PLT modes by using the classifiers in [16] and [18], although a SCB is finally encoded by either IBC mode or PLT mode. To address this problem, we propose a more flexible coding framework by inserting a classifier before checking a mode in a CU that is completely different from [16] and [18]. The flexible coding framework makes mode decision for various modes sequentially, so that many SCBs can check only one mode from IBC mode or PLT mode. This new coding framework considers all modes one by one which has the following two advantages. (1) The previous fast mode decision approaches in [16]–[18] only classify CUs into SCBs and NIBs, so that IBC and PLT modes are always checked together for SCBs. On the contrary, the coding framework proposed in this paper performs mode decision one by one, and it allows the case that only one mode is checked for SCBs such that computational complexity can be further reduced; (2) The proposed coding framework facilitates the use of the dynamic features newly suggested in this paper, while only static features describing CU content are used in [16]–[18]. These dynamic features vary as a CU goes through different classifiers, which provides more precise intermediate coding information of a CU to the classifiers, resulting in accurate mode decision in SCC. Intermediate coding information is referred to as the best mode or the best RD cost so far of the current CU before checking a target mode. Besides, a feature subset selection approach is applied to allow classifiers to select feature subsets for different tasks adaptively, such as for different modes in different depth levels. Therefore, the impact of irrelevant or redundant features is removed, and a valuable insight into feature importance is provided for different tasks.

The rest of this paper is organized as follows. Section II presents the overview and analysis of intra mode decision in SCC. Section III presents the proposed fast mode decision techniques for SCC. The experimental results are presented in Section IV to verify the performance of the proposed work. Finally, Section V concludes the paper.

II. Overview and Analysis of Intra Mode Decision in SCC

SCC inherits the quadtree-based block partitioning scheme from HEVC and the intra mode decision process is performed for CUs with different sizes recursively. An example of a CTU partition and its corresponding partitioning structure is shown in Fig. 2. In SCC, a frame is divided into non-overlapping CTUs of 64×64 pixels (depth level of 0). Then a CTU is further divided into 4 CUs of 32×32 pixels (depth level of 1), and this partitioning process continues until CUs of 8×8 pixels (depth level of 3) are reached. To efficiently encode a CU, two additional modes, IBC mode and PLT mode, are introduced. IBC mode is a block matching based intraframe approach, and it is also referred to as motion compensation in the same picture in the international coding standard. For the sake of simplicity,

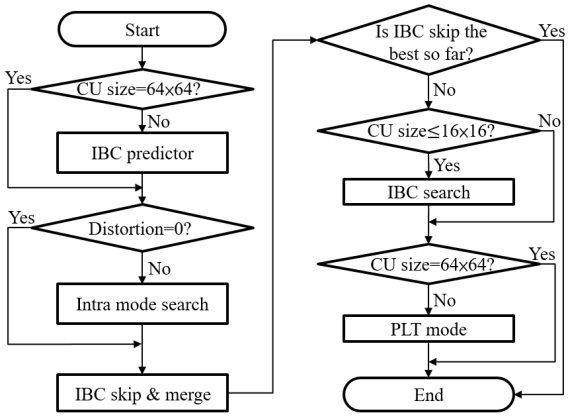


Fig. 3. Encoding procedure implemented in SCM.

its short form "IBC" is used in this paper. IBC mode searches in the reconstructed regions of the current frame to find the best reference block for the current CU. It includes three steps – IBC predictor, IBC merge & skip and IBC search. IBC predictor simply checks a set of block vectors (BVs) from the two last encoded CUs and the neighbor CUs of left, above, collocated, below left, above right and above left. IBC merge & skip is the intra version of the merge and skip mode for inter prediction in HEVC, where IBC merge signals residues to a SCC decoder but IBC skip does not. IBC search finds the best matched block in the reconstructed region of the current frame for 16×16 CUs and 8×8 CUs, and it provides different searching strategies for different CU and PU sizes. The searching strategies include the full vertical and horizontal searches, local vertical and horizontal 1D searches, and 2D pre-defined area search. For a 16×16 CU, only the 2N×2N PU with full vertical and horizontal searches are performed. It is due to the fact that a large CU size tends to have fewer repeated patterns within the same frame. For an 8×8 CU, additional PU sizes are allowed to find more repeated patterns. If it is a N×2N PU, only full vertical and horizontal searches are carried out. If it is a 2N×N or 2N×2N PU, local vertical and horizontal 1D searches, and 2D pre-defined area search within the current CTU and left CTU are performed. Besides, a hash value based fast searching method is implemented for 8×8 CUs with 2N×2N PUs, where only blocks having the same hash value as the current CU are searched. Therefore, IBC search comes with the highest computational complexity among the three steps. A detailed technical overview of IBC can be found in [4]. PLT mode is designed to improve the encoding efficiency for CUs with limited colors. Several representative colors in a CU are selected to form a palette table. Then an index map is generated to indicate the index of the representative color for each pixel location. In the encoding process of a CU, Intra mode, IBC mode and PLT mode are exhaustively checked, and the encoding procedure implemented in the HEVC-SCC reference software, Screen Content Model (SCM), is shown in Fig. 3. At the beginning, IBC predictor is checked for CUs with sizes from 32×32 down to 8×8, If the distortion is zero after checking IBC predictor, Intra mode inherited from HEVC is skipped. Otherwise, Intra mode is checked, which includes 33 directional modes, plus planar and DC modes. Then it is followed by checking BV predictors of IBC skip & merge for all CUs. If IBC skip is selected as the best mode among IBC predictor, Intra, and IBC skip & merge, further mode searching is terminated. Otherwise, if the best mode is IBC predictor, Intra or IBC merge, the following IBC search and PLT modes are checked. Specifically, only CUs with sizes of 16×16 and 8×8 need to check IBC search. Finally, PLT mode is checked for CUs with sizes from 32×32 down to 8×8. In the mode searching process, the coding performance of each mode is evaluated by calculating a Lagrange RD cost function, J_{mode}, as

$$J_{mode} = D_{SSE} + \lambda \times R_{mode} \quad (1)$$

where D_{SSE} denotes the sum of the squared error between the current CU and its reconstructed CU, λ is a Lagrange multiplier and R_{mode} is the actual encoding bits for signaling the mode and the residues. The mode with the smallest RD cost is selected as the best mode of the CU. All CU partitions in a CTU need to go through this mode searching process, and the final partitioning structure of a CTU is selected as the one with the smallest RD cost, and it is involved in the final encoding bitstream.

To have a better understanding of the intra mode decision process in SCC, several experiments were performed. First, to analyze the computational complexity distribution in SCC, we encoded SCC test sequences by SCM of version 8.3, SCM-8.3 [19]. The test sequences are shown in Table I, which were selected by the experts in the JCT-VC group. They are classified into 4 categories according to their video content, where TGM represents text and graphics with motion, M represents mixed content, A represents animation and CC represents camera-captured content. While sequences in TGM and M are typical screen content videos that contain both NIBs and SCBs, the sequence in A is similar to the camera-captured content video. Therefore, sequences in A and CC are grouped together for the analyses in the following sections. Sequences marked with *T* are used for extracting training frames, and sequences marked with

TABLE I
SCC TEST SEQUENCES IN 4 CATEGORIES

Categories	Sequences	Resolution	No. of Frame	Frame Rate (Hz)
TGM	ChineseEditing (*T*)	1920×1080	0-599	60
	Console (*NT*)	1920×1080	0-599	60
	Desktop (*NT*)	1920×1080	0-599	60
	FlyingGraphics (*T*)	1920×1080	0-299	60
	Map (*T*)	1280×720	0-599	60
	Programming (*NT*)	1280×720	0-599	60
	SlideShow (*T*)	1280×720	0-499	20
	WebBrowsing (*NT*)	1280×720	0-299	30
M	BasketballScreen (*T*)	2560×1440	322-621	60
	MissionControlClip2 (*T*)	2560×1440	120-419	60
	MissionControlClip3 (*NT*)	1920×1080	0-599	60
A	Robot (*T*)	1280×720	0-299	30
CC	EBURainFruits (*T*)	1920×1080	0-249	50
	Kimono1(*NT*)	1920×1080	0-119	24

T: Training; *NT*: unseen sequences to the trained decision trees.

Table II
ENCODING TIME DESTRIBUTION OF EACH MODE

CU size	Intra (%)	IBC			PLT (%)
		Predictor (%)	Merge & Skip (%)	Search (%)	
64×64	5.07		3.37		
32×32	4.82	0.24	5.14		5.56
16×16	7.43	0.54	7.04	6.30	4.65
8×8	23.10	0.46	3.79	17.91	4.58
Total	40.42	1.24	19.34	24.21	14.79
			44.78		

TABLE III
BDBR AND ENCODING TIME INCREASE BROUGHT BY DISABLING DIFFERENT MODES COMPARED WITH ORIGINAL SCC

Sequences	Disable IBC		Disable PLT		Disable PLT+IBC		Disable Intra	
	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)
ChineseEditing	21.41	-35.98	50.86	-14.16	142.21	-51.61	0.88	-30.85
Console	40.20	-26.67	42.87	6.63	239.62	-36.02	0.69	-31.58
Desktop	115.02	-29.37	39.91	3.25	451.92	-41.04	1.49	-30.26
FlyingGraphics	72.25	-20.78	24.30	-2.24	191.38	-48.03	2.67	-26.60
Map	9.46	-42.78	19.03	-13.64	49.19	-49.98	11.67	-20.30
Programming	29.94	-23.32	16.43	-11.58	101.47	-43.70	9.32	-28.00
SlideShow	13.49	-20.77	8.00	-20.89	33.12	-35.84	15.65	-34.24
WebBrowsing	130.36	-27.71	34.80	-4.60	469.80	-40.96	6.95	-24.11
BasketballScreen	40.16	-28.10	15.69	8.10	109.88	-47.77	15.54	-17.74
MissionControlClip2	26.18	-29.49	7.78	-17.69	55.68	-50.40	17.56	-19.41
MissionControlClip3	45.38	-28.38	13.01	-11.03	121.55	-46.94	16.32	-20.10
Robot	1.37	-33.02	0.84	-20.33	2.35	-52.63	32.86	6.87
EBURainFruits	0.15	-36.42	-0.04	-13.33	0.10	-47.86	32.02	36.60
Kimono1	0.02	-29.79	-0.02	-13.70	-0.02	-40.47	43.29	44.90
Average (TGM+M)	49.44	-28.49	24.79	-7.08	178.71	-44.75	8.98	-25.74
Average (A+CC)	0.51	-33.08	0.26	-15.7	0.81	-46.99	36.06	29.46
Average (ALL)	38.96	-29.47	19.53	-8.94	140.59	-45.23	14.78	-13.92

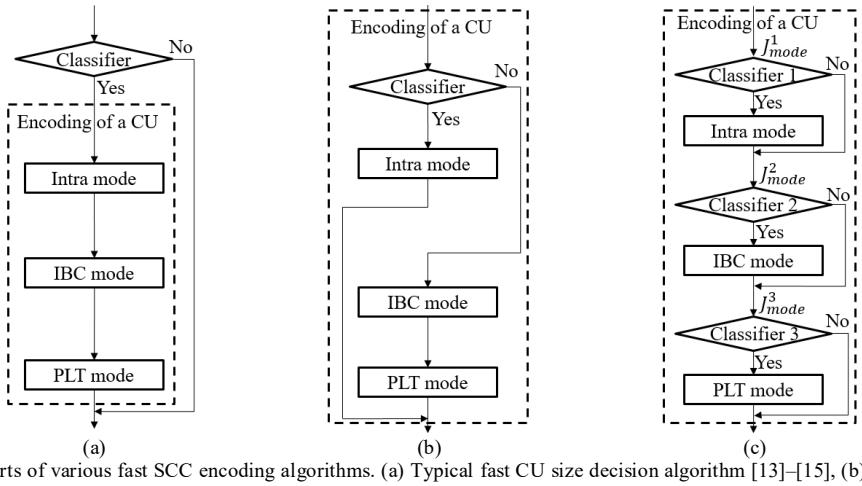


Fig. 4. CU encoding flowcharts of various fast SCC encoding algorithms. (a) Typical fast CU size decision algorithm [13]–[15], (b) typical fast mode decision algorithm by CU type classification [16]–[18], and (c) proposed fast mode decision algorithm.

NT are unseen sequences to the trained decision trees. They will be explained in Section III. A. All test sequences were encoded with quantization parameters (QPs) of 22, 27, 32, and 37 using SCM-8.3 under All Intra (AI) configuration and the common test conditions (CTC) [20]. The test platform used for simulations was a HP EliteDesk 800 G1 computer with a 64-bit Microsoft Windows 10 OS running on an Intel Core i7-4790 CPU of 3.6 GHz and 32.0 GB RAM. Table II shows the distribution of encoding time for each mode. We can see from the table that Intra mode and IBC mode are the two modes with very high computational complexity, which take up 40.42% and 44.78% of the total encoding time, respectively. Among the three steps of IBC mode, IBC predictor has negligible computational complexity while IBC merge & skip and IBC search bring high computational burden to the SCC encoder. It is also interesting to analyze the impact on quality of omitting certain mode candidate in SCC. Experiments were performed by disabling IBC mode, PLT mode, IBC+PLT modes and Intra mode, respectively. Table III shows the Bjøntegaard delta bitrate [21] and the change in encoding time brought by disabling different modes compared with the original SCM-8.3, which are denoted by BDBR and ΔTime, respectively. It should be noted that a negative value of BDBR or ΔTime denotes decrement in percentage as compared with SCM-8.3. We can see from the table that the smallest BDBR increase is obtained with 14.78% on average if Intra mode is disabled, but encoding time is only reduced by 13.92%, which is not enough considering the high computational complexity of SCC. Besides, disabling IBC mode achieves the largest encoding time reduction by 29.47% on average, but it brings a very high increase in BDBR of 38.96%. However, it can be observed that for sequences in A and CC, encoding time is reduced by 46.99% while BDBR is increased by only 0.81% on average if PLT mode and IBC mode are both disabled. For a sequence with almost pure SCBs, such as “ChineseEditing”, “Console” or “Desktop”, disabling Intra mode leads to about 30% encoding time reduction while less than 1.5% increase in BDBR is observed. This observation proves that NIBs usually select Intra mode while SCBs usually select IBC mode or PLT mode. It should be noted that for the sequences with almost pure SCBs, the results of disabling Intra mode are the upper limit of the fast mode decision methods [16]–[18] which only classify CUs into NIBs and SCBs. To break the limit, it is desired the further classifications are made inside IBC and PLT modes for SCBs.

III. PROPOSED DECISION TREE BASED FRAMEWORK FOR FAST INTRA MODE DECISION

The previous fast SCC encoding algorithms are mainly focused on fast CU size decision and fast mode decision made by CU type classification, as shown in Fig. 4(a) and (b),

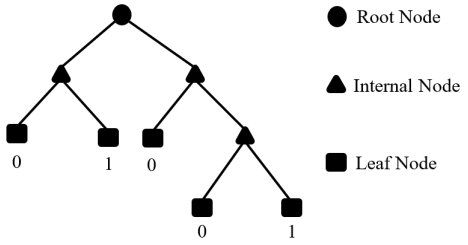


Fig. 5. General structure of a decision tree based classifier.

respectively. However, these frameworks are not flexible and are difficult to achieve a good tradeoff between the computational complexity and coding efficiency. For fast CU size decision approaches [13-15], all modes are either checked or skipped together in a CU as shown in Fig. 4(a). For fast mode decision approaches [16-18] using CU type classification in Fig. 4(b), the screen content modes, IBC mode and PLT mode, are either checked or skipped together. In screen content videos, some CUs are very difficult to be decided whether they are SCBs or NIBs even by human beings, and the CU type classification approaches are not efficient for these CUs. On the contrary, our proposed framework provides larger flexibility by inserting a classifier before checking a mode in a CU, as shown in Fig. 4(c). By deriving the dynamic features right before checking a mode, more accurate decision is made. On the one hand, encoding time can be further reduced by allowing the case that only one mode is checked for a SCB. On the other hand, RD performance can be improved by allowing PLT mode to be checked for a SCB even if IBC mode is wrongly skipped. It is also noted that the values of the dynamic features newly defined in this paper are changing as a CU goes through different classifiers, and only our framework in Fig. 4(c) can adopt these dynamic features proposed in this work. Since there are numerous mode candidates in different CU sizes, it is difficult to manually select the optimal features and classification criterions to build accurate mathematical models. To solve this problem, 11 features, which are related to the mode decision, are proposed to train various decision tree based classifiers from off-line learning. Therefore, the optimal features and classification criterions are reasonably selected based on the training data. In the test phase, the trained classifiers are implemented in SCM to make fast mode decision.

A. Description of the Classifier Using Decision Tree

Decision tree is one of the most popular machine learning algorithms. In this paper, we utilize a decision tree as the classifier, because it comes with low complexity in the testing phase and can be easily implemented into a SCC encoder as a set of "if-then-else" conditions. A decision tree based classifier is a flowchart like tree structure, as shown in Fig. 5. It is composed of a root node, internal nodes and leaf nodes. For each non-leaf node, i.e., a root node or an internal node, it denotes a test on a feature of the incoming sample. Each branch after a non-leaf node denotes the outcome of the test, and each leaf node denotes a class label. In the specific case of the mode selection problem in a CU, the class label of 1 or 0 represent whether the target mode is checked or not.

The classifiers based on decision trees are trained by the C4.5 algorithm [23] in the Waikato Environment for Knowledge Analysis (WEKA) [24] version 3.8 in this paper. To generate training frames which reflect the characteristic of SCC sequences, 8 frame-skipped sequences are formed by extracting the first frame of each second from the sequences marked with T in Table I. Training frames from different sequences were encoded by the original SCM-8.3 encoder with QPs at 22, 27, 32, and 37 using AI configuration to generate training data.

If a node of a decision tree only contains samples from one class, it is defined to have pure samples. Otherwise, the impurity is calculated to represent how impure the samples in the node are. To reduce the impurity of the node, a feature A with a classification threshold TH_A is selected to further classify the samples into two child nodes, and the impurity reduction is calculated by comparing the impurities of two child nodes and the parent node. In the training process of a decision tree, the impurity reduction by splitting a parent node to two child nodes is calculated iteratively for each feature A with a classification threshold TH_A. The larger the impurity reduction is, the better the feature and the classification threshold are. In the C4.5 algorithm, the impurity is calculated by entropy. Then the impurity reduction with A and TH_A is measured by the gain ratio $GainRatio(A, TH_A)$

$$GainRatio(A, TH_A) = \frac{InfoGain(A,TH_A)}{SplitInfo(A,TH_A)} \quad (2)$$

where $InfoGain(A, TH_A)$ is the information gain by splitting a node t_0 into its child nodes t_L, t_R using a feature A with a threshold TH_A. It is calculated by the entropy reduction after splitting as

$$InfoGain(A, TH_A) = En(t_0) - \sum_i \frac{N_{t_i}}{N_{t_0}} En(t_i) \quad (3)$$

where N_{t_0} and N_{t_i} represent the number of samples in the node t_0 and child nodes t_i, $i \in \{L, R\}$. Let $p(\omega_j)$ be the probability of training samples belonging to the class ω_j in a node t, $j \in \{0,1\}$. The entropy $En(t)$ in the node t is calculated as

$$En(t) = -\sum_{j=0}^{1} p(\omega_j) log_2\, p(\omega_j). \quad (4)$$

The normalization term $SplitInfo(A, TH_A)$ is defined by

$$SplitInfo(A, TH_A) = -\sum_i \frac{N_{t_i}}{N_{t_0}} log_2 \frac{N_{t_i}}{N_{t_0}}. \quad (5)$$

The best feature and the threshold are selected as the ones with maximum gain ratio to split a node. A decision tree is trained node by node, and the splitting of a node is terminated if the number of training samples arrived the node is less than or equal to 1% of the total training samples. Then a reduced error pruning process [25] is performed to prune the decision tree backward to avoid overfitting. After generating a decision tree, the classification accuracy of the tree is given by a 10-fold cross-validation process [26], which calculates the percentage of correctly classified samples in the total training samples.

B. Proposed Dynamic Features and Their Advantages

In general, the precision of SCC mode decision in a classification task is highly dependent on the feature space used to train the model. In most of the machine learning algorithms adopted in mode decision of video coding, the features extracted from a CU is always determined by its static content, such as background color number, gradient, etc. These features are called as static features in this paper.

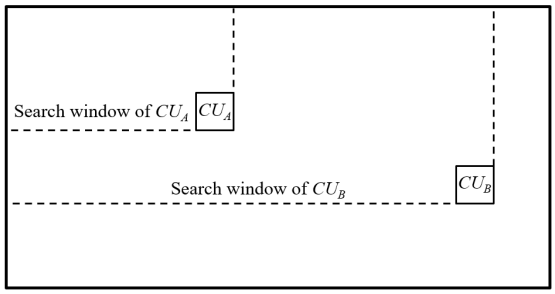


Fig. 6. Two CUs with same content in a frame.

In contrast, we find that the probability of selecting IBC mode as the optimal one depends on the spatial location of the current CU, as shown in Fig. 6. Assume that CU_A and CU_B in the example of Fig. 6 have the same static content. Even though the static features extracted from CU_A and CU_B are the same, the mode decision of these two CUs may be different. For example, CU_A may select PLT mode while CU_B with the same content is likely to select IBC mode because the search window of CU_B is larger resulting in a higher chance to find a good repeated pattern with very low RD cost by using IBC mode. By taking this specific characteristic of screen content videos into account, we propose to extract the IBC mode flag of the current CU before checking the target mode, $Flag_{IBC}$, (**Feature 1**) as dynamic information. If the best mode so far of the current CU before checking the target mode is a sub-mode (i.e. IBC predictor, IBC merge & skip, or IBC search) of IBC mode, $Flag_{IBC}$ is set to 1. Otherwise, $Flag_{IBC}$ is set to 0. It is noted that, for CU_B in Fig. 6, the chance of $Flag_{IBC}$ equal to 1 is higher as compared with that of CU_A even though they have same content. Therefore, this feature may vary according to the spatial location, and it is considered as a dynamic feature.

In addition, the dynamic RD cost of the best mode so far in the current CU before checking the target mode, J_{mode} (**Feature 2**) is another dynamic feature proposed in this paper for fast mode decision. Similarly, J_{mode} of CU_B in Fig. 6 is likely to become smaller since it is easier to get a good repeated pattern in the reconstructed area. Besides, J_{mode} is not only related to the spatial location but varies during the encoding process. For instance, J^2_{mode} to Classifier 2 in Fig. 4(c) may be different from J^3_{mode} to Classifier 3 since J^3_{mode} has already gone through Intra mode and IBC mode while only intra mode is tested for computing J^2_{mode}. The variation property is well suited for our proposed framework in Fig. 4(c) in which the values of this dynamic feature entered to various decision trees are different. This new arrangement is of great importance to SCC mode decision process using classification, which will be verified in the following sections. J_{mode} reveals the unique intermediate coding information of a CU, and its value varies as the CU goes through different decision trees. By implementing decision trees right before the target mode, the most updated values of these dynamic features (**Feature 1** and **Feature 2**) are obtained for different trees to improve the decision accuracy.

By using the proposed framework with decision trees prior to checking a mode in a CU, the new dynamic features with the following nine static features are then selected based on our prior knowledge for the training of decision trees in Fig. 4(c).

Feature 3: Background color number *BCN*. The background color in a CU is defined as the color with the highest occurrence frequency within the CU, and *BCN* is calculated by counting the number of the background color pixels.

Feature 4: Distinct color number *DCN*. *DCN* is calculated by counting the pixels in a CU with different sample values.

For *BCN* and *DCN*, all three components of a pixel (Y, U, V in YVV 4:4:4 or R, G, B in RGB 4:4:4) are stacked to form a 24-bit sample value. For sequences in YUV 4:2:0 format, only the luminance component is utilized as an 8-bit sample value.

Features 5–8: High gradient pixel number HGN_0, HGN_1, HGN_2, HGN_3. The high gradient pixel is utilized to detect sharp edges in a CU. A pixel is defined as a high gradient pixel [27] if the luminance difference of the current pixel $Y_{i,j}$ and one of the neighbor pixels $Y_{i\pm1,j}$ and $Y_{i,j\pm1}$ located at 0°, 90°, 180° and 270° is larger than a threshold TH_S

$$\left|Y_{i,j} - Y_{i\pm1,j}\right| > TH_S \text{ or } \left|Y_{i,j} - Y_{i,j\pm1}\right| > TH_S \quad (6)$$

where i and j denote the row and column indices of the pixel. TH_S is a threshold controlling the sharpness of the edges for detection, which is set to 32 in [27]. To detect edges with different sharpness in our proposed algorithm, we set another three values to TH_S. Totally 4 different high gradient pixel numbers, HGN_0, HGN_1, HGN_2, and HGN_3, are calculated by counting high gradient pixels with TH_S at 4, 8, 16, and 32, respectively. Considering that the proposed algorithm is a machine learning based approach, it lets the decision tree select the features to be used based on the off-line training. It implies to select which value(s) of TH_S to be used in each decision tree. Therefore, we do not need to manually select which particular value(s) of TH_S in the final decision trees.

It is noted that sequences in RGB 4:4:4 format are converted to YUV 4:4:4 format to get the luminance component.

Features 9–10: CU horizontal and vertical activities *HorAct* and *VerAct*. They have been used for skipping IBC mode adaptively in the original SCM-8.3 and defined as

$$HorAct = \sum_{i=0}^{2N}\sum_{j=0}^{2N-1}|Y_{i,j} - Y_{i,j+1}| \quad (7)$$

$$VerAct = \sum_{j=0}^{2N}\sum_{i=0}^{2N-1}|Y_{i,j} - Y_{i+1,j}|. \quad (8)$$

Feature 11: CU variance *Var*. *Var* can well represent the smoothness of a CU, which is defined as

$$Var = \frac{1}{2N\times 2N}\sum_{j=0}^{2N}\sum_{i=0}^{2N}(Y_{i,j} - \bar{Y})^2 \quad (9)$$

where $\bar{Y}$ is the average luminance value over the current CU.

Features 3–11 are static features which have fixed values in a CU. Therefore, they are obtained once for a CU and shared among different decision trees.

C. Fast Mode Decision Design

To make fast mode decision in SCC, the selection of Intra mode, IBC mode, and PLT mode is investigated by adopting different decision trees in our new coding framework. Then, a decision tree for performing CU type classification is trained at the last depth level to avoid the situation that all modes are skipped for a CTU. In this sub-section, the detailed design of the new coding framework is discussed.

1) Feature Analysis

Among the three coding modes in SCC, Intra mode is the only mode inherited directly from HEVC. While Intra mode is very efficient for NIBs, IBC mode and PLT mode are both specially designed for SCBs. To perform fast mode decision, a

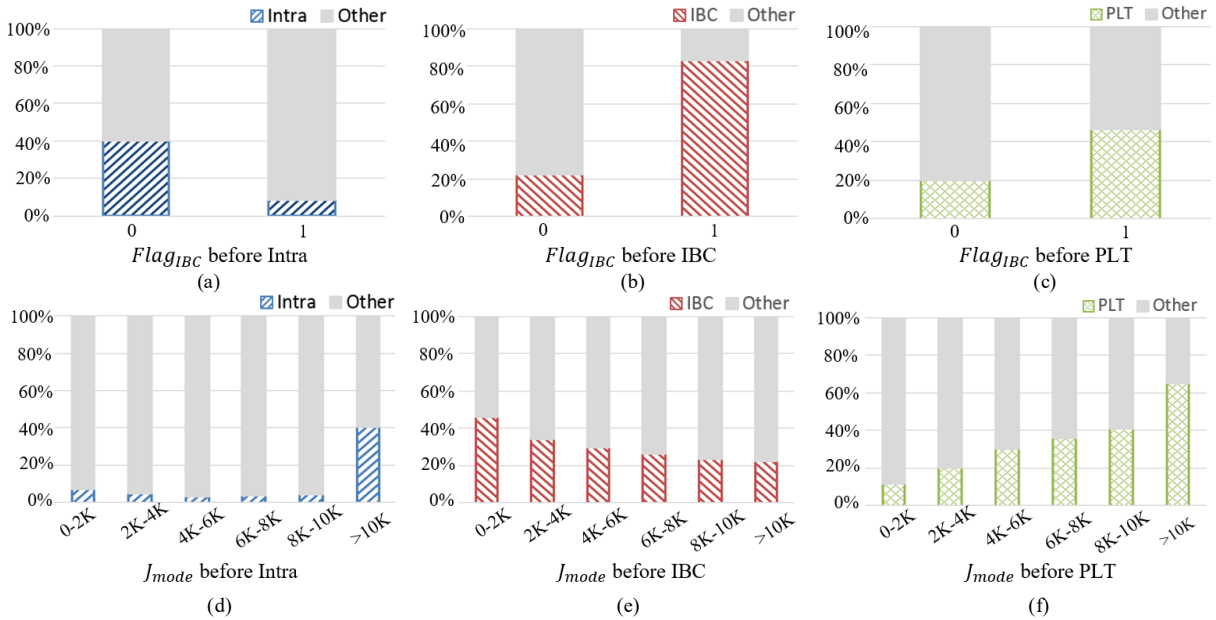


Fig. 7. The percentages of the target mode and other modes in terms of $Flag_{IBC}$ (a)–(c) and J_{mode} (d)–(e) for 16×16 CUs.

common idea is to classify CUs into NIBs and SCBs by analyzing their content characteristics. Then IBC and PLT modes are checked for SCBs while Intra mode is checked for NIBs. However, such approach is not optimal since IBC and PLT modes are always checked together for SCBs.

To understand the distributions of Intra, IBC and PLT modes over different features, we randomly selected 300,000 16×16 CUs from the training samples, and the number of the CUs with each mode is 100,000. First, the mode distributions over the dynamic features obtained right before the target mode were investigated. Fig. 7 shows the percentages of CUs selecting the target mode and other modes in terms of $Flag_{IBC}$ (Fig. 7(a)-(c)) and J_{mode} (Fig. 7(d)-(e)). If $Flag_{IBC}$ before checking the target mode is 1, the CU is more likely to be a SCB, otherwise, it more likely to be a NIB. Therefore, it is observed in Fig. 7(a) that the percentage of Intra mode is very low if $Flag_{IBC}$ before checking Intra mode is 1. On the contrary, the percentages of IBC mode and PLT mode are low if $Flag_{IBC}$ before checking the target mode is 0, as shown in Fig. 7(b) and Fig. 7(c), respectively. Before checking Intra mode, J_{mode} is highly correlated to $Flag_{IBC}$. If IBC predictor does not provide a valid BV for a CU, $Flag_{IBC}$ would be 0 and the value of J_{mode} becomes very large. Otherwise, the value of J_{mode} is relatively small if $Flag_{IBC}$ is 1. Therefore, the percentage of CUs selecting Intra mode is very low if the value of J_{mode} is small, as shown in Fig. 7(d). It is also observed in Fig. 7(e) that if J_{mode} before checking IBC mode is very large, the percentage of IBC mode would be low. The reason is that for CUs with very large values of J_{mode}, they usually have complex texture, and it is difficult to find repeated patterns for the complex texture CUs by IBC mode. Besides, Fig. 7(f) shows that the percentage of PLT mode is low for CUs with small value of J_{mode}. The reason is that the CU with small value of J_{mode} before checking PLT mode may have been efficiently encoded and the checking of PLT mode becomes unnecessary. The discrepancy between Fig. 7(e) and Fig. 7(f) verifies that PLT mode and IBC mode have different characteristics and should not use the same classifier when the dynamic features are adopted.

Then the mode distributions of the static features shared among different decision trees were also investigated. Fig. 8 shows the mode distributions in terms of 5 representative features: (a) *DCN*, (b) *BCN* (c) HGN_3, (d) *HorAct* and (e) *Var*. It is observed that the percentage of Intra mode increases as *DCN* gets larger, or *BCN*, HGN_3, *HorAct*, and *Var* get smaller. The reason is that Intra mode is designed for NIBs, and they tend to have larger *DCN*, smaller *BCN* and be smoother. Besides, it is also observed that the percentage of PLT mode is much higher than IBC mode when CUs get more complex, such as CUs with larger values of HGN_3, *HorAct* and *Var*. It further implies that PLT mode and IBC mode should not share the same classifier for SCC intra mode selection that is always adopted in the algorithms proposed in the literature [16]–[18].

Based on these observations, we trained decision trees in the proposed coding framework to adaptively check Intra mode, IBC mode and PLT mode separately.

2) Decision Tree Training

As described before, IBC mode contains three steps, which are IBC predictor, IBC merge & skip and IBC search. While the step of the IBC predictor only checks several BV predictors and our experiment shows that it takes up only 1.24% of the total encoding time, and the computational complexities of IBC merge & skip and IBC search are relatively high. Therefore, by collecting the most updated features, two sets of decision trees are generated inside IBC mode to adaptively check IBC merge & skip and IBC search. After generating the decision trees for all modes, they are implemented in the SCM-8.3 encoder to perform fast mode decision. Before checking a mode, the incoming CU goes through the decision tree for the mode to decide whether it should be tested. If the outcome or the class label of the decision tree is 1, it is involved in the mode searching process. Otherwise, the current CU does not check the target mode so that the computational complexity brought by this mode is reduced. However, there is a case that all modes are decided to be skipped for a CTU when all mode decision trees are implemented, and finally the CTU cannot be encoded. To solve this problem, a CU type decision tree is also trained at

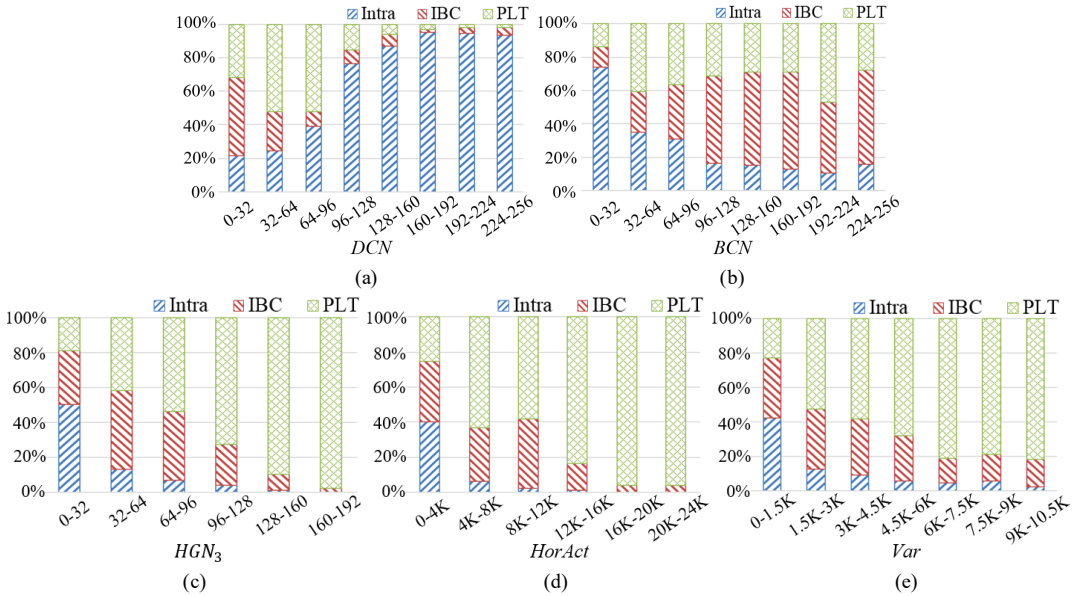


Fig. 8. Intra, IBC and PLT mode distributions in terms of (a) Distinct color number *DCN*, (b) high gradient pixel number HGN_3, (c) horizontal activity *HorAct*, and (d) CU variance *Var* for 16×16 CUs.

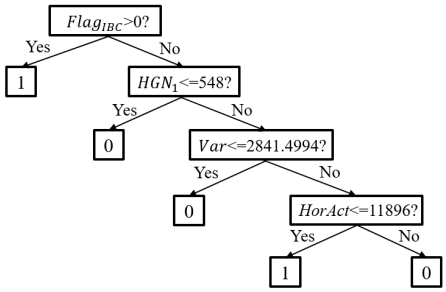


Fig. 9. IBC merge & skip mode decision tree for 32×32 CUs.

the last depth level, and at least one possible mode is assigned to the CU if all modes are skipped for it. The CU type decision tree can classify incoming CUs into NIBs and SCBs. If the outcome for a CU is a NIB, i.e., 1, Intra mode is checked for it. Otherwise, IBC and PLT modes are both checked for it.

As SCC supports CU sizes of 64×64, 32×32, 16×16, and 8×8, 4 decision trees are trained for CUs with different sizes. To avoid the data imbalance problem caused by more training samples in one class than in the other [28], we let 50% of the training samples come from CUs with the target mode as their optimal modes, and they are treated as the positive data. The other 50% of training samples come from the CUs which are not encoded by the target mode, and they are treated as the negative data. Besides, for the training of the CU type decision tree at the last depth level, the positive training data are from NIBs, i.e. CUs encoded by Intra mode, while the negative data are from SCBs, i.e. CUs encoded by IBC or PLT mode.

The training data number and the depth of each decision tree are shown in Table IV and V, respectively. Since a frame can be partitioned into more CUs with a small size than CUs with a large size, more training data are obtained as the CU size gets smaller. Besides, we can see that the largest depth of the trained decision trees is 14, which means the decision for a mode is made after going through at most 14 "if-then-else" conditions. Therefore, the computational complexity brought by those decision trees is negligible. As an example, the decision tree based IBC merge & skip mode classifier for 32×32 CUs is shown in Fig. 9, and other trained classifiers can be found in our website [22].

TABLE IV
TRAINING DATA NUMBER FOR EACH DECISION TREE

CU Size	Intra	IBC		PLT	CU Type
		Merge & Skip	Search		
64×64	28452	14224			
32×32	216072	111980		80804	
16×16	715548	573848	168736	219192	
8×8	3166280	2724108	1522712	453080	906164

TABLE V
DEPTH OF EACH DECISION TREE

CU Size	Intra	IBC		PLT	CU Type
		Merge & Skip	Search		
64×64	14	13			
32×32	7	4		6	
16×16	8	1	10	6	
8×8	9	9	6	7	7

3) Feature Subset Selection

When training classifiers for different tasks, the valid features are quite different, and the performance of a classifier is very sensitive to the features utilized to train the classifier. Therefore, to eliminate the impact of irrelevant or redundant features and provide a better understanding of the valid features for each mode decision, a feature subset selection [29] approach is applied in our paper.

We implemented the feature subset selection in WEKA using the wrapper evaluation with a greedy search strategy, which is computationally advantageous and robust against overfitting. The feature subset selection consists of the following steps:

Step 1: Initialize the feature subset set $F^k=\emptyset$ at $k=0$.

Step 2: Find the best remaining feature f which provides the largest accuracy increase when added to F^k.

Step 3: k++ and $F^k=F^{k-1}\cup\{f\}$.

Step 4: Iterate step 2 and step 3 until the classifier accuracy is no longer improved.

Table VI shows the valid features of each decision tree, and the importance of each valid feature is also shown in this table by measuring its gain ratio. It is observed that the proposed dynamic features, $Flag_{IBC}$ and J_{mode} obtained right before the

TABLE VI
THE GAIN RATIO OF EACH FEATURE FOR EACH DECISION TREE

Decision Tree		$Flag_{IBC}$	J_{mode}	BCN	DCN	HGN_0	HGN_1	HGN_2	HGN_3	$HorAct$	$VerAct$	Var
Intra	64×64			0.044	0.029		0.071	0.094		0.074	0.065	0.054
	32×32	0.179		0.043	0.022				0.084	0.045	0.046	
	16×16		0.102	0.035	0.029					0.029	0.025	0.027
	8×8	0.180		0.051	0.046			0.005		0.020	0.019	0.025
IBC Merge & Skip	64×64		0.065	0.099		0.084					0.097	0.100
	32×32	0.669					0.045			0.070		0.066
	16×16	0.431										
	8×8	0.178	0.014	0.032	0.019			0.024		0.017	0.018	0.025
IBC Search	16×16	0.275	0.007	0.028	0.030			0.019	0.023	0.010	0.016	0.022
	8×8	0.281		0.039	0.027			0.061	0.067	0.035	0.036	
PLT	32×32	0.001	0.048	0.064	0.041		0.026			0.035		
	16×16	0.166	0.038	0.035	0.025	0.009		0.034		0.031		
	8×8	0.105	0.027	0.029	0.021			0.031	0.039	0.027	0.023	
CU Type	8×8			0.058	0.045					0.031	0.028	

TABLE VII
DECICIOSN ACCURACY FOR EACH DECISION TREE

CU Size	Intra (%)	IBC		PLT (%)	CU Type (%)
		Merge & Skip (%)	Search (%)		
64×64	75.44	82.31			
32×32	87.95	94.51		82.27	
16×16	83.34	84.57	81.89	82.48	
8×8	78.23	83.13	85.66	79.83	83.17

target mode, are quite important for most decision trees, with the gain ratio up to 0.669 and 0.102, respectively. This verifies that the dynamic features play a critical role in the decision process with the introduction of the new coding framework. By adopting the feature subset selection approach, the number of features fed into a decision tree is reduced to 1–8, and 6.07 on average. Compared with the original feature set with 11 features, the feature number is reduced by 44.92%, and the impact of the feature subset selection approach in terms of coding performance will be discussed in Section IV.

4) Accuracy of Decision Trees

The decision accuracy for each decision tree is shown in Table VII. We can see from the table that the accuracies of those decision trees vary from 75.44% to 94.51%, where the decision accuracies for 64×64 CUs are relatively low. The reason is that there are many CUs with pure horizontal edges, pure vertical edges or a single color in the training data set of 64×64 CUs, and they are difficult for classification. However, this kind of CUs can be encoded efficiently by all modes with very low computational complexity. Therefore, the low decision accuracy of 64×64 CUs will not lead to large computational complexity or coding efficiency degradation.

There are two kinds of false classifications for each mode decision tree. One is the missed detection, which is the case that the optimal mode of a CU is not a certain mode, but it is not detected, and the mode is checked for the CU redundantly. Although the missed detection leads to the increase in computational complexity, it does not bring RD performance loss. The other is the incorrect decision, which is the case that the optimal mode of a CU is a certain mode, but the mode is skipped for the CU incorrectly. The incorrect decision leads to RD performance loss because the optimal mode for a CU is skipped. The incorrect decision rate of each decision tree is shown in Table VIII, and we can see that only 3.12% to 13.14% of the mode decision lead to RD performance loss.

D. Decision Tree Constraint

To reduce the RD performance loss caused by skipping the optimal mode for a CU incorrectly, a decision tree constraint technique based on the spatial content correlation is derived for CUs with the size of 8×8 in this sub-section.

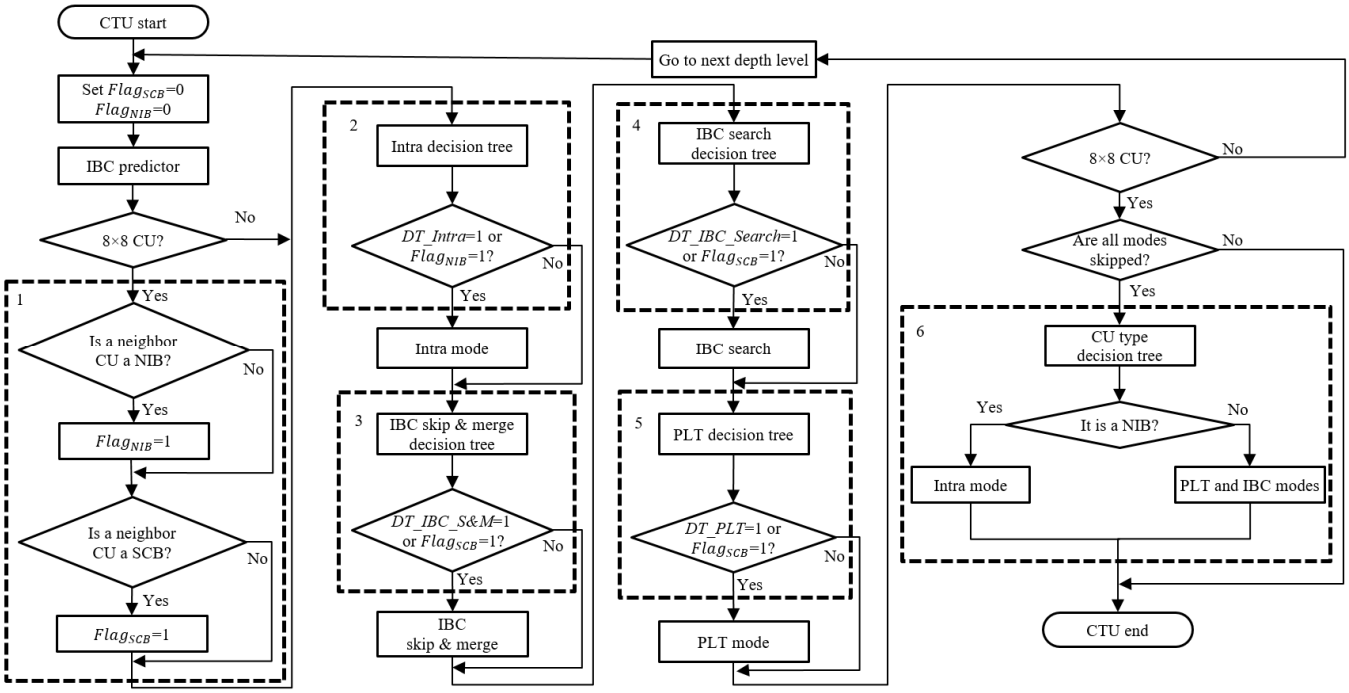
TABLE VIII
INCORRECT DECISION FOR EACH DECISION TREE

CU Size	Intra (%)	IBC		PLT (%)
		Merge & Skip (%)	Search (%)	
64×64	3.81	5.00		
32×32	2.88	3.77		5.33
16×16	4.07	13.14	8.73	5.06
8×8	9.95	6.07	6.75	6.60

Table IX
SAME CONTENT NEIGHBOR CU NUMBER DISTRIBUTIONS FOR 8×8 CUS

CU content	0 (%)	1 (%)	2 (%)
NIB	7.58	18.12	74.30
SCB	4.28	15.08	80.64

There is usually strong spatial content correlation in screen content videos. A CU with the neighbor of NIBs is very likely to be a NIB while a CU with the neighbor of SCBs is very likely to be a SCB. To prove the strong spatial correlation, we encoded the frame-skipped sequences with QPs at 22, 27, 32, and 37 by using the original SCM-8.3 encoder. For each CU, the optimal modes of its top and left neighbor CUs were recorded. We treat a CU selecting Intra mode as a NIB, and a CU selecting IBC mode or PLT mode as a SCB. If a top or left neighbor CU has the same type of content as the current CU, we call it a same content neighbor CU. Table IX shows spatial content correlation of 8×8 CUs by giving the distributions of the same content neighbor CU number. We can see from the table that over 90% CUs have one or two same content neighbor CUs. Only 7.58% of SCBs and 4.28% of NIBs have no same content neighbor CU. Therefore, when encoding an 8×8 CU, if one of its neighbor CUs from the top and left selects Intra mode, i.e. $Flag_{NIB}$=1, we additionally check Intra mode for it based on the outcomes of decision trees, and if one of its neighbor CUs from the top and left selects PLT or IBC mode, i.e. $Flag_{SCB}$=1, we additionally check IBC mode and PLT mode for it based on the outcomes of decision trees. Although there is also strong spatial correction of optimal modes for large CU sizes, it is unnecessary to check more mode candidates for them in order to achieve higher encoding reduction. For a large CU, if decision trees assign an incorrect mode to it, it still has a chance to select good modes when partitioned into 8×8 CUs by using the decision tree constraint technique, so that the RD performance loss brought by the incorrect decision of large CU is decreased. The impact of the decision tree constraint technique will be discussed in Section IV.



Proposed techniques: 1. The decision tree constraint, 2. Intra mode decision tree, 3. IBC merge & skip decision tree, 4. IBC search mode decision tree, 5. PLT mode decision tree, and 6. CU type decision tree.

Fig. 10. Flowchart of the proposed fast mode decision algorithm in a CTU.

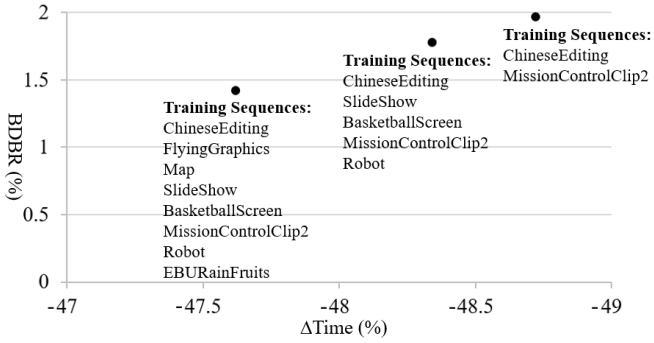


Fig. 11. Simulation results with two, five, eight training sequences.

All proposed techniques are treated as additional mode checking conditions based on the original encoding process when they are implemented in SCM-8.3. As a summary, the flowchart of the proposed fast mode decision algorithm is shown in Fig. 10, where $Flag_{SCB}$, and $Flag_{NIB}$ are used to denote the outcome of the decision tree constrain technique, and *DT_Intra*, *DT_IBC_S&M*, *DT_IBC_Search*, and *DT_PLT* are used to denote the outcomes of the decision trees for Intra, IBC merge & skip, IBC search, and PLT modes, respectively. For simplicity, the original mode checking conditions in SCM-8.3 are not shown in this figure.

IV. Experimental Results and Discussions

To evaluate the performance of the proposed fast mode decision framework, the coding efficiency and computational complexity of the proposed algorithm were compared with those of the original SCM-8.3 and they are measured by BDBR and encoding time increase in percentage (%), ΔTime under AI configuration defined in CTC. Four sets of experiments have been conducted to analyze the performance of the proposed work from different aspects. First, a study on different number of training sequences is discussed. Second, the performance of the proposed framework is evaluated by comparing it with existing fast SCC encoding algorithms. Third, the contribution of each individual mode decision algorithm is analyzed. At last, the efficiency of the feature subset selection and decision tree constraint techniques is validated.

A. Study on Different Training Set

To understand the impact of the training sequences to the performance of the proposed algorithm, we gradually reduce the number of the training sequences and then compare their performances. Fig. 11 shows the simulation results with two, five, eight training sequences, respectively. It is observed that the proposed algorithm can provide relatively good performance even though two training sequences are used, where 48.72% encoding time is reduced with 1.97% increase in BDBR. Besides, it is observed that using more training sequences helps to reduce the increase in BDBR. When training sequences are increased from two to eight, the increase in BDBR is reduced from 1.97% to 1.42%.

B. Performance of the Decision Tree Based Framework

Table X shows the performance of the proposed framework using 8 training sequences in Fig. 11 and four state-of-the-art SCC fast intra prediction algorithms [15]–[18] in terms of BDBR and ΔTime, where the largest value of ΔTime in each sequence is marked in boldface. It is noted that they were implemented in different reference software from ours in their original publications. Zhang *et al.*'s method [15] was simulated using HM-12.1+RExt-5.1 rather than SCM, while Duanmu *et al.*'s method [16], Lei *et al.*'s method [17] and Yang *et al.*'s method [18] were simulated using SCM-4.0, SCM-2.0 and SCM-5.0, respectively. There are numerous enhancements, speed-up techniques and codes clean-up in SCM-8.3 compared

Table X
ΔTime and BDBR OF DIFFERENT ALGORITHMS COMPARED WITH SCM-8.3 UNDER CTC FOR YVU 4:4:4 SEQUENCES

Sequences	Zhang *et al.* [15]		Duanmu *et al.* [16]		Lei *et al.* [17]		Yang *et al.* [18]		Proposed Framework	
	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)
ChineseEditing (*T*)	0.14	-3.40	1.10	-17.47	0.99	-18.96	4.30	-34.16	0.60	**-53.06**
Console (*NT*)	2.64	-8.23	1.87	-28.12	2.87	-23.40	7.38	-42.83	0.60	**-54.14**
Desktop (*NT*)	0.67	-4.94	2.19	-26.24	1.97	-23.85	6.27	-35.91	1.03	**-62.34**
FlyingGraphics (*T*)	0.54	-3.24	0.98	-20.13	1.72	-18.13	5.47	-31.19	1.56	**-52.13**
Map (*T*)	0.97	-10.66	1.55	-19.16	1.23	-20.05	2.84	**-41.66**	1.36	-31.89
Programming (*NT*)	0.44	-11.76	1.89	-22.16	2.50	-22.92	4.71	-27.38	2.20	**-48.94**
SlideShow (*T*)	0.36	-46.92	2.82	-52.47	2.32	**-55.58**	3.69	-34.45	3.76	-35.67
WebBrowsing (*NT*)	0.79	-6.99	1.91	-28.17	6.02	-26.75	5.00	-53.00	0.98	**-57.23**
BasketballScreen (*T*)	0.45	-11.98	1.25	-22.43	1.46	-24.83	3.00	-31.54	1.87	**-48.60**
MissionControlClip2 (*T*)	0.40	-20.5	2.86	-33.9	1.71	-25.49	2.51	-38.54	2.51	**-47.30**
MissionControlClip3 (*NT*)	0.37	-11.28	2.03	-24.61	1.69	-33.81	2.90	-34.15	1.68	**-52.21**
Robot (*T*)	0.43	-17.89	1.18	-29.36	5.21	-46.91	0.59	-28.19	1.51	**-47.19**
EBURainFruits (*T*)	0.21	-18.96	0.88	-26.47	1.76	**-48.58**	0.17	-25.89	0.16	-39.07
Kimono1(*NT*)	0.14	-26.67	1.23	-25.75	1.52	**-75.55**	0.13	-36.18	0.05	-36.93
Average (*NT*)	0.84	-11.65	1.85	-25.84	2.76	-34.38	4.40	-38.24	1.09	-51.97
Average (TGM+M)	0.71	-12.72	1.86	-26.81	2.23	-26.71	4.37	-36.80	1.65	-49.41
Average (A+CC)	0.26	-21.17	1.10	-27.19	2.83	-57.01	0.30	-30.09	0.57	-41.06
Average (ALL)	0.61	-14.53	1.70	-26.89	2.36	-33.20	3.50	-35.36	1.42	-47.62

Sequences with *T* are videos used to generate training frames. Sequences with *NT* are unseen sequences to the trained decision trees.

with the older versions. In the older versions, the BV signal in IBC mode was not unified with the inter mode which only has left and above BVs as predictors with no skip and merge modes. Consequently, incoming CUs always need to check the time-consuming IBC search and PLT modes without early termination. Moreover, N×N IBC search was done after 2N×N search while it is eliminated in SCM-8.3. In addition, the older versions enable PLT mode in the depth level of 0 while it is disabled in SCM-8.3 because of the occasional use. Due to those differences, we re-implemented them into SCM-8.3 for fair comparisons. It is observed that our proposed framework shows the best performance compared with other SCC fast intra prediction algorithms [15]–[18], and it provides the largest encoding time reduction for 10 sequences out of 14 sequences. Compared with the anchor SCM-8.3, our proposed framework achieves up to 62.34% encoding time reduction on the "Desktop" sequence. On average, 47.62% encoding time reduction is obtained with a negligible increase in BDBR of 1.42%. Zhang *et al.*'s method [15] adopts the fast CU size decision framework shown in Fig 4(a), and it is observed in Table X that it only reduces the encoding time by 14.53% on average. Compared with Duanmu *et al.*'s method [16], Lei *et al.*'s method [17] and Yang *et al.*'s method [18] which adopt the hybrid method by combining the frameworks in Fig. 4(a) and (b) for fast CU size decision and fast mode decision based on CU type classification, the proposed framework substantially outperforms them in both coding efficiency and computation complexity. Duanmu *et al.*'s method [16] provides 26.89% encoding time reduction while BDBR is increased by 1.70% on average. When compared with the anchor SCM-8.3, our proposed framework shows 22.60% larger encoding time reduction with 0.21% smaller increase in BDBR than Duanmu *et al.*'s method [16] for sequences in TGM and M. For sequences in A and CC, the proposed framework shows 13.87% larger encoding time reduction with 0.53% smaller increase in BDBR than Duanmu *et al.*'s method [16]. Lei *et al.*'s method [17] achieves 33.20% encoding time reduction while BDBR is increased by 2.36% on average. Although Lei *et al.*'s method [17] shows larger encoding time reduction than the proposed framework for sequences in A and CC, the increase in BDBR is about 4 times higher than the proposed framework. For the sequences in TGM and M, Lei *et al.*'s method [17] also shows a very high increase in BDBR while the encoding time is only reduced by 26.71%. Yang *et al.*'s method [18] shows 35.36% encoding time reduction with a very high increase in BDBR of 3.50% on average. Since it always checks Intra mode for 2N×2N PUs, it brings only 0.30% increase in BDBR to the sequences in A and CC. However, the BDBR of the sequences in TGM and M is increased by 4.37% due to the low decision accuracy for SCBs.

It should be noted that our proposed decision trees were trained by the sequences marked with *T*, where the first frame of each second from these sequences is extracted to generate training data, while the sequences were not used for training are marked with *NT*. It is observed in Table X that our proposed framework provides similar performance for the training sequences and the unseen sequences. Besides, the best performance of our proposed framework is not achieved for the training sequences but for the unseen "Desktop" sequence where 62.34% encoding time is reduced with 1.03% negligible increase in BDBR. Specifically, the average performances of the *NT* sequences are also shown in Table X. The *NT* sequences show 51.97% encoding time reduction with 1.09% increase in BDBR, which outperforms algorithms in [15]–[18]. This shows that the proposed framework is generalizable to the unseen sequences. It is noted that the 14 sequences in CTC [20] are carefully selected to be representatives for other screen content sequences, and all existing fast SCC encoding algorithms always utilize some sequences from CTC [20] for both training and testing. To further show the generalization of the proposed algorithm to other screen content sequences, ten more test sequences [30]–[34] that are not included in CTC [20] were evaluated. The results are shown in Table XI with comparison to the existing fast SCC encoding algorithms [15]–[18], where the largest value of ΔTime in each sequence is marked in boldface. It is observed that the proposed algorithm again outperforms the fast SCC encoding algorithms [15]–[18], and it provides the largest encoding time reduction for seven sequences out of the ten test sequences. Although Lei *et al.*'s method [17] shows larger encoding time reduction in other three sequences, the increase in BDBR is remarkably higher

Table XI
ΔTime and BDBR of Different Algorithms Compared With SCM-8.3 For Sequences not in CTC for YVU 4:4:4 Sequences

Sequences	Zhang *et al.* [15]		Duanmu *et al.* [16]		Lei *et al.* [17]		Yang *et al.* [18]		Proposed Framework	
	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)
BigBuckBunnyStudio	0.64	-20.44	1.90	-31.34	2.58	**-44.83**	1.98	-35.08	1.78	-41.94
ClearTypeSpreadsheet	0.54	-1.14	1.81	-22.83	0.72	-20.86	7.67	-41.76	0.71	**-62.56**
EBULupoCandlelight	0.33	-36.23	1.18	-41.42	3.41	**-66.49**	0.43	-43.05	0.13	-38.35
CadWaveform	1.22	-6.19	6.40	-33.64	4.66	-19.97	4.85	-36.16	0.50	**-58.40**
PcbLayout	0.96	-10.77	2.58	-36.07	3.08	-27.30	4.95	-38.08	1.67	**-48.82**
PptDocXls	1.35	-4.31	1.47	-24.01	1.29	-17.38	4.16	-32.72	0.74	**-55.70**
RealTimeData	3.32	-6.19	1.55	-26.15	2.11	-22.43	7.11	-34.65	0.82	**-50.62**
VideoConferencingDocSharing	0.37	-3.60	1.57	-24.57	3.63	-19.93	7.06	-34.76	0.55	**-58.27**
Viking	0.33	-18.66	1.00	-30.41	5.00	**-65.84**	0.47	-29.61	1.36	-45.02
WordEditing	0.36	-10.30	0.97	-23.17	1.24	-24.42	4.21	-42.10	1.57	**-53.71**
Average (ALL)	0.94	-11.78	2.04	-29.36	2.78	-32.95	4.29	-36.80	0.98	-51.34

Table XII
Average Computational Overheards of the Proposed Algorithm

Computational Overhead	Proportion (%)	
	Feature Extraction	Decision Determination
	1.32	0.01

Table XIII
ΔTime and BDBR of the Proposed Algorithm for YUV 4:2:0 and RGB 4:4:4 Seuqnces Under CTC

Sequence Categories	YUV 4:2:0		RGB 4:4:4	
	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)
Average (TGM+M)	1.66	-37.92	1.58	-48.98
Average (A+CC)	1.79	-59.82	0.80	-55.30
Average (ALL)	1.68	-41.29	1.41	-49.98

than the proposed framework. On average, the fast SCC encoding algorithms [15]–[18] reduce 11.78%–36.80% encoding time with 0.94%–4.29% increase in BDBR. Comparatively, the proposed algorithm reduces 51.34% encoding time with only 0.98% increase in BDBR. Again, this confirms the generalization ability of the proposed algorithm.

The proposed algorithm includes additional processes of feature extraction and decision determination for making fast mode decision, and these computational overheads are further analyzed and summarized in Table XII. It is observed that the average computational overhead proportions of feature extraction and decision determination are only 1.32% and 0.01%, respectively. It is noted that these computational overheads have been counted in Table X to calculate the encoding time reduction.

We also extend our work to support sequences in YUV 4:2:0 and RGB 4:4:4 formats based on the same methodology, and their performances are summarized in Table XIII. It is observed that for sequences in YUV 4:2:0 and RGB 4:4:4 formats, encoding time of 41.68% and 49.98% is reduced with 1.68% and 1.41% increase in BDBR on average, respectively. The results are very similar to that of YUV 4:4:4 sequences, which demonstrates the proposed framework is generalizable to other color formats. Since the fast SCC encoding algorithms [15]–[18] only investigated the fast prediction for YUV 4:4:4 sequences, we cannot make comparisons for sequences in YUV 4:2:0 and RGB 4:4:4 formats.

Since Intra-prediction is also needed in inter frame coding, Fig. 12 also shows the impact of the proposed algorithm on inter frame coding under Low Delay (LD) configuration. BDBR and ΔTime of five typical sequences in YUV 4:4:4 format are shown in Fig. 12, and similar results are observed for other sequences. It is observed that the proposed algorithm reduces 6.52%–8.44% encoding time with negligible increase in BDBR, which implies the proposed algorithm also benefits to inter frame coding.

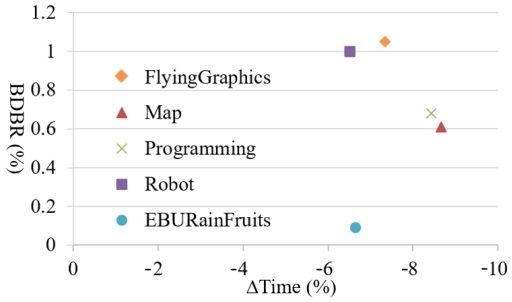


Fig. 12. BDBR and ΔTime of the proposed algorithm under LD configuration.

C. *Performance of the Individual Mode Decision Algorithm*

To further investigate the contribution of each mode decision algorithm, additional experiments were performed by implementing decision trees for IBC mode, PLT mode, IBC+PLT modes, Intra mode, respectively, and the results are shown in Table XIV. We can see from the table that IBC mode decision trees provide the largest encoding time reduction, followed by Intra mode and PLT mode, which are 23.84%, 16.26% and 7.15%, respectively. When the decision trees of IBC mode and PLT mode are both implemented, sequences in A and CC show 39.55% encoding time reduction, which is nearly the same as the results of the overall framework with all decision trees enabled. It is because nearly all CUs in A and CC sequences are encoded by Intra mode, and it shows that the IBC and PLT decision trees can efficiently skip these CUs. Smaller encoding time is saved for sequences in TGM and M because they contain many SCBs, so that fewer IBC mode and PLT mode are skipped. In contrast, only 0.13% encoding time reduction is observed for sequences in A and CC when only the Intra mode decision trees are implemented. For sequences with almost pure SCBs, such as “ChineseEditing”, “Console” and “Desktop”, over 27% encoding time is saved by using Intra mode decision trees. The reason is that almost all CUs in these sequences can skip Intra mode and large encoding time reduction is achieved. Furthermore, Table XIV shows that the overall framework provides 23.52%–34.60% larger encoding time reduction for “ChineseEditing”, “Console” and “Desktop”, as compared with the results that only intra mode decision trees are enabled. The reason is that besides the Intra mode which is not suitable for encoding SCBs, the overall framework considers PLT mode and IBC mode separately and then further skips unnecessary PLT mode and IBC mode for SCBs. To support this statement, we investigated the mode decision of the proposed overall framework by encoding all sequences with QPs of 22, 27, 32, 37, and the average distribution of mode decision is shown in

Table XIV
PERFORMANCE OF EACH INDIVIDUAL MODE DECISION ALGORITHM AND THEIR COMBINATIONS FOR YVU 4:4:4 SEQUENCES

Sequences	IBC Mode Decision		PLT Mode Decision		PLT+IBC Mode Decision		Intra Mode Decision		Overall Framework	
	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)
ChineseEditing (*T*)	0.35	-21.79	0.27	-1.37	0.62	-23.08	0.17	-29.54	0.60	-53.06
Console (*NT*)	0.15	-26.56	0.12	-2.12	0.25	-27.68	0.37	-27.81	0.60	-54.14
Desktop (*NT*)	0.45	-33.40	0.37	-1.58	0.65	-34.64	0.33	-27.74	1.03	-62.34
FlyingGraphics (*T*)	0.31	-26.52	0.11	-2.01	0.45	-27.40	1.21	-24.14	1.56	-52.13
Map (*T*)	0.34	-13.97	0.06	-4.24	0.57	-17.97	0.85	-14.56	1.36	-31.89
Programming (*NT*)	0.87	-21.64	0.93	-6.17	1.88	-27.62	0.37	-20.84	2.20	-48.94
SlideShow (*T*)	2.26	-16.23	0.41	-11.09	3.09	-28.26	0.71	-8.31	3.76	-35.67
WebBrowsing (*NT*)	0.67	-31.58	0.11	-3.83	0.78	-34.83	0.25	-22.56	0.98	-57.23
BasketballScreen (*T*)	1.05	-22.76	0.21	-8.98	1.31	-31.58	0.56	-17.85	1.87	-48.60
MissionControlClip2 (*T*)	1.65	-21.72	0.58	-11.66	2.10	-33.55	0.74	-14.50	2.51	-47.30
MissionControlClip3 (*NT*)	0.91	-24.09	0.30	-8.28	1.04	-31.53	0.67	-19.42	1.68	-52.21
Robot (*T*)	0.57	-23.08	0.67	-17.97	1.38	-43.02	0.12	-3.11	1.51	-47.19
EBURainFruits (*T*)	0.12	-26.12	0	-11.03	0.13	-38.15	0.02	0.22	0.16	-39.07
Kimono1 (*NT*)	0.04	-24.28	0	-9.76	0.04	-37.48	0.04	2.51	0.05	-36.93
Average (*NT*)	0.52	-26.93	0.31	-5.29	0.77	-32.30	0.34	-19.31	1.09	-51.97
Average (TGM+M)	0.82	-23.66	0.32	-5.58	1.16	-28.92	0.57	-20.66	1.65	-49.41
Average (A+CC)	0.24	-24.49	0.22	-12.92	0.52	-39.55	0.06	-0.13	0.57	-41.06
Average (ALL)	0.70	-23.84	0.30	-7.15	1.02	-31.20	0.46	-16.26	1.42	-47.62

Sequences with *T* are videos used to generate training frames. Sequences with *NT* are unseen sequences to the trained decision trees.

Table XV
MODE DECISION DISTRIBUTION OF THE PROPOSED OVERALL ALGORITHM FOR YVU 4:4:4 SEQUENCES

CU Size	Intra only	IBC only	PLT only	Intra+IBC	Intra+PLT	IBC+PLT	Intra+IBC+PLT	No Mode
64×64	27.60	6.70		8.86				56.84
32×32	41.79	1.15	41.52	0.08	4.30	3.674	0.02	7.48
16×16	53.23	3.89	24.31	1.49	3.06	12.18	1.03	0.82
8×8	52.06	0	0	4.01	1.97	27.14	14.81	0

Table XV. It is observed that in the depth level of 3, IBC or PLT mode is always checked with other modes because of the decision tree constrain technique. However, the proposed overall framework is very efficient for larger CU sizes. In the depth level of 0, 56.84% CUs directly go to the next depth level because only Intra mode and IBC mode exist. In the depth levels of 1 and 2, 42.67% and 28.20% CUs select either IBC mode or PLT mode, respectively. By further reducing the mode candidates for SCBs in the proposed overall framework, it provides 20.48%–38.49% larger encoding time reduction for "ChineseEditing", "Console" and "Desktop" compared with [16]–[18], as shown in Table X. Furthermore, by considering IBC mode and PLT mode individually, our proposed framework can also break the upper limit of the fast mode decision methods which only use Intra mode for NIBs and test both IBC mode and PLT mode for SCBs, as illustrated in Table III. This observation shows the advantage of the sequential arrangement of mode decision compared with the fast mode decision frameworks in [16]–[18], which only perform CU type classifications. In the practical application of real video services, parallelizing is an effective way to speed up the encoding process. One concern of the proposed algorithm can be the difficulties in the mode decision parallelizing due to the sequential arrangement of mode decision. However, the proposed algorithm can be integrated with other parallel coding techniques which encode several regions in a frame in parallel to solve this problem, such as the existing tile structures and wavefront parallel processing (WPP) in SCM.

D. Evaluation of the Feature Subset Selection and the Decision Tree Constraint Techniques

To validate the efficiency of the feature subset selection and the decision tree constraint techniques, experiments were performed by implementing the overall framework without the feature subset selection and decision tree constraint techniques,

Table XVI
PERFORMANCES OF THE PROPOSEED ALGORITHM WITH OTHER SETINGS FOR YVU 4:4:4 SEQUENCES

Sequences	Without Feature Subset Selection		Without Decision Tree Constraint	
	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)
ChineseEditing	0.60	-48.06	1.77	-57.46
Console	0.81	-48.06	1.75	-59.07
Desktop	1.00	-58.62	3.06	-66.93
FlyingGraphics	1.68	-42.28	3.35	-55.79
Map	1.52	-28.83	3.38	-40.19
Programming	2.17	-43.10	4.98	-55.51
SlideShow	3.93	-33.84	6.94	-56.91
WebBrowsing	1.25	-57.68	2.17	-60.75
BasketballScreen	1.79	-43.58	4.72	-54.01
MissionControlClip2	3.13	-42.57	4.85	-51.85
MissionControlClip3	1.98	-45.97	3.52	-56.63
Robot	1.61	-45.69	2.11	-48.76
EBURainFruits	0.15	-37.16	0.29	-39.13
Kimono1	0.05	-35.68	0.07	-37.64
Average (TGM+M)	1.81	-44.78	3.68	-55.92
Average (A+CC)	0.60	-39.51	0.82	-41.84
Average (ALL)	1.55	-43.65	3.07	-52.90

and the results are shown in table XVI. Compared with the case without performing feature subset selection, the proposed overall framework in Table XIV provides 3.97% larger encoding time reduction with 0.13% decrease in BDBR. Therefore, better performance is provided by adopting the feature subset selection technique, because the impact of irrelevant or redundant features is removed. Besides, it is observed that the decision tree constraint technique helps to reduce BDBR increase of the proposed framework at the cost of less encoding time reduction. On average, the encoding time saving of the proposed framework is slightly reduced from 52.90% to 47.62% while the increase in BDBR is reduced from 3.07% to 1.42% by implementing the decision tree constraint technique. Specifically, we can see that the performance improvement for sequences in A and CC is limited, but sequences in TGM and M gain large benefits from the decision

tree constraint technique and the increase in BDBR is reduced by 2.03%. The reason is that IBC and PLT modes are very effective in 8×8 SCBs. Therefore, even small incorrect decision rate of IBC and PLT modes in 8×8 SCBs leads to large RD performance loss. By using the decision tree constraint technique, additional mode candidates are available at the last depth level (depth level of 3), and the RD performance loss brought by the incorrect decision is reduced effectively.

V. Conclusion and Future Work

In this paper, a machine learning based fast mode decision framework is proposed for SCC. To avoid the exhaustive mode searching process, a flexible intra mode decision framework is proposed by utilizing a sequential arrangement of mode classifiers. Compared with the traditional methods that IBC and PLT modes are both checked for SCBs, we insert a decision tree before checking each mode with the help of new dynamic features, so that the decision of each mode is made separately, and it allows the case that only mode is checked for a SCB. Experiments results have shown that the proposed framework can provide an average computational complexity reduction of 47.62% with a negligible increase in BDBR of 1.42%. Future works may include fast SCC encoding algorithms based on CNNs, which is a powerful tool in many classification problems Nevertheless, the drawback CNNs is the high computational complexity brought by convolutional operation in the test phase, and it might be solved by increasing the stride size and designing a multi-output CNN that makes predictions for multiple CUs in one test. However, this paper can be treated as the baseline for CNN approaches in the future.

References

[1] Y. Lu, S. Li, and H. Shen, "Virtualized screen: A third element for cloud-mobile convergence," *IEEE Multimedia*, vol. 18, no. 2, pp. 4–11, Feb. 2011.

[2] G. J. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[3] J. Xu, R. Joshi, and R. A. Cohen, "Overview of the emerging HEVC screen content coding extension," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 1, pp. 50–62, Jan. 2016.

[4] X. Xu *et al.*, "Intra block copy in HEVC screen content coding extensions", *IEEE J. Emerg. Sel. Topic Circuits Syst.*, vol. 6, no. 4, pp. 409–419, Dec. 2016.

[5] Z. Ma, W. Wang, M. Xu, and H. Yu, "Advanced screen content coding using color table and index map", *IEEE Trans. Image Process.*, vol. 23, no. 10, pp. 4399–4412, Oct. 2014.

[6] S.-H. Tsang, Y.-L. Chan, and W.-C. Siu, "Exploiting inter-layer correlations in scalable HEVC for the support of screen content videos," in *Proc. 19th Int. Conf. Digital Signal Process*, Hong Kong, China, Aug. 2016, pp.1–5.

[7] H.-S. Kim, and R.-H. Park, "Fast CU partitioning algorithm for HEVC using an online-learning-based bayesian decision rule," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 1, pp. 130–138, Jan. 2016.

[8] S-H. Jung, and H.W. Park, "a fast mode decision method in HEVC using adaptive ordering of modes," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 10, pp. 1846–1858, Oct. 2016.

[9] Q. Hu, X.-Y. Zhang, Z.-R. Shi, and Z.-Y. Gao, "Neyman-Pearson-based early mode decision for HEVC encoding," *IEEE Trans. Multimedia*, vol. 18, no. 3, pp.379–391, Mar. 2016.

[10] S.-H. Tsang, Y.-L. Chan, and W.-C. Siu, "Fast and efficient intra coding techniques for smooth regions in screen content coding based on boundary prediction samples," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Brisbane, Australia, Apr. 2015, pp.1409–1413.

[11] S.-H. Tsang, Y.-L. Chan, and W.-C. Siu, "Hash based fast local search for intra block copy (IntraBC) mode in HEVC screen content coding," in *Proc. APSIPA ASC*, Hong Kong, Dec. 2015, pp. 396–400.

[12] S.-H. Tsang, W. Kuang, Y.-L. Chan and W.-C. Siu, "Fast HEVC screen content coding by skipping unnecessary checking of intra block copy mode based on CU activity and gradient," in *Proc. APSIPA ASC*, Jeju, Korea, Dec. 2016, pp.1–5.

[13] F. Duanmu, Z. Ma, and Y. Wang, "Fast CU partition decision using machine learning for screen content compression," in *Proc. IEEE Int. Conf. Image Process.*, Quebec, QC, Canada, Sep. 2015, pp. 4972–4976.

[14] H. Zhang, Q. Zhou, N.-N Shi, F. Yang, X. Feng, and Z. Ma, "Fast intra mode decision and block matching for HEVC screen content compression," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Shanghai, China, Mar. 2016, pp.1377–1381.

[15] M. Zhang, Y. Guo, and H. Bai, "Fast intra partition algorithm for HEVC screen content coding," in *Proc. IEEE Vis. Commun. Image Process.*, Valletta, Malta, Dec. 2014, pp. 390–393.

[16] F. Duanmu, Z. Ma, and Y. Wang, "Fast mode and partition decision using machine learning for intra-frame coding in HEVC screen content coding extension," *IEEE J. Emerg. Sel. Topic Circuits Syst.*, vol. 6, no. 4, pp.517–531, Dec. 2016.

[17] J. Lei, D. Li, Z, Pan, Z. Sun, S. Kwong, and C. Hou, "Fast intra prediction based on content property analysis for low complexity HEVC-based screen content coding," *IEEE Trans. Broadcast.*, vol. 63, no.1, pp.48–58, Mar. 2017.

[18] H. Yang, L. Shen, and P. An, "An efficient intra coding algorithm based on statistical learning for screen content coding", in *Proc. IEEE Int. Conf. Image Process.*, Beijing, China, Sep. 2017, pp. 2468–2472.

[19] HM-16.12+SCM-8.3, HEVC test model version 16.12 screen content model version 8.3, [Online], available at: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.12+SCM-8.3/.

[20] H. -P. Yu, R. Cohen, K. Rapaka, and J. -Z Xu, "Common test conditions for screen content coding", 24th JCT-VC meeting, document JCTVC-X1015-r1, Geneva, Switzerland, May. 2016.

[21] G. Bjontegaard, "Calculation of average PSNR differences between rd-curves," document VCEG-M33, VCEG, Austin, Texas, USA, Mar. 2001.

[22] Machine Learning Based Fast Intra Mode Decision for HEVC Screen Content Coding Via Decision Trees. [Online]. Available at: http://www.eie.polyu.edu.hk/~ylchan/research/DT-FastSCC/.

[23] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann, 1993.

[24] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: An update," *ACM SIGKDD Explorations Newslett.*, vol. 11, no. 1, pp. 10–18, 2009.

[25] Y. Mansour, "Pessimistic decision tree pruning based on tree size," in *Proc. Int. Conf.* Machine Learning, pp. 195–201, 1997.

[26] P. Burman, "A comparative study of ordinary cross-validation, v-fold cross-validation and the repeated learning-testing methods," *Biometrika*, vol. 76, no. 3, pp. 503-514, Sep. 1989.

[27] Z. Pan, H. Shen, Y. Lu, S. Li, and N. Yu, "A low-complexity screen compression scheme for interactive screen sharing," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 6, pp. 949–960, Jun. 2013.

[28] N. Japkowicz, "The class imbalance problem: Significance and strategies," in *Proc. Int. Conf. Artif. Intell.*, 2000, pp. 111–117.

[29] I. Guyon, and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Mar. 2003.

[30] J. Guo, L. Zhao, and T. Lin, "Response to B1002 Call for test materials: Five test sequences for screen content video coding", 3th JVET meeting, document JVET-C0044, Geneva, Switzerland, May. 2016.

[31] R. Cohen, "AHG8: 4:4:4 game content sequences for HEVC range extensions development", 14th JCT-VC meeting, document JCTVC-N0294, Vienna, Austria, Aug. 2013.

[32] A. M. Tourapis, D. Singer, and K. Kolarov, "New test sequences for screen content coding", 15th JCT-VC meeting, document JCTVC-O0222, Geneva, Switzerland, Nov. 2013.

[33] H. -P. Yu, W. Wang, X. Wang, J. Ye, and Z. Ma, "AHG8: New 4:4:4 test sequences with screen content", 15th JCT-VC meeting, document JCTVC- O0256, Geneva, Switzerland, Nov. 2013.

[34] K. Sharman, and K. Suehring, "Common test conditions", 24th JCT-VC meeting, document JCTVC-X1100, Geneva, Switzerland, May. 2016.

Wei Kuang (S'17) received the B. S. degree in School of Electronic and Optical Engineering from Nanjing University of Science and Technology, Nanjing, China, in 2015. Now, he is currently pursuing the Ph.D. Degree in the Department of Electronic and Information Engineering at The Hong Kong Polytechnic University. His research interests include machine learning and deep learning in video coding and video transcoding. He serves as a reviewer of international journals including the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY and KSII TRANSACTIONS ON INTERNET AND INFORMATION SYSTEMS.

Yui-Lam Chan (S'94-A'97-M'00) received the B.Eng. (Hons.) and Ph.D. degrees from The Hong Kong Polytechnic University, Hong Kong, in 1993 and 1997, respectively.

He joined The Hong Kong Polytechnic University in 1997, where he is currently an Associate Professor with the Department of Electronic and Information Engineering. He is actively involved in professional activities. He has authored over 120 research papers in various international journals and conferences. His research interests include multimedia technologies, signal processing, image and video compression, video streaming, video transcoding, video conferencing, digital TV/HDTV, 3DTV/3DV, multiview video coding, machine learning for video coding, and future video coding standards including screen content coding, light-field video coding, and 360-degree omnidirectional video coding.

Dr. Chan serves as an Associate Editor of IEEE TRANSACTIONS ON IMAGE PROCESSING. He was the Secretary of the 2010 IEEE International Conference on Image Processing. He was also the Special Sessions Co-Chair and the Publicity Co-Chair of the 2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, and the Technical Program Co-Chair of the 2014 International Conference on Digital Signal Processing.

Sik-Ho Tsang (M'10) received the Ph.D. degree from The Hong Kong Polytechnic University (PolyU), Hong Kong, in 2013.

He was a Postdoctoral Fellow from 2013 to 2016, and involved numerous industrial projects for video coding and transcoding. He is currently a Research Fellow in PolyU. He has authored numerous international journals, conferences and patents. His current research fields involve video coding such as HEVC, VVC, multiview video plus depth coding, screen content coding, and immersive video coding including light field coding and 360-degree video coding. His research interests also includes machine learning and deep learning.

He serves as a reviewer of international journals including the IEEE TRANSACTIONS ON IMAGE PROCESSING, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, and Elsevier Journal of Signal Processing: Image Communication.

Wan-Chi Siu (S'77-M'77-SM'90-F'12-Life-F'16) received the MPhil and PhD degrees from The Chinese University of Hong Kong in 1977 and Imperial College London in 1984. He is Life-Fellow of IEEE and Fellow of IET, and Immediate-Past President (2019-2020) of APSIPA (Asia-Pacific Signal and Information Processing Association). Prof. Siu is now Emeritus Professor, and was Chair Professor, Director of Signal Processing Research Centre, Head of Electronic and Information Engineering Department and Dean of Engineering Faculty of The Hong Kong Polytechnic University. He is an expert in DSP, fast algorithms, super-resolution imaging, 2D and 3D video coding, and machine learning for object recognition and tracking. He has published 500 research papers (over 200 are international journal papers) and has 9 recent patents granted. Prof. Siu was also an independent non-executive director (2000-2015) of a publicly-listed video surveillance company and convenor of the First Engineering/IT Panel of the RAE(1992/93) in Hong Kong. He is an outstanding scholar, with many awards, including the Best Faculty Researcher Award (twice) and IEEE Third Millennium Medal (2000). Prof. Siu has been Guest Editor/Subject Editor/AE for IEEE Transactions on Circuits and System II, Image Processing, Circuit & System for Video Technology, and Electronics Letters, and organized very successfully over 20 international conferences including IEEE society-sponsored flagship conferences, such as TPC Chair of ISCAS1997 and General Chair of ICASSP2003 and General Chair of ICIP2010. He was Vice-President, Chair of Conference Board and Core Member of Board of Governors (2012-2014) of the IEEE Signal Processing Society, and is now a member of IEEE Fourier Award for Signal Processing Committee and some other IEEE committees.