IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

iCUS: Intelligent CU Size Selection for HEVC Inter Prediction

BUDDHIPRABHA ERABADDA[1] (Student Member, IEEE), THANUJA MALLIKARACHCHI[2] (MEMBER, IEEE), GOSALA KULUPANA[1], AND ANIL FERNANDO[1] (Senior Member, IEEE)

[1]Centre for Vision Speech and Signal Processing, University of Surrey, Surrey, GU2 7XH, United Kingdom (e-mails: {e.buddhiprabha, g.kulupana, w.fernando}@surrey.ac.uk)

[2]Cardiff School of Technologies, Cardiff Metropolitan University, Llandaff Campus, Western Avenue, Cardiff, CF5 2YB, UK.(e-mail: tmallikarachchi@cardiffmet.ac.uk)

Corresponding author: Buddhiprabha Erabadda (e-mail: e.buddhiprabha@ surrey.ac.uk).

This work was supported by the CONTENT4ALL project, which is funded under European Commission's H2020 Framework Program (Grant number: 762021).

ABSTRACT The hierarchical quadtree partitioning of Coding Tree Units (CTU) is one of the striking features in HEVC that contributes towards its superior coding performance over its predecessors. However, the brute force evaluation of the quadtree hierarchy using the Rate-Distortion (RD) optimisation, to determine the best partitioning structure for a given content, makes it one of the most time-consuming operations in HEVC encoding. In this context, this paper proposes an intelligent fast Coding Unit (CU) size selection algorithm to expedite the encoding process of HEVC inter-prediction. The proposed algorithm introduces (i) two CU split likelihood modelling and classification approaches using Support Vector Machines (SVM) and Bayesian probabilistic models, and (ii) a fast CU selection algorithm that makes use of both offline trained SVMs and online trained Bayesian probabilistic models. Finally, (iii) a computational complexity to coding efficiency trade-off mechanism is introduced to flexibly control the algorithm to suit different encoding requirements. The experimental results of the proposed algorithm demonstrate an average encoding time reduction performance of 53.46%, 61.15%, and 58.15% for *Low Delay B*, *Random Access*, and *Low Delay P* configurations, respectively, with Bjøntegaard Delta-Bit Rate (BD-BR) losses of 2.35%, 2.9%, and 2.35%, respectively, when evaluated across a wide range of content types and quality levels.

INDEX TERMS Coding Unit (CU), encoder complexity reduction, High Efficiency Video Coding (HEVC), inter-prediction, Support Vector Machine (SVM)

RECENT advancements in multimedia technologies that span across Consumer Electronics (CE) in video content capturing, transmission and display have made video data the most frequently exchanged type of content over the modern communication networks. The increasing mobile consumption of High Definition (HD) and Ultra High Definition (UHD) video contents has contributed immensely towards the ever-growing IP video traffic and it is expected to reach over 82% of the overall Internet traffic in 2021 [1]. However, the estimated growth in network bandwidth (1.9 fold from 2017-2022, which is 39.0 Mbps to 75.4 Mbps for fixed broadband [1]) with time is still not sufficient to cater for the ever-growing user demands. Furthermore, the video requirements for emerging applications such as Augmented Reality (AR)/Virtual Reality (VR), interactive television, multi-party video conferences and over-the-top (OTT) multimedia consumption demand continuous improvements in the compression efficiency [2].

In this regard, High Efficiency Video Coding (HEVC) which was introduced in 2013 is the most recent stable video coding standard. It provides greater compression efficiency through an assortment of new features and coding tools over its predecessor H.264/AVC [3]. Out of these, the hierarchical quadtree partitioning structure introduced in HEVC that entails a wide range of Coding Unit (CU) sizes (i.e., 8×8 to 64×64) and their combinations, is one of the important contributors towards HEVC's improved coding performance. However, at the same time, it is also a major source of the complexity within the HEVC architecture [4], [5]. In addition, the brute-force Rate-Distortion (RD) optimisation followed by the encoder to determine the best coding mode configuration and quadtree partitioning structure for a given

content, is considered as a main reason for the increased encoding complexity. For example, a mere increase of maximum CU size from 16×16 to 64×64 results in an average encoding time increase of 43% [4]. Hence, it is often identified that the complexity of the coding tools introduced in HEVC would require significant improvements to make them operational in real-time [5]. This is further corroborated in the recent efforts by Motion Pictures Expert Group (MPEG) to standardise several approaches for low complexity video encoding illustrating the importance of making the video coding algorithms less complex and less resource-intensive [6].

The recent literature identifies numerous mechanisms to reduce the HEVC encoding complexity by reducing the time taken by the RD optimisation to select the optimal coding modes and quadtree structure (in this case CU size) for a given video content [7]. The state-of-the-art fast encoding methods in this domain can be broadly categorised into two main approaches; rule-based methods and learning-based methods. Rule-based methods typically utilise the depth correlation of spatial and temporal blocks, RD cost statistics of previous CUs, and prediction modes [8], [9] to generate a fixed set of rules to determine the optimal CU size for a given content. However, vast differences in video characteristics and the dynamic nature of the video contents make it difficult to maintain a fixed set of rigid rules to determine the optimal CU sizes for particular video content. On the other hand, learning-based approaches often utilise machine learning methods with pre-trained models generated from vast amounts of data from previously encoded sequences [10]–[14]. Yet, the use of fixed rigid models makes them less adaptive to the changing properties (i.e., texture and motion characteristics) of natural video sequences. Algorithms that use online training [15], [16] facilitate generating content-adaptive dynamic models, but at the same time suffer from lack of training data for certain features making the decisions unreliable. Therefore, it is highly beneficial to investigate dynamic and flexible encoding algorithms that make use of the advantages of both offline and online trained prediction models to reduce the computational complexity of HEVC encoding while retaining the coding efficiency intact.

To this end, this paper proposes iCUS; an intelligent CU split decision making algorithm, that takes a hybrid approach by utilising offline-trained support vector machine (SVM) models together with Bayesian statistical models that track the probability of occurrence of features in the current video being encoded. The experimental results reveal that the proposed algorithm achieves a significant encoding time reduction performance compared to the HM16.8 [17] implementation and the state-of-the-art algorithms, with a minimal impact on the coding efficiency, for a variety of content types.

The remainder of this paper is organised as follows. Section I provides an overview of the HEVC block partitioning, CU size selection process, and the existing work in the literature on computational complexity reduction of HEVC

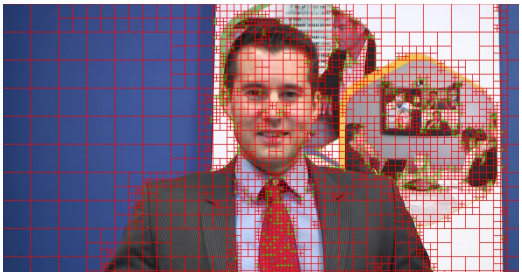
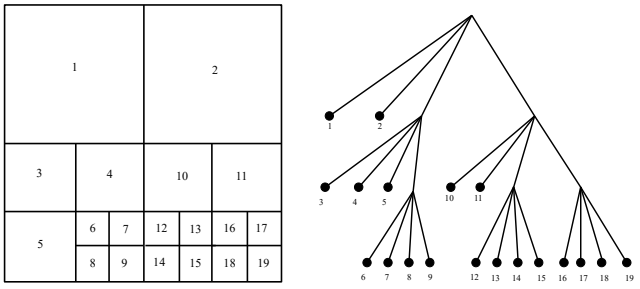


FIGURE 1. An example CU partitioning structure for a frame in the "Johnny" sequence.



(a) CTU partitioning (b) Quadtree structure

FIGURE 2. Sample partitioning structure of a CTU and the corresponding quadtree structure.

inter-prediction. The CU split likelihood modellings using SVMs and probabilistic models are discussed in Sections II and III, respectively. The proposed encoding complexity reduction algorithm that uses both SVM and probabilistic models is described in Section IV. Experimental results are discussed in Section V and finally, Section VI concludes and discusses the potential for future improvements.

I. BACKGROUND AND RELATED WORK

This section first provides an overview of the hierarchical quadtree partitioning employed in the block-based HEVC encoding architecture. Next, state-of-the-art methods which focus on reducing the resulting encoding complexity due to the brute-force RD optimisation are discussed.

A. BACKGROUND

HEVC utilises a block-based partitioning structure to determine the encoding parameters for a given content. In this case, a frame from a sequence is initially divided into blocks called Coding Tree Units (CTUs), which are the basic units of partitioning in HEVC. A CTU can have a maximum size of 64×64 and it can be recursively sub-divided into four equally-sized smaller blocks called Coding Units (CUs) that can have sizes varying from 64×64 to 8×8. As an example, Fig. 1 shows the partitioning structure for a frame from the *Johnny* video sequence, where the frame is divided into CTU and CU blocks. Fig. 2(a) depicts how a CTU can be recursively sub-divided into CUs, and the corresponding quadtree structure of the CTU is depicted in Fig. 2(b). A CU can contain one or more Prediction Units (PUs) and Transform Units (TUs) that contain prediction

and transform information, respectively [4]. In general, an HEVC compatible encoder goes through a process of evaluating all possible combinations of the quadtree structure to determine the coding parameter combination that achieves the best coding efficiency performance for a given content. In the HM [17] reference encoder, this is achieved by an RD optimisation process and obtaining the coding parameter combination that gives the minimum RD cost computed by evaluating the Lagrangian optimisation cost function given by,

$$p^* = arg \min_{\forall p \in \mathcal{A}} \Big\{ D(p) + \lambda R(p) \Big\} \tag{1}$$

where $\lambda \geq 0$, and p^* denote the Lagrange multiplier and the optimal coding parameters from the set of all coding options ($\mathcal{A}$) considered for the minimisation, respectively. The terms $D(p)$ and $R(p)$ denote the distortion and the rate associated with the p set of coding parameters, respectively [18]. This brute-force approach of evaluating all possible combinations of coding parameters when determining the optimal coding structure enables HEVC to achieve very high coding efficiency. However, it adds an immense computational complexity burden to the encoder, resulting in excessive encoding time and energy consumption. As a result, numerous algorithms are proposed in the recent literature to reduce the encoding time complexity of HEVC while keeping the coding efficiency unscathed. Detailed encoder profiling results suggest that a large proportion of the overall encoding time is spent at the CU level to determine the best CU structure for a given content [5]. Hence, the following subsection analyses some of these fast encoding algorithms that are aimed at minimising the encoding time spent at the CU size selection stage within the encoding pipeline.

B. RELATED WORK

In HEVC, inter-prediction accounts for the highest portion of the encoding time [19]. The encoding time complexity in HEVC inter-prediction can be reduced by introducing changes to a range of operations within the HEVC architecture. For example, optimisation attempts in motion estimation, in-loop filtering, and coding structure selection processes are reported in the recent literature. The complexity profiling results in [5] and recent literature report that the encoding time gain that can be achieved from the optimisation of motion estimation and loop filtering processes is minimal. This is further corroborated by the experimental results presented in [20] and [21] for their fast motion estimation methods that report $\approx 20\%$ encoding time reductions. Similarly, fast Sample Adaptive Offset (SAO) parameter estimation algorithms proposed in [22] and [23] are limited to average encoding time gains that are in the range of $42\% - 45\%$. On the other hand, recent literature reports that expediting the HEVC quadtree size selection process can lead to much higher overall encoding time reductions. In addition, such fast motion estimation and in-loop filtering algorithms are not mutually exclusive and can be utilised in conjunction with fast coding structure selection algorithms, resulting in much higher overall encoding time reductions. The algorithm proposed in this paper falls into the category of fast CU size selection methods, thus, the following subsection mainly discusses the state-of-the-art work that focuses on reducing the encoding time complexity of the CU size selection process within the HEVC architecture.

The state-of-the-art work that targets fast CU size selection can be broadly categorised into two areas; rule-based approaches and learning-based approaches. The following subsections provide summaries of the related work that fall into each of these two categories.

1) Rule-based methods

In general, rule-based methods utilise statistical inferences based on the video content and data gathered during the encoding process to generate a fixed set of rules to determine the best CU size for a given content. In this context, the use of features extracted from the encoding loop such as RD cost, Skip mode, and Merge mode, is one of the popular approaches in the recent literature. For instance, Lee *et al.* [24] use previous RD cost details of the inter $2N \times 2N$ PU mode together with Skip and Merge modes to estimate the CU size decision of the current CU. The algorithm, however, demonstrates a considerable variance in the encoding time reduction with the Quantisation Parameter (QP) and content type. For example, encoding complexity reduction achievable is minimal with highly textured and complex sequences, especially at lower QPs, where Skip mode is less significant. Similar behaviour is observed in the algorithm proposed by Vanne *et al.* [25], which skips the evaluation of certain PU modes depending on the Skip mode decisions at the current CU depth level. The algorithm proposed by Choi *et al.* [26] follows a similar approach, but achieves a limited encoding time reduction compared to similar approaches.

HEVC reference encoder implementation (i.e., HM Test Model 16.8 [17]) comes with built-in fast encoding options that also utilise encoding information extracted within the encoding loop. These include Early CU (ECU) termination [27], Early Skip Detection (ESD) [28], and Coding Flag Mode (CFM) [29]. ECU approach terminates further recursive splitting of a CU, if the best mode for the current CU depth is determined to be the Skip mode. In the ESD method, Skip mode detection is further expedited if the motion vector difference and coded block flag of Inter $2N \times 2N$ are identified as zeros. In CFM, when the coding block flag of the current PU is zero, the subsequent PU mode evaluations are bypassed. Deficiencies arising from the rigidness of the decision rules are visible in these algorithms when encoding highly complex video sequences.

Characteristics of the content being encoded are also often used as features in the algorithms that estimate the CU size and coding structures for a given video content prior to the encoding process. For instance, Pyramid Motion Divergence (PMD) features calculated from the optical flow of downsampled frames are used in [30] to early determine the CU

size. However, computation of optical flow itself within the encoding loop is considered computationally expensive and resource-intensive. Similar research is presented in [31], [32] that use motion data for early CU/PU decisions.

Following a different approach, Shen *et al.* [8] utilise previously encoded neighbouring and co-located CU information to skip the evaluation of unnecessary CU depth levels of the current CU. Such algorithms demonstrate significant degradation in the RD performance due to sub-optimal decisions and subsequent error propagation, especially in the case of video contents with complex textures and high motions [15], [33].

Similar methods that use statistical models generated from data within the encoding loop [8], [9], commonly rely on rigid and less flexible rule-based models, which cannot adapt to the dynamic changes of the video content. That being said, Mallikarachchi *et al.* [15] propose a content-adaptive CU size selection algorithm that utilises two CU classification models generated from the data collected online during the encoding phase. These, together with the moving window based feature selection process, ensure that the CU decisions obtained are relevant and adaptive to the content being encoded. However, the lack of data points for certain features within the limited window size makes certain CU split decisions unreliable, leading to coding losses.

2) Learning-based methods

On the other hand, learning-based methods utilise machine learning approaches to build models and to tune hyper-parameters based on historical data collected during the video encoding process. SVM-based methods are commonly used in the literature for CU split decision prediction applications. For example, Grellert *et al.* [10] utilise features from the current CU depth level to construct SVM models to determine whether to early-stop the recursive CU split process. However, the method shows deficiencies for complex sequences encoded at low QPs where high CU depths are common. In this context, the evaluation of current CU depth level becomes ineffectual if the CU is decided to split further, which is often the case with complex sequences encoded at low QPs. The complexity reduction method proposed by Shen and Yu [12] makes use of Inter $2N \times 2N$ RD cost and Skip/Merge data as features for the offline-trained SVM models. This method relies on the coding information of the current depth level. Hence, the time gains are hindered when the CU is decided to split, due to the redundant calculations at the current CU depth level.

The use of decision trees for CU size selection is considered in the algorithm proposed by Correa *et al.* [13]. In addition, Xu *et al.* [14] propose a Convolutional Neural Network (CNN) based approach for HEVC complexity reduction. However, the rigidness of the models in these approaches is seen as a common drawback which makes the CU decisions less adaptable for dynamics of the video content. Li *et al.* [34] also propose a complexity reduction algorithm using CNNs, yet, the anticipated time gains are relatively lower in this approach. CNNs are widely used in HEVC encoding to reduce the complexity of intra-prediction operations (e.g. [35], [36]). Nevertheless, these methods only rely on spatial information, which becomes less useful in the context of inter-prediction that operates in the temporal domain.

II. CU CLASSIFICATION FOR SPLIT LIKELIHOOD USING SVM

This section first introduces the CU split likelihood modelling followed by a description of iCUS, the SVM based CU classification model proposed in this paper.

A. CU SPLIT LIKELIHOOD MODELLING

In HEVC, the basic processing unit is a CTU, which has a maximum size of 64×64. During the encoding process, any frame in the sequence is initially divided into CTUs. Subsequently, each CTU is recursively sub-divided into four equal-sized blocks called CUs, that have sizes varying from 64×64 to 8×8, also identified as CU depth levels from 0 to 3, respectively. During the RD optimisation process, all possible PU modes for a given CU size are evaluated to determine the combination of partitioning structures and prediction modes that gives the best RD cost. Therefore, the final partitioning structure of a CTU is determined by the RD costs at each CU depth level. For a given CU depth level (say *i*), if the RD cost at depth *i* is higher than that of the depth *i*+1, the CU is marked as *split* and it is marked as *non-split* otherwise. Therefore, the decision for sub-division at each CU depth level can be modelled as a binary classification problem, with classes $y \in \{+1, -1\}$ where $y = +1$ represents a *split* and $y = -1$ represents a *non-split* decision. This can be given by,

$$y = \begin{cases} +1 & C_0 > \sum_{s=1}^{s=4} C_s \\ -1 & \text{otherwise} \end{cases}, \tag{2}$$

where C_0 and C_s ($s \in 1, 2, 3, 4$) denote the RD cost of the current CU in the $2N \times 2N$ mode and the RD cost for a sub CU, respectively.

Binary classification problems can be modelled using various machine learning techniques such as Naïve Bayes [37], decision trees [38], neural networks [39], SVMs [40] and logistic regression [41]. However, the time taken by the model's inference process to predict the split decisions is very crucial to achieve higher encoding time complexity reductions. Hence, in this paper, the classification problem in (2) is modelled using SVMs due to their ability in handling binary classification problems with significant computational advantages [42].

1) Support Vector Machines (SVM)

SVMs are a supervised machine learning algorithm that is heavily used in classification, regression, and outlier detection problems. They first determine a separating hyperplane for the training data that maximises the separation among

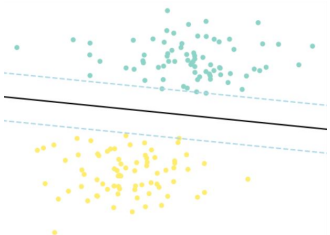


FIGURE 3. Depiction of SVM hyperplane with sample data. The black line represents the optimal hyperplane that gives the maximum possible margin between the two classes (denoted by two different colours).

classes, known as the margin. Once the hyperplane is established, SVMs can predict the class label for a given feature vector from unseen data depending on the location of the new data item, relative to the hyperplane. For example, a training dataset T_r of size n where there are two classes with labels $+1$ and -1 can be represented as,

$$T_r = \{(x_i, y_i) \quad | \quad x_i \in \mathbb{R}^p, \quad y_i \in \{-1, +1\}\}_{i=1}^{n}. \quad (3)$$

Here, x_i is a p-dimensional input vector, and y_i is the corresponding class to which each x_i belongs. During the training phase, SVM constructs a hyperplane that represents the largest separation of the dataset (known as the margin) in a higher-dimensional space, solving the primal optimisation function defined as,

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2} w^t w + C \sum_{i=1}^{N} \xi_i \\ \text{s.t.} \quad & y_i(w\phi(x_i + b)) + \xi_i - 1 \\ & \xi \geq 0, \quad i = 1, ..., n, \end{aligned} \quad (4)$$

where w, ξ and $C > 0$, are the normal vector to the hyperplane, slack variable and regularisation parameter, respectively. Here, $\phi(x_i)$ acts as the Kernel function that can map x_i into a higher-dimensional space in the case of non-linear boundaries among the classes. Once the hyperplane is established, the classifications for unseen samples are obtained using the decision function,

$$\text{sgn}(w^T \phi(x) + b). \quad (5)$$

Fig. 3 graphically illustrates a sample dataset with two classes and the corresponding optimal hyperplane (in *black*) that achieves the maximum possible margins with the data points of the two classes.

2) Feature Selection

To use SVMs as the CU classification model, it is crucial to scrutinise the HEVC encoding process to identify the features that closely contribute to the CU split decision. Besides, it is important to identify and construct a dataset that is required for the training phase of the models. In this context, numerous features and parameters that can be extracted from the encoding loop per CU depth are considered, and their relative importance towards the CU split decision is analysed.

TABLE 1. Training Sequences

Sequence	Class	Resolution
Traffic	A	2560×1600
Kimono	B	1920×1080
BQMall	C	832×480
BlowingBubbles	D	416×240

TABLE 2. Features Analysed for CU Split Likelihood Modelling

Type	Feature
Context Information	Neighbouring CTU depths
	Co-located CU depths
	Encodig information of Neighbouring CTUs
Texture Information	Homogeneity
	Contrast
	Energy
	Local range values
Current CU Information	Quantisation Parameter (QP)
	$2N \times 2N$ coding information

The feature selection is carried out using a set of video sequences that is listed in Table 1. These sequences are carefully selected to represent a wide range of characteristics in video contents such as different texture complexities and motion details. Twenty frames from each sequence are encoded under fixed QP settings such that the $QP \in \{22, 27, 32, 37\}$, using the HM16.8 reference software [17]. Then, a set of features that encompass context, texture, and coding information of the current CU are extracted along with the CU split decision under RD optimisation to form the training dataset. Table 2 illustrates the set of features considered during the feature extraction step. The relative importance of subsets of these features to the CU split decision has been identified in the literature as well as in our previous work [10], [13], [15]. A brief overview of the features summarised in Table 2 is given below.

a: Context Information

The context information defines the features that can be extracted from the previously encoded neighbouring and co-located CTUs of the current CTU. For each neighbouring and co-located CTU, the information extracted as the maximum PU depth, average PU depth, number of bits consumed for the block, RD cost, and resulted distortion in the block, are considered as the dominant features.

b: Texture Information

Texture information for the CTU is extracted from the raw pixel values of the current encoding block. In this case, statistical information of the image block extracted by computing the Grey Level Co-occurrence Matrix (GLCM) [43] is used. GLCM is defined over an image as a matrix that represents the distribution of co-occurring pixel values. For example, for a $m \times n$ image with p distinct pixel values, the co-occurrence

matrix C can be given as,

$$C_{\Delta x,\Delta y}(i,j) = \sum_{x=1}^{n}\sum_{y=1}^{m} \begin{cases} 1, & I(x,y) = i \ \& \\ & I(x+\Delta x, y+\Delta y) = j \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where I is the image, i and j are pixel values, x and y are the spatial positions in the image I, and $(\Delta x, \Delta y)$ are the offsets in the horizontal and vertical directions in the image. Following the standard co-occurrence matrix definition [43], GLCM is created by calculating how often a certain pixel value i occurs horizontally adjacent to a pixel with value j. Therefore, the offsets Δx and Δy are defined as 1 and 0, respectively. Once the co-occurrence matrix is determined, the texture features representing the homogeneity, contrast, and local range data are computed for the current image segment that is being encoded. In this case, the contrast (Υ) and the homogeneity (Ψ) can be defined by,

$$\Upsilon = \sum_{i,j} \mid i-j \mid^2 C(i,j) \tag{7}$$

and

$$\Psi = \sum_{i,j} \frac{C(i,j)}{1+ \mid i-j \mid}, \tag{8}$$

respectively, where C denotes the co-occurrence matrix while i and j are pixel values for which the GLCM is defined.

c: Current CU information

This category corresponds to the information that is available for the block that is currently being encoded. These include QP, RD cost, and distortion of the $2N \times 2N$ PU mode. In this case, the proposed model is applied to the CU once the initial $2N \times 2N$ PU mode is evaluated for the current CU. Hence, the resulting RD cost and distortion for the $2N \times 2N$ PU mode are available for the inference process of the proposed algorithm (Fig. 5 in Sec-IV).

The features identified in Table 2 carry different weight contributions towards the CU split decision. Therefore, F-score for each feature at each CU depth level is calculated, in order to identify the features that contribute the most towards the split decision of the given CU [44]. In this case, for the binary classification problem with classes $+1$ (positive) and -1 (negative), F-Score for a given feature i is calculated as,

$$F_{score}(i) = \frac{(\bar{x}_i^+ - \bar{x}_i)^2 + (\bar{x}_i^- - \bar{x}_i)^2}{\frac{1}{P-1}\sum_{n=1}^{P}(x_{n,i}^+ - x_i^+)^2 + \frac{1}{M-1}\sum_{m=1}^{M}(x_{m,i}^- - x_i^-)^2} \tag{9}$$

where $\bar{x}_i$, $\bar{x}_i^+$, $\bar{x}_i^-$, P, and M refer to the average of the i^{th} feature, average of the positive instances, average of the negative instances, the number of positive instances, and the number of negative instances, respectively [45]. Here, $x_{n,i}^+$ and $x_{m,i}^-$ refer to the values of i^{th} feature in n^{th} positive instance and m^{th} negative instance, respectively.

The F-Score values computed for the features in each depth level are presented in Tables 3-5. Higher F-score value of a feature indicates that its contribution towards the CU split decision is significant. Hence, these F-Score figures are used to limit the number of features actually selected per CU depth level when generating the classification models. This ensures that the computational complexity of the inference process is maintained low while retaining the accuracy of the prediction models. In this regard, the number of features used at each CU depth level of the proposed algorithm is empirically decided to be four, which has given a balanced trade-off between model accuracy and computational complexity. In addition, the feature selection ensures only one of average or maximum depth of neighbouring CTUs is included in the final feature list to ensure that multiple aspects of the CU are taken into consideration for the models. The final sets of selected features for each CU depth level are depicted in Table 6.

B. TRAINING OFFLINE MODELS

Once the features are finalised and the dataset is prepared, the models are trained offline. In this case, several hyper-parameters specific to the SVMs are determined as follows.

First, the kernel function for the SVMs is determined as Radial Basis Function (RBF) as it demonstrates capabilities to handle SVM decision boundaries that are non-linear and is shown to efficiently work with small numbers of features [46], both of which are crucial for the proposed algorithm. Next, hyper-parameters associated with the SVMs are selected by following a grid search across a range of possible values as proposed in [47]. In this case, the hyper-parameters of the SVMs include cost (C), gamma (γ), and the number of training samples (N). Table 7 shows the range of values evaluated in the grid search to estimate the optimal values for the respective parameters. For parameters C and γ, a step size of 2 between the upper and lower bounds is used. The combination of hyper-parameters that provide the highest accuracy for the SVM models is used for model training. In this case, the accuracy (ζ) for each combination of hyper-parameters is evaluated using,

$$\zeta = \frac{T^+ + T^-}{N_{tot}}, \tag{10}$$

where T^+, T^-, and N_{tot} denote the numbers of true positive samples, true negative samples, and total predictions, respectively.

Training and cross-validation datasets for hyper-parameter tuning are formed by splitting the data gathered during the feature selection process. In this case, datasets are divided to ensure that there are equal numbers of samples from both *split* and *non-split* classes in the training sets, because SVMs perform poorly for unbalanced data [48]. The number of training samples used in this hyper-parameter tuning stage varies from 400 up to 1200, with a step size of 100.

In general, higher numbers of training samples result in higher accuracy, yet at the same time increase the number of

TABLE 3. F-Score Values for Depth 0

Feature	F-Score
CTU A-avg. depth	0.1862
CTU A-max. depth	0.1842
Colocated CU-max. depth	0.1833
Colocated CU-avg. depth	0.1748
QP	0.1599
CTU AR-max. depth	0.1010
CTU L-avg. depth	0.0938
CTU L-max. depth	0.0930
CTU A-bits	0.0922
CTU AR-avg. Depth	0.0876

TABLE 4. F-Score Values for Depth 1

Feature	F-Score
Colocated CU-max. depth	0.1932
Colocated CU-avg. depth	0.1777
QP	0.1291
CTU L-max. depth	0.1046
CTU A-max. depth	0.0977
CTU L-avg. depth	0.0943
CTU A-avg. depth	0.0865
CTU L-bits	0.0722
CTU A-bits	0.0708
CTU AR-max. depth	0.0619

TABLE 5. F-Score Values for Depth 2

Feature	F-Score
Colocated CU-max. depth	0.2319
Colocated CU-avg. depth	0.2146
QP	0.1699
CTU L-max. depth	0.1099
CTU L-avg. depth	0.0962
CTU A-bits	0.0674
CTU L-bits	0.0670
CTU A-max. depth	0.0660
CTU A-avg. depth	0.0641
homogeneity	0.0584

TABLE 6. Selected Features for each CU Depth Level

Depth 0	Depth 1	Depth 2
CTU A-avg depth	Coloc. CU-max depth	Coloc. CU-max depth
Coloc. CU-max depth	QP	QP
QP	CTU L-max depth	CTU L-max depth
CTU AR-max. depth	CTU A-max depth	CTU A-bits

TABLE 7. Hyper-parameters and their value ranges

Parameter	Values
Cost Parameter (C)	2^{-7} - 2^{7}
Gamma Parameter (γ)	2^{-7} - 2^{7}
Number of training Data (N)	400 - 1200

support vectors. This leads to an increase in the prediction time, which ultimately affects the encoding time gains that can be achieved. Therefore, the maximum number of training samples considered for this grid search is empirically determined to be limited to 1200 in this work. The upper bound of 1200 ensures that the maximum possible number of support vectors is limited to 1200. A cross-validation set is defined with the size of 600, that encompasses 300 samples for each *split* and *non-split* classes.

The hyper-parameter value combination that results in the highest accuracy is selected as the optimal combination. This search is conducted for all depth levels, i.e., depth 0 to 2, and Table 8 shows the selected hyper-parameter values for each depth.

C. COMPUTING THE CU SPLIT DECISION

The posterior probabilities of the split and non-split classes, calculated during the SVM prediction, are checked to predict the class for the new instances. Then, the CU split decision from the SVM prediction models (O_{SVM}) is determined as,

$$O_{SVM} = \begin{cases} \text{split} & \text{if } p^{+} \geq T_{svm} \\ \text{not decided} & \text{otherwise} \end{cases}, \qquad (11)$$

where p^{+} and T_{svm} denote the posterior probability determined by the SVM model for the *split* class and the SVM decision threshold, respectively.

By default, the probability threshold (T_{svm}) is set to 0.5. However, increasing T_{svm} on the other hand increases the confidence of the *split* decision taken in (11). For example, if the decision to *split* a CU is taken with higher confidence (i.e., higher posterior probability for class *split*), the decision is more likely to be accurate. In this case, the proposed algorithm offers the flexibility to change T_{svm} to trade-off the encoding complexity reduction to the coding efficiency. If the p^{+} probability does not exceed the T_{svm} threshold, the CU split decision is handed over to the traditional RD optimisation. Hence, T_{svm} acts as a design parameter that controls the number of CU split decisions taken by the SVM models and RD optimisation. Increasing T_{svm} leads to increases in the number of CU split decisions taken by RD optimisation resulting in less encoding time reductions. Yet, at the same time, this ensures that the impact on the coding efficiency is minimal as the less confident SVM decisions are discarded in the process.

TABLE 8. Selected Hyper-parameter values from the Grid Search

Parameter	Depth 0	Depth 1	Depth 2
C	4	64	0.5
γ	2^{-7}	0.5	2^{-7}
N	400	600	600

III. CU SPLIT DECISION CLASSIFICATION USING PROBABILISTIC MODELS

The SVM models discussed in Sec. II are generic models generated by performing an offline training process on the data collected offline. However, it has been shown that offline trained generic models may not perform well with the dynamics of the video content. Thus, content specific CU split likelihood models are important to achieve a higher prediction accuracy [49]. Therefore, this section describes an online prediction model (that acts alongside the offline SVM prediction models) that can keep the CU split decision prediction content-adaptive.

The CU split likelihood can be modelled as a Bayesian probabilistic model [15], [37]. In this case, the posterior probability of whether a CU is split at a particular depth level

d is given by,

$$P(S_d \mid X_d) = \frac{P(X_d \mid S_d)\, P(S_d)}{P(X_d)}, \tag{12}$$

where $P(S_d)$, is the prior probability of a CU being split at depth d, $P(X_d)$ is the marginal likelihood of observing a feature vector X_d in the feature space at depth d, and $P(X_d \mid S_d)$ is the likelihood of a feature vector X_d, given the CU is split at depth d.

In this case, the features that have been identified in Table 6 are taken as inputs to X_d. The relationship among the features in X_d and CU split decision is complex and content-dependant as depicted in the histograms in Fig. 4. The graphs show the impact of a feature vector X_d towards the CU split decision for a given QP.

The data for the probabilistic model given in (12) are accumulated during the encoding process. The initial frames of the sequence are encoded using traditional RD optimisation until N_p number of data points are collected for each feature vector at each CU depth level. Once the data points are accumulated, the decision to split or not to split a CU is now obtained by comparing (12) with an empirically determined threshold T_b. Hence, the CU split decision for a CU at depth d is given by

$$D_{fs}\big|_d = \begin{cases} 1 & P(S \mid \mathbf{X})\big|_d \geq T_b \\ 0 & \text{otherwise} \end{cases}. \tag{13}$$

The threshold T_b acts as a switch that decides either to split or not to split the CUs. Empirical observations reveal that the value of T_b impacts both the bit rate and the quality. A large T_b results in less CUs being split whereas a smaller T_b generally forces more CUs to split. In this case, T_b becomes an important parameter that can trade-off between coding efficiency and encoding complexity. Thus, T_b is maintained as a design parameter that can be preset to achieve the desired trade-off of the quality and the bit rate.

The same set of features are taken for both offline trained SVMs and online built probabilistic models. The relationship among the features in X and CU split decision is complex and content-dependant as depicted in the histograms in Fig. 4. The graphs show the impact of feature value to the CU split decision for all features for a given QP value. For the data depicted in the graphs, the QP has been fixed at 24.

TABLE 9. Encoding time reduction and coding efficiency loss when using SVM and probabilistic models at each CU depth

CU depth	SVM		Probabilistic Model	
	ΔT (%)	BDBR (%)	ΔT (%)	BDBR (%)
Depth 0	22.26	0.11	21.32	0.50
Depth 1	22.42	0.26	16.59	0.22
Depth 2	20.65	0.81	20.07	0.05

IV. PROPOSED METHOD

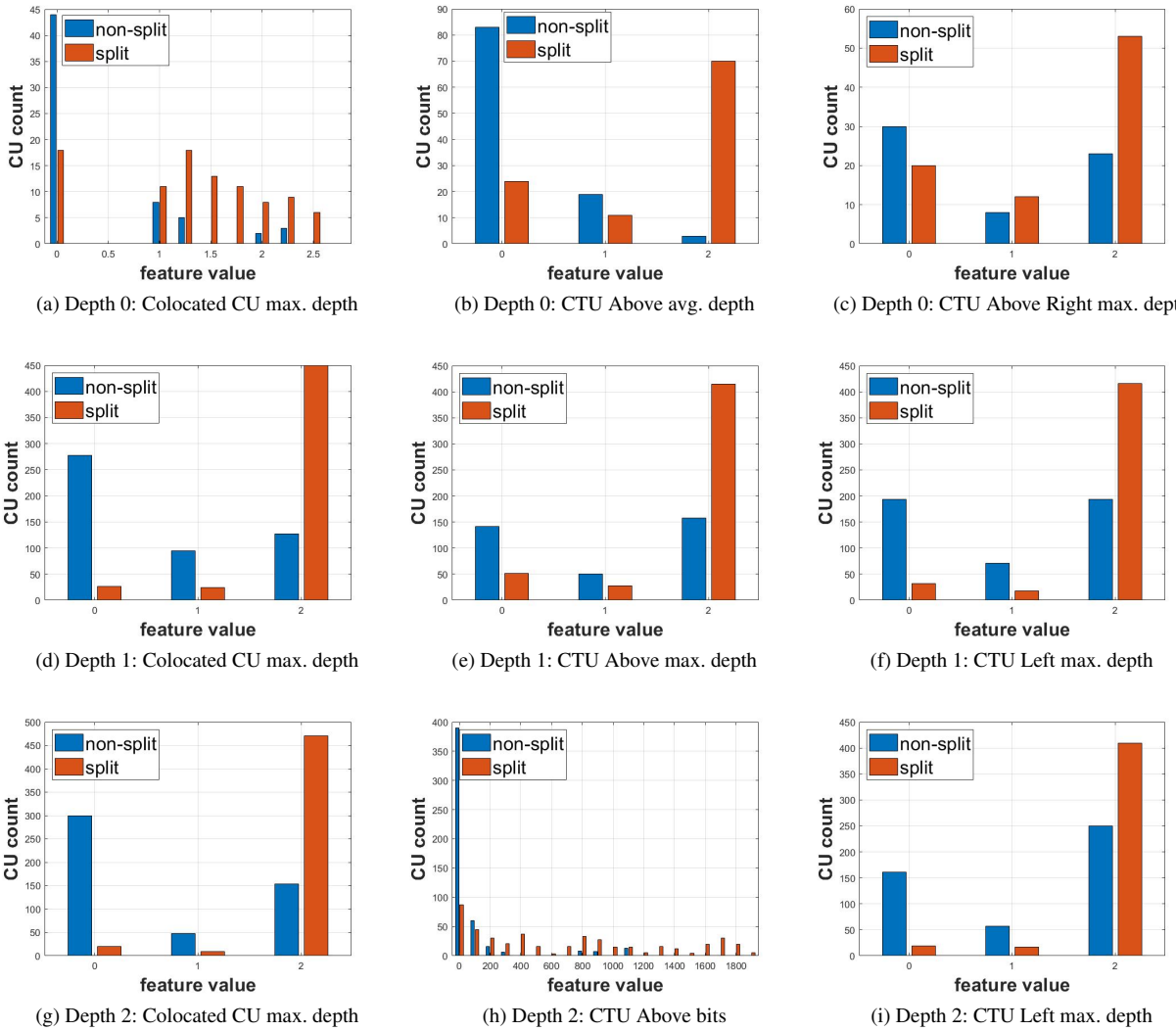
This section describes the overall CU size selection algorithm that makes use of the two CU split likelihood modelling methods described in Sec. II and Sec. III.

The SVM and Bayesian prediction models discussed in Sec. II, and Sec. III, respectively, demonstrate different characteristics in their performance in terms of encoding time reduction and coding efficiency. For example, offline-trained models (such as SVM based model described in Sec. II) generally achieve high computational complexity reductions compared to the online-trained models (such as probabilistic model described in Sec. IV). The time spent during the encoding process to collect sequence-specific data hinders the encoding time reduction achieved in the latter. On the other hand, online-trained probabilistic models demonstrate less Bjøntegaard Delta Bit Rate (BDBR) [50] losses since the CU split decisions made in general are content-adaptive and relevant to the video content that is being encoded [49].

In this context, an experimental sweep is conducted using a set of training sequences to evaluate the impact of each CU split decision estimation model on each CU depth level. Table 9 depicts the encoding time performance achieved against the BDBR loss at each CU depth level for each prediction model. The experiment is carried out by enabling only the SVM or the probabilistic model at a particular depth level while keeping the RD optimisation for the remaining depth levels to make the split decision. The set of video sequences (*BasketBallPass*, *Cactus*, *Kimono*, *PartyScene*, and *RaceHorses*) used for this experiment is chosen such that it is a representative of a wide diversity of video characteristics. T_{svm} used in this experiment varies from 0.1 to 0.9 and T_b varies from 0.5 to 0.9, where a step-size of 0.1 is used for both parameters.

The experimental results in Table 9 show that at depth 0, SVM prediction model yields a slightly higher encoding time complexity reduction when compared to the Bayesian probabilistic model, but with much less BDBR loss. Similarly, at depth 1, the SVM prediction model achieves a low BDBR loss difference at higher time complexity reduction when compared with the probabilistic model. Based on this, the proposed CU size selection algorithm is defined to use the offline trained SVM prediction models for the CU depth levels 0 and 1 to determine whether to split the current CU. Referring to the performance in Table 9, SVM model at CU depth level 2 shows higher BDBR loss compared to the probabilistic model for a similar encoding time reduction. Hence, the Bayesian probabilistic model is adopted at depth 2 to make the CU split/non-split decision in the proposed fast encoding algorithm.

In addition to the proposed offline-trained SVM and online-trained Bayesian prediction models, the status of the *SKIP* mode is also taken into consideration for the CU size selection. When the CU split decision is set to be taken by the RD optimisation, the encoder evaluates PU modes of both the current and the next depth levels. Once the calculations for the PU modes of the current depth level are complete,



(a) Depth 0: Colocated CU max. depth (b) Depth 0: CTU Above avg. depth (c) Depth 0: CTU Above Right max. depth

(d) Depth 1: Colocated CU max. depth (e) Depth 1: CTU Above max. depth (f) Depth 1: CTU Left max. depth

(g) Depth 2: Colocated CU max. depth (h) Depth 2: CTU Above bits (i) Depth 2: CTU Left max. depth

FIGURE 4. The relationship between the features identified in Table 6 and CU split decision for each depth $d \in 0, 1, 2$. The histograms depict the number of occurrences of each feature for split and non-split CUs for QP=24.

TABLE 10. Sequence details and abbreviations

Sequence Name	Resolution	frames per second	Abbreviation
PeopleOnStreet	2560×1600	30	S1
ParkScene	1920×1080	24	S2
Cactus	1920×1080	50	S3
BQTerrace	1920×1080	60	S4
BasketBallDrill	832×480	50	S5
PartyScene	832×480	50	S6
RaceHorses(D)	416×240	30	S7
BQSquare	416×240	60	S8
FourPeople	1280×720	60	S9
Johnny	1280×720	60	S10
KristenAndSara	1280×720	60	S11

the best PU mode for the current CU size (i.e., depth level) is selected by the RD optimisation. If the *SKIP* mode is selected as the best PU mode for the current depth level, further splitting of the CU is terminated. The *SKIP* mode usually corresponds to a static image region where there is no residual [51] (i.e., the image block can be perfectly predicted from a previous image). Hence, further splitting the CU into sub-CUs is proven to be ineffectual [25]. A similar approach is adopted in the proposed algorithm to avoid any redundant sub-CU level PU mode evaluations, and thereby further reducing the encoding time.

Fig. 5 illustrates the overall architecture of the proposed encoding algorithm that uses both the SVM and probabilistic CU split likelihood models to determine the CU split decisions.

V. EXPERIMENTAL RESULTS AND DISCUSSION

This section presents the experimental results of the proposed fast CU size selection algorithm for HEVC inter-prediction. The proposed algorithm is implemented in HM16.8 reference

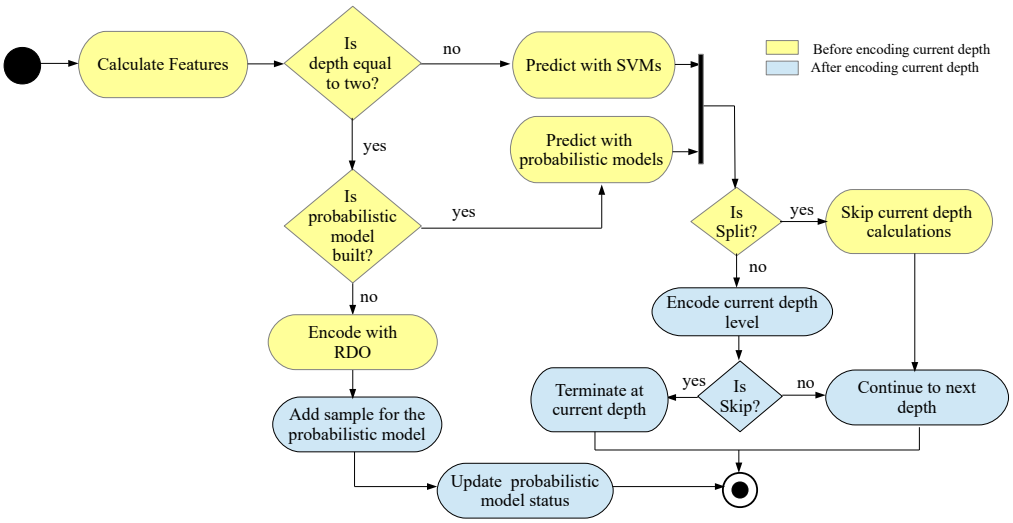
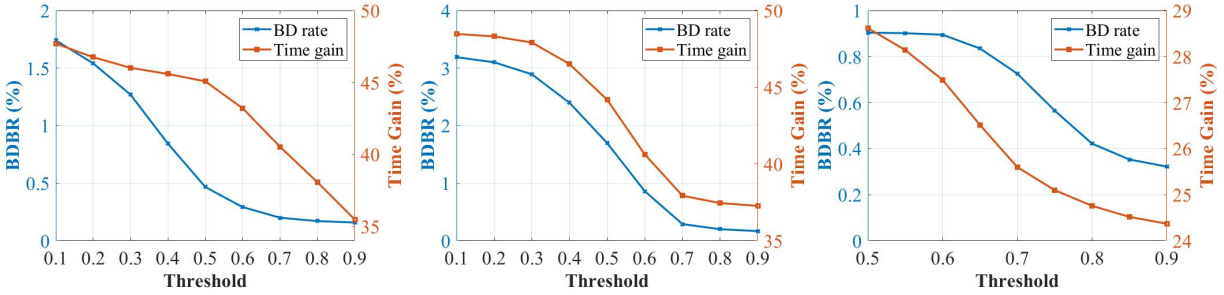


FIGURE 5. iCUS high-level architecture: elements in *yellow* and *blue* depict steps before and after the current level encoding, respectively



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

FIGURE 6. Impact of the model threshold on BD rate and time gain

software [17] and the optimised SVM library *libSVM* [46] is employed in implementations of the SVM prediction models described in Sec.II. The impacts on the coding efficiency and encoding time gains of the proposed algorithm are compared with several state-of-the-art algorithms that include HM16.8 [17], SVM based CU size selection algorithm proposed by Grellert *et al.* [10], content adaptive fast CU size selection algorithms proposed by Mallikarachchi *et al.* [15] and Lee *et al.* [24], and fast PU size selection algorithm proposed by Vanne *et al.* [25].

A. EXPERIMENTAL SETUP AND ENCODING CONFIGURATIONS

The proposed algorithm is evaluated for a range of HD and UHD video sequences that consist of content types ranging from simple to highly complex motion with diverse spatial and temporal characteristics. The list of sequences used in the experiments and their abbreviations (as reported in the subsequent tables) are presented in Table 10. In this experiment, the video sequences, encoding configurations, and QP values are selected as defined in the HEVC common test configurations [52]. For instance, *Low Delay B*, *Low Delay P*, and *Random Access* configurations are used for the encoding with QP $\in \{22, 27, 32, 37\}$. All experiments are carried out in an environment consisting of an AMD 64-Core CPU fixed at 2.5 GHz system with a 64 GB RAM and running Ubuntu 14.04 64-bit operating system. The number of processes (i.e., encoding instances) executed in parallel at any given time is set to four in order to ensure that the system is not overloaded.

The impact on the RD performance (i.e., coding efficiency) is evaluated using the BDBR metric [50] and the average percentage encoding time saving, ΔT, is evaluated for the proposed and state-of-the-art algorithms by,

$$\Delta T = 100 \times \frac{T_{HM} - T_{\rho}}{T_{HM}}, \tag{14}$$

where T_{HM} and T_{ρ} denote the encoding times of HM reference software and each of the fast encoding approaches, respectively.

B. RESULTS AND PERFORMANCE ANALYSIS

This section first discusses the impact of the CU split decision thresholds utilised in the SVM and probabilistic prediction models described in Sec. II and Sec. III, respectively. This is followed by an analysis of the overall performance and the implications of using the proposed fast coding framework.

TABLE 11. Overall performance of the proposed encoding algorithm (*Low Delay B* configuration)

Sequence	iCUS (τ=0.7) vs HM			iCUS (τ=0.6) vs HM			Grellert *et al.* [10] ($threshold$=0.5) vs HM			Mallikarachchi *et al.* [15] vs HM			Lee *et al.* [24] vs HM			Vanne *et al.* [25] vs HM		
	ΔT (%)	BDBR (%)	BD-PSNR (dB)	ΔT (%)	BDBR (%)	BD-PSNR (dB)	ΔT (%)	BDBR (%)	BD-PSNR (dB)	ΔT (%)	BDBR (%)	BD-PSNR (dB)	ΔT (%)	BDBR (%)	BD-PSNR (dB)	ΔT (%)	BDBR (%)	BD-PSNR (dB)
S1	33.61	1.44	-0.06	48.39	2.08	-0.09	21.77	0.93	-0.04	25.07	0.86	-0.03	31.94	1.81	-0.07	49.44	1.68	-0.07
S2	48.93	0.82	-0.02	56.18	2.42	-0.07	46.60	0.76	-0.02	42.53	1.07	-0.03	57.26	3.14	-0.09	55.37	1.74	-0.05
S3	43.14	0.95	-0.02	51.73	3.60	-0.07	44.38	1.51	-0.03	31.75	0.99	-0.02	51.83	2.97	-0.06	53.04	1.78	-0.04
S4	49.46	0.60	-0.01	55.88	2.82	-0.04	47.84	0.57	-0.01	41.38	0.92	-0.01	60.48	4.27	-0.06	54.45	1.96	-0.03
S5	34.97	1.09	-0.04	43.07	2.32	-0.09	37.09	1.04	-0.04	34.32	0.88	-0.03	44.75	2.62	-0.10	52.44	1.58	-0.06
S6	39.28	1.32	-0.05	49.11	2.26	-0.09	29.21	0.51	-0.02	32.36	0.89	-0.03	46.39	3.18	-0.12	52.01	1.48	-0.06
S7	28.31	1.33	-0.06	38.89	2.17	-0.10	17.87	0.82	-0.04	25.94	0.99	-0.04	29.03	3.15	-0.14	44.69	1.70	-0.07
S8	34.92	0.60	-0.02	42.78	1.78	-0.06	34.26	0.50	-0.02	31.05	1.08	-0.03	56.13	5.28	-0.16	55.72	1.71	-0.05
S9	61.74	0.43	-0.01	65.36	2.61	-0.08	62.75	1.05	-0.04	52.26	1.40	-0.04	73.60	1.42	-0.04	59.58	2.63	-0.08
S10	67.03	0.22	-0.00	69.30	1.94	-0.03	68.73	0.94	-0.02	56.56	1.69	-0.04	81.74	3.74	-0.08	61.98	3.43	-0.07
S11	64.80	0.19	-0.01	67.35	1.88	-0.05	65.13	0.42	-0.01	52.81	1.78	-0.05	76.21	2.97	-0.08	60.68	3.02	-0.08
Avg.	**46.02**	**0.82**	**-0.03**	**53.46**	**2.35**	**-0.07**	**43.24**	**0.82**	**-0.03**	**38.73**	**1.14**	**-0.03**	**55.40**	**3.14**	**-0.09**	**54.49**	**2.06**	**-0.06**

1) Impact of the model thresholds

The ideal scenario for a complexity reduction algorithm is to obtain maximum time gain with minimum coding loss. However, as observed in Fig. 6, the time gain and the BDBR loss have a direct relationship, where BDBR loss increases with the increasing time gain, and vice-versa. These graphs correspond to the average encoding time gain and the BDBR loss data from a set of four video sequences (*BasketballDrive*, *BasketbllPass*, *RaceHorses*, and *Vidyo*) with the prediction models for each CU depth level being utilised as described in Sec. IV (i.e. SVM model is active for depth levels 0 and 1 and the probabilistic model is active for depth level 2). In this case, an experimental sweep is conducted such that T_{svm} and T_b is varied from 0.1 to 0.9 with a step size of 0.1. The resulting encoding time reduction (ΔT %) and BDBR reduction are illustrated in Fig. 6.

These thresholds in the proposed algorithm are defined as design parameters that facilitate the algorithm to trade-off the encoding time gain to the coding efficiency depending on the requirement. For the results presented in the subsequent sections, the two thresholds (both represented by τ)) are set as $T_{svm} = 0.7, 0.6$ and $T_b = 0.7, 0.6$.

2) Overall performance of the proposed algorithm

The performance of the proposed algorithm is presented in the Tables 11, 12, and 13 for the *low delay B*, *random access* and *low delay P* configurations, respectively. The impact of the video content, QPs, and other relevant attributes on the

TABLE 12. Overall performance of the proposed encoding algorithm (*Random Access* configuration)

Sequence	iCUS (τ=0.7) vs HM			iCUS (τ=0.6) vs HM			Grellert *et al.* [10] ($threshold$=0.5) vs HM			Mallikarachchi *et al.* [15] vs HM			Lee *et al.* [24] vs HM			Vanne *et al.* [25] vs HM		
	ΔT (%)	BDBR (%)	BD-PSNR (dB)	ΔT (%)	BDBR (%)	BD-PSNR (dB)	ΔT (%)	BDBR (%)	BD-PSNR (dB)	ΔT (%)	BDBR (%)	BD-PSNR (dB)	ΔT (%)	BDBR (%)	BD-PSNR (dB)	ΔT (%)	BDBR (%)	BD-PSNR (dB)
S1	42.03	2.11	-0.09	55.18	2.98	-0.12	23.85	2.23	-0.09	27.31	0.79	-0.03	30.52	5.08	-0.19	45.18	1.59	-0.06
S2	57.80	1.10	-0.03	63.48	3.33	-0.10	54.66	0.91	-0.03	43.33	0.73	-0.02	60.20	4.82	-0.14	52.91	1.17	-0.03
S3	54.46	1.61	-0.03	61.48	5.08	-0.09	51.81	1.80	-0.04	40.09	0.58	-0.01	64.61	3.60	-0.07	62.88	1.30	-0.03
S4	60.90	1.60	-0.02	65.50	3.95	-0.05	55.23	1.11	-0.02	41.82	0.62	-0.01	66.00	5.04	-0.07	54.85	1.25	-0.02
S5	52.35	1.41	-0.06	59.36	3.79	-0.15	46.28	1.59	-0.06	35.81	0.57	-0.02	46.34	5.26	-0.20	49.51	1.40	-0.05
S6	47.53	1.30	-0.05	53.82	2.13	-0.09	42.97	1.17	-0.05	31.63	0.56	-0.02	48.07	3.32	-0.13	49.03	0.99	-0.04
S7	29.62	1.63	-0.07	36.81	2.34	-0.10	35.49	2.04	-0.09	27.47	1.06	-0.04	30.62	4.41	-0.19	44.05	1.92	-0.08
S8	51.80	0.79	-0.03	56.60	1.44	-0.05	49.55	0.78	-0.03	45.28	0.57	-0.02	61.98	2.42	-0.08	57.83	0.76	-0.03
S9	70.45	0.54	-0.02	72.48	1.86	-0.06	68.40	0.86	-0.03	52.44	0.35	-0.01	74.72	2.12	-0.07	59.20	0.96	-0.03
S10	74.04	0.47	-0.01	75.78	2.51	-0.06	72.22	0.45	-0.01	56.12	0.33	-0.01	79.31	3.58	-0.08	58.24	1.24	-0.03
S11	70.34	0.52	-0.02	72.20	2.44	-0.07	70.76	0.74	-0.02	54.56	0.48	-0.01	77.20	2.69	-0.08	60.19	1.44	-0.04
Avg.	**55.57**	**1.19**	**-0.04**	**61.15**	**2.90**	**-0.09**	**51.93**	**1.24**	**-0.04**	**41.44**	**0.60**	**-0.02**	**58.14**	**3.85**	**-0.12**	**53.99**	**1.27**	**-0.04**

TABLE 13. Overall performance of the proposed encoding algorithm (*Low Delay P* configuration)

Sequence	iCUS (τ=0.7) vs HM			iCUS (τ=0.6) vs HM			Grellert *et al.* [10] ($threshold$=0.5) vs HM			Mallikarachchi *et al.* [15] vs HM			Lee *et al.* [24] vs HM			Vanne *et al.* [25] vs HM		
	ΔT (%)	BDBR (%)	BD-PSNR (dB)	ΔT (%)	BDBR (%)	BD-PSNR (dB)	ΔT (%)	BDBR (%)	BD-PSNR (dB)	ΔT (%)	BDBR (%)	BD-PSNR (dB)	ΔT (%)	BDBR (%)	BD-PSNR (dB)	ΔT (%)	BDBR (%)	BD-PSNR (dB)
S1	49.11	1.59	-0.07	61.28	2.45	-0.12	27.09	2.57	-0.09	26.09	1.71	-0.19	42.88	1.38	-0.06	25.65	0.74	-0.03
S2	48.06	0.89	-0.03	54.94	2.42	-0.10	47.45	1.69	-0.03	51.61	3.58	-0.14	51.71	1.57	-0.03	39.12	1.12	-0.02
S3	54.17	1.46	-0.03	62.13	3.83	-0.09	48.74	3.23	-0.04	44.54	2.87	-0.07	48.57	1.50	-0.03	36.73	0.94	-0.01
S4	56.93	1.73	-0.02	63.14	2.84	-0.05	49.79	2.00	-0.02	52.71	4.11	-0.07	48.93	1.82	-0.02	40.90	1.01	-0.01
S5	41.13	1.17	-0.04	48.55	2.55	-0.15	43.87	3.04	-0.06	36.61	1.86	-0.20	46.79	1.42	-0.05	31.59	0.82	-0.02
S6	41.69	1.46	-0.05	50.69	2.28	-0.09	35.18	1.27	-0.05	37.00	3.21	-0.13	44.84	1.32	-0.04	33.23	0.82	-0.02
S7	27.12	1.63	-0.07	35.45	2.35	-0.10	16.56	2.54	-0.09	27.85	3.37	-0.19	42.46	1.52	-0.08	23.72	0.94	-0.04
S8	45.60	1.15	-0.04	51.52	1.64	-0.05	38.43	1.63	-0.03	44.79	5.17	-0.08	45.12	1.43	-0.03	35.06	1.03	-0.02
S9	69.45	0.51	-0.02	72.34	2.57	-0.06	64.85	2.36	-0.03	66.66	1.35	-0.07	56.84	2.28	-0.03	50.61	1.24	-0.01
S10	70.14	-0.04	0.00	72.24	1.58	-0.06	70.36	1.78	-0.01	74.70	3.42	-0.08	59.02	2.99	-0.03	56.64	1.77	-0.01
S11	64.84	-0.27	0.01	67.32	1.38	-0.07	67.31	1.29	-0.02	70.54	2.87	-0.08	59.22	3.12	-0.04	52.85	1.51	-0.01
Avg.	**51.66**	**1.03**	**-0.03**	**58.15**	**2.35**	**-0.09**	**46.33**	**2.13**	**-0.04**	**48.46**	**3.05**	**-0.12**	**49.67**	**1.85**	**-0.04**	**38.74**	**1.09**	**-0.02**

TABLE 14. Performance comparison with ECU [27], ESD [28], and CFM [29] (*Low Delay B Configuration*)

Sequence	iCUS (τ=0.7) vs HM			ECU [27] vs HM			ESD [28] vs HM			CFM [29] vs HM			ECU [27], ESD [28], and CFM [29] vs HM		
	ΔT (%)	BDBR (%)	BD-PSNR (dB)	ΔT (%)	BDBR (%)	BD-PSNR (dB)	ΔT (%)	BDBR (%)	BD-PSNR (dB)	ΔT (%)	BDBR (%)	BD-PSNR (dB)	ΔT (%)	BDBR (%)	BD-PSNR (dB)
S1	33.61	1.44	-0.06	15.10	0.24	-0.01	16.60	0.37	-0.02	23.38	1.28	-0.06	24.69	1.37	-0.06
S2	48.93	0.82	-0.02	38.19	0.34	-0.01	33.66	0.31	-0.01	39.87	0.84	-0.02	50.54	1.46	-0.04
S3	43.14	0.95	-0.02	34.92	0.52	-0.01	30.04	0.36	-0.01	35.45	1.01	-0.02	46.03	1.72	-0.04
S4	49.46	0.60	-0.01	42.01	0.19	-0.00	35.97	0.23	-0.00	41.17	0.91	-0.01	54.29	1.35	-0.02
S5	34.97	1.09	-0.04	25.43	0.12	-0.01	23.92	0.29	-0.01	27.66	0.60	-0.02	34.81	0.79	-0.03
S6	39.28	1.32	-0.05	18.52	0.08	-0.00	21.01	0.37	-0.02	31.12	0.98	-0.04	32.61	1.16	-0.05
S7	28.31	1.33	-0.06	12.02	0.20	-0.01	13.92	0.35	-0.02	23.01	1.56	-0.07	21.60	1.32	-0.06
S8	34.92	0.60	-0.02	24.04	0.08	-0.00	28.32	0.53	-0.02	39.61	1.56	-0.05	40.99	1.55	-0.05
S9	61.74	0.43	-0.01	57.72	0.05	-0.00	46.29	0.22	-0.01	49.80	1.06	-0.04	70.30	1.01	-0.03
S10	67.03	0.22	-0.00	63.92	0.24	-0.00	51.26	0.46	-0.01	55.14	1.73	-0.04	77.39	1.45	-0.03
S11	64.80	0.19	-0.01	58.81	0.11	-0.01	47.38	0.52	-0.02	51.68	1.60	-0.05	71.12	1.27	-0.04
Avg.	**46.02**	**0.82**	**-0.03**	**35.52**	**0.20**	**-0.01**	**31.68**	**0.36**	**-0.01**	**37.99**	**1.19**	**-0.04**	**47.67**	**1.31**	**-0.04**

performance are analysed in the following discussion with additional tables.

a: Performance variation with QP

Table 15 demonstrates the encoding time gains achieved for a subset of the sequences for each QP $\in$ $22, 27, 32, 37$ for the *Random Access* configuration. It can be observed that the proposed algorithm (iCUS) achieves significant and consistent encoding time gains for all QP values. However, an increase in the achievable time gain is noted with the increasing QPs. This can be attributed to the features selected in the SVM and probability models. A category of features used in the proposed algorithm (Ref. Table 2) corresponds to the information attained from neighbouring CTUs. This indicates that the CU split decisions are influenced by the split decisions of the neighbouring blocks. When the QP value is low, a picture is partitioned into more CUs, reducing the correlation among the neighbouring blocks. This is comparable to [25] that reports increasing time gains with increasing QP values. However, other state-of-the-art methods demonstrate considerably weak performance at lower QP values.

In addition, the selection of *SKIP* mode as the best PU mode for a CU depth level is minimal when using lower QPs. Hence, it is noticeable that large CU depth levels (i.e., smaller CU sizes such as $16 \times 16, 8 \times 8$) are frequently utilised within a frame when the QP is low. Hence, the achievable encoding time gain can be hindered for algorithms that utilise *SKIP* mode as a decision making feature (i.e., [24]) when they are operating with lower QPs.

Furthermore, a similar observation can be made for highly textured and complex sequences such as *PeopleOnStreet* (S1) when using lower QPs for encoding. The complex textures

TABLE 15. Performance variation with QP (Random Access Configuration)

Sequence	iCUS (τ=0.7)				Grellert *et al.* [10] ($threshold$=0.5)				Mallikarachchi *et al.* [15]				Lee *et al.* [24]				Vanne *et al.* [25]			
	QP 22	QP 27	QP 32	QP 37	QP 22	QP 27	QP 32	QP 37	QP 22	QP 27	QP 32	QP 37	QP 22	QP 27	QP 32	QP 37	QP 22	QP 27	QP 32	QP 37
S1	37.2	42.1	43.8	45.0	10.0	22.7	23.4	31.8	18.0	25.2	28.9	37.1	18.5	26.0	32.2	45.3	38.3	42.7	47.4	52.4
S3	45.7	51.5	58.4	62.2	28.2	47.2	55.6	62.2	31.4	38.8	44.1	46.0	50.6	62.4	70.1	75.4	56.1	62.8	65.6	67.1
S6	37.2	44.4	50.2	58.3	15.3	30.8	45.1	55.2	22.9	28.1	35.0	40.5	29.5	43.1	55.0	64.6	41.1	46.9	50.9	57.2
S8	32.9	47.0	57.9	69.5	18.7	40.3	56.7	63.8	30.4	41.8	52.5	56.5	38.3	59.9	72.4	77.3	48.9	57.5	61.2	63.6
S9	61.3	69.7	74.3	76.5	56.8	64.6	69.7	71.5	44.5	50.9	55.6	58.8	63.8	73.6	79.3	82.1	54.3	59.0	61.2	62.3
Avg.	**42.9**	**50.9**	**56.9**	**62.3**	**25.8**	**41.1**	**50.1**	**56.9**	**29.4**	**37.0**	**43.2**	**47.8**	**40.1**	**53.0**	**61.8**	**69.0**	**47.7**	**53.8**	**57.3**	**60.5**

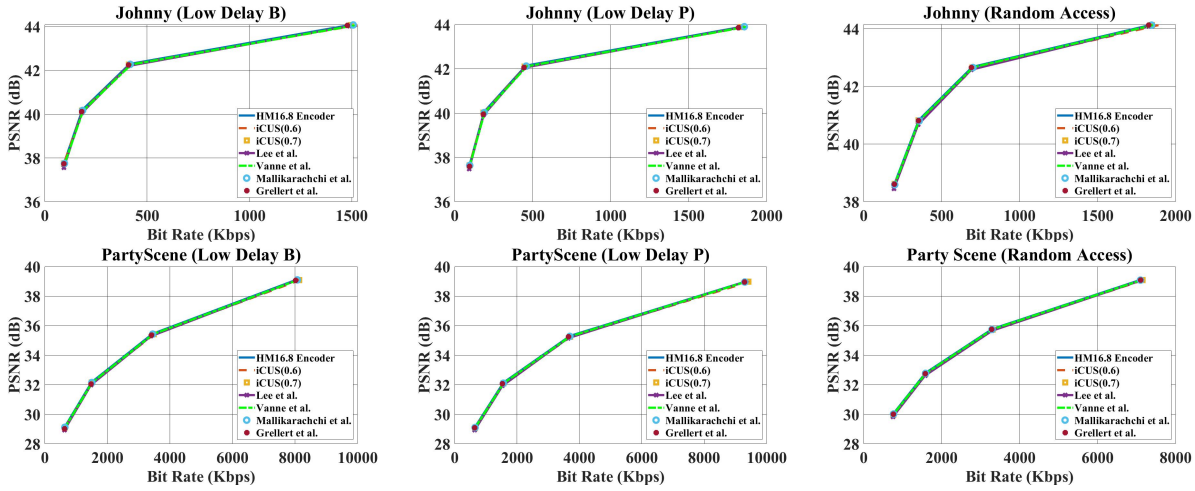


FIGURE 7. RD curves for Johnny(top) and PartyScene(bottom) for configurations *Low Delay B* (Colomn 1), *Low Delay P* (Column 2), and *Random Access* (Column 3).

and high motions result in a lower number of *SKIP* mode selections. Hence, the encoded bitstreams contain a large number of smaller CUs per frame, resulting in more decisions being taken by the RD optimisation compared to the proposed prediction models.

Following the same phenomenon, in the *FourPeople* (S9) video sequence, there is relatively lower motion with a large portion attributed to a homogeneous background. Therefore, a higher number of predictions can be made with the *SKIP* mode. Therefore, even at lower QPs, the method in [24] performs relatively better. However, the proposed method also achieves comparable results for all QPs in the sequence.

b: Performance variation with the content

The observations in Tables 11, 12, and 13 show that the proposed method achieves higher encoding time complexity reductions for sequences that have very limited motion (e.g. *Johnny*, *Four People*, and *KristenAndSara*). On the contrary, the sequences that have higher motion (e.g. *RaceHorses*, *PartyScene*, and *BasketballDrill*) achieve comparably less time gains. The high motion indicates more foreground motions (i.e., moving objects), leading to less correlation among neighbouring blocks and less occurrence in the selection of *SKIP* mode. To this end, iCUS decisions are influenced by both the neighbouring information as well as the *SKIP* mode, leading to more CUs being sent to RD optimisation, reducing the time complexity reductions that can be achieved.

Furthermore, the observed results show that for sequences with relatively higher motion (S1-S3), the proposed method achieves higher time gains when compared with the state-of-the-art methods. In these sequences, there is a high correlation in the colocated blocks in the reference frames. Incorporating this information as features in the models (Table 6) enables the proposed method to achieve higher time gains. Due to the high texture granularity and relatively higher motion, the methods that use *SKIP* as the main feature (e.g. [24]) fail to achieve higher time gains. However, as a result of using RD optimisation in such cases lead to less impact to the BDBR. On the contrary, the proposed method has a relatively higher impact on the BDBR when the threshold is increased. However, at lower values of the threshold, the proposed method manages to outperform the state-of-the-art methods with higher time gains and relatively lower BDBR impacts.

c: Performance variation with the encoding configuration

From the results presented in Table 11, it is observed that iCUS with $\tau = 0.6$ achieves significant time complexity reductions of 53.46% with BDBR losses of 2.35%. This outperforms [10] and [15] with higher time complexity reductions. Although [24] reports slightly higher time complexity reductions, this comes at a higher BDBR loss of 3.14%.

Vanne *et al.* [25] algorithm records a similar time complexity reduction as iCUS with $\tau = 0.7$. However, their algorithm lacks the flexibility to control the encoding time reduction to the coding efficiency achieved; a crucial improvement in the proposed algorithm compared to [25] and [24].

Table 12 depicts the results for the *Random Access* configuration. iCUS with $\tau = 0.7$ outperforms [10], [15] and [25], in terms of time gains and BDBR losses. When $\tau = 0.6$, iCUS achieves very high complexity reduction of 61.15%, at a much less and negligible BDBR loss of 2.9%, outperforming [24] both in terms of the encoding time complexity reduction and the BDBR loss. A similar observation can be made with the *Low Delay P* configuration results presented in Table 13, where the proposed algorithm with $\tau = 0.7$ outperforms all state-of-the-art methods.

RD curves for two sample sequences, *Johnny* and *PartyScene*, are given for all configurations in Fig. 7. It can be observed that the RD performance of the proposed iCUS algorithm is similar to that of HM16.8, thus, the BDBR loss of the proposed method can be considered negligible.

d: Impact of model selection on the encoding performance

As indicated in Sec. IV, the proposed algorithm utilises SVM prediction models for CU depths 0 and 1, whereas the CU depth 2 uses a Bayesian probabilistic model. The average experimental results presented in Table 16 summarise the encoding time and BDBR loss for the proposed algorithm when using different combinations of prediction models at each CU depth level. For instance, configurations defined in Table 16 correspond to **A**: using SVMs for all CU depths, **B**: using Bayesian models for all CU depths, **C**: using SVMs for CU depths 0 and 1, and RD optimisation for CU depth 2, and **D**: the proposed iCUS configuration. These experiments are conducted for the test video sequences that are not part of the initial training set given in the Table 1.

It can be observed that the configuration **D**, corresponding to using SVM models at depths 0 and 1, while using the probabilistic model at depth 2, gives the highest encoding time gain while also keeping the BDBR intact at an average of 0.62%. The impact of the rigidness of SVM models is evident in the performance of configuration **A** when all CU depth levels use the SVM models. The considerably low time gain in **B** suggests that building the Bayesian probabilistic models for all depth levels during the encoding time is inefficient from the perspective of time complexity. This is because the model keeps making RD optimisation based decisions until a sufficient number of data points are accumulated to to use the Bayesian model given in (13) . Configuration **C** on the other hand also provides a considerable time gain with a marginal BDBR loss. It can be seen that the addition of Bayesian models to obtain the decisions for CU depth level 2 increases the BDBR loss by 0.09%, whereas the encoding time reduction has improved by a margin of $\approx 7\%$.

TABLE 16. Performance of the proposed algorithm (iCUS with $\tau = 0.7$) when using multiple combinations of SVMs and Bayesian models at each CU depth level (*Low delay B main*)

Configuration	ΔT (%)	BDBR (%)
A	39.20	1.28
B	28.65	0.31
C	35.12	0.53
D	42.47	0.62

e: Performance Comparison with the HM16.8 built-in fast encoding methods

The performance of the proposed algorithm is also compared against the fast encoding techniques that are embedded in the HM16.8 reference encoder implementation, that include ECU [27], ESD [28], and CFM [29]. Their encoding time performance, together with the coding efficiency impact, is illustrated in Table 14. In this case, the experimental results presented in Table 14 correspond to the *low delay B main* configuration which gives the minimum BDBR increase for the proposed iCUS algorithm ($\tau = 0.7$). It can be observed that the proposed algorithm outperforms the individual built-in fast methods in terms of the encoding time reduction. Further, when all three built-in methods are simultaneously activated, the encoding time reduction achieved is similar to that of the proposed iCUS algorithm. However, the coding efficiency loss is much noticeable in the former compared to the latter. Furthermore, the ability of the proposed algorithm to trade-off the encoding time reduction to the coding efficiency loss is an extra benefit as opposed to the fixed time gains achieved when using the built-in fast encoding methods.

Finally, it should be noted that the computational costs associated with the SVM inference, data collection for the probabilistic models, and the decision-prediction are all included in performance calculations reported. Therefore, it is evident that the additional complexity overheads introduced by the proposed algorithm are negligible when compared with the significant time savings that can be achieved by incorporating these algorithms into the encoding cycle.

VI. CONCLUSION

This paper proposes an intelligent fast CU selection algorithm for HEVC inter-prediction. In this context, two split likelihood models (an offline-trained SVM model and an online-trained Bayesian probabilistic model) are introduced to predict the CU split and non-split decisions. The hybrid use of offline and online trained models for the CU size selection keeps the split/non-split decisions made during the encoding content-adaptive and accurate. Moreover, the flexibility of the algorithm to effectively trade-off the computational complexity to the coding efficiency is also investigated.

One of the main conclusions of this work is that the use of online models for CU size selection at specific CU depths keeps the algorithm content-adaptive. The data collection

during the encoding phase ensures the decisions made by the algorithm are content relevant hence avoids increases in the BDBR losses. Experimental results reveal that the proposed algorithm achieves significant encoding time gains across all QP values, with slightly higher performance when using higher QP values. As a result, it can be concluded that the proposed algorithm can be used to reduce the encoding time complexity in applications that need a diverse range of video quality levels. Furthermore, configuring the threshold levels in the SVM model and conditional probability for a given feature vector allows the algorithm to control the percentage of CUs that are evaluated using traditional RD optimisation. This eventually facilitates the algorithm to trade-off the coding complexity to the coding efficiency.

In conclusion, the simulation results demonstrate average encoding time performances of 53.46%, 61.15%, and 58.15 % for Low Delay B, Random Access, and Low Delay P configurations, respectively, with negligible impacts on the coding efficiency, across a wide range of content types and quality levels. The future work will focus on extending the algorithm to address the partitioning of PUs and TUs. Furthermore, other lightweight neural networks will also be analysed to train models to predict the entire CU, PU and TU structure for a given block to achieve higher encoding time complexity reductions while keeping the coding efficiency intact.

REFERENCES

[1] Cisco, "Cisco visual networking index: global mobile data traffic forecast update 2017-2022," Cisco, White Paper.

[2] Moving Picture Experts Group (MPEG), "Requirements for Future Video Coding Standard."

[3] 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.

[4] I.-K. Kim, J. Min, T. Lee, W.-J. Han, and J. Park, "Block partitioning structure in the HEVC standard," *IEEE transactions on circuits and systems for video technology*, vol. 22, no. 12, pp. 1697–1706, 2012.

[5] F. Bossen, B. Bross, K. Suhring, and D. Flynn, "HEVC complexity and implementation analysis," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1685–1696, 2012.

[6] "Mpeg-5 lcevc," https://lcevc.com/, accessed: 2019-12-01.

[7] C. E. Rhee, K. Lee, T. S. Kim, and H.-J. Lee, "A survey of fast mode decision algorithms for inter-prediction and their applications to high efficiency video coding," *IEEE Transactions on Consumer Electronics*, vol. 58, no. 4, pp. 1375–1383, 2012.

[8] L. Shen, Z. Liu, X. Zhang, W. Zhao, and Z. Zhang, "An effective CU size decision method for HEVC encoders," *IEEE Transactions on Multimedia*, vol. 15, no. 2, pp. 465–470, 2013.

[9] J. Xiong, H. Li, Q. Wu, and F. Meng, "A Fast HEVC Inter CU Selection Method Based on Pyramid Motion Divergence," *IEEE transactions on multimedia*, vol. 16, no. 2, pp. 559–564, 2014.

[10] M. Grellert, B. Zatt, S. Bampi, and L. A. da Silva Cruz, "Fast coding unit partition decision for HEVC using support vector machines," *IEEE Transactions on Circuits and Systems for Video Technology*, 2018.

[11] Y. Zhang, S. Kwong, X. Wang, H. Yuan, Z. Pan, and L. Xu, "Machine learning-based coding unit depth decisions for flexible complexity allocation in high efficiency video coding," *IEEE Trans. on Image Processing*, vol. 24, no. 7, pp. 2225–2238, 2015.

[12] X. Shen and L. Yu, "CU splitting early termination based on weighted SVM," *EURASIP Journal on Image and Video Processing*, vol. 2013, no. 1, p. 4, 2013.

[13] G. Correa, P. Assuncao, L. V. Agostini, and L. A. C. Silva, "Fast HEVC encoding decisions using data mining," *IEEE trans. on circuits and systems for video technology*, vol. 25, no. 4, pp. 660–673, 2015.

[14] M. Xu, T. Li, Z. Wang, X. Deng, R. Yang, and Z. Guan, "Reducing complexity of HEVC: A deep learning approach," *IEEE Transactions on Image Processing*, vol. 27, no. 10, pp. 5044–5059, 2018.

[15] T. Mallikarachchi, D. S. Talagala, H. K. Arachchi, and A. Fernando, "Content-Adaptive Feature-Based CU Size Prediction for Fast Low-Delay Video Encoding in HEVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 3, pp. 693–705, 2018.

[16] H.-S. Kim and R.-H. Park, "Fast cu partitioning algorithm for hevc using an online-learning-based bayesian decision rule," *IEEE transactions on circuits and systems for video technology*, vol. 26, no. 1, pp. 130–138, 2015.

[17] "HM Encoder 16.8," https://hevc.hhi.fraunhofer.de/trac/hevc/browser/tags/HM-16.8, accessed: 2018-02-04.

[18] J.-R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the coding efficiency of video coding standards—including high efficiency video coding (hevc)," *IEEE Transactions on circuits and systems for video technology*, vol. 22, no. 12, pp. 1669–1684, 2012.

[19] M. Grellert, M. Shafique, M. U. K. Khan, L. Agostini, J. C. Mattos, and J. Henkel, "An adaptive workload management scheme for hevc encoding," in *2013 IEEE International Conference on Image Processing*. IEEE, 2013, pp. 1850–1854.

[20] Z. Pan, J. Lei, Y. Zhang, X. Sun, and S. Kwong, "Fast motion estimation based on content property for low-complexity h. 265/hevc encoder," *IEEE Transactions on Broadcasting*, vol. 62, no. 3, pp. 675–684, 2016.

[21] Q. Liu, L. Liu, L. Hao, and T. Peng, "Fast motion estimation algorithm for high efficient video coding," in *2018 IEEE 4th International Conference on Computer and Communications (ICCC)*. IEEE, 2018, pp. 6–10.

[22] J. Joo and Y. Choi, "Dominant edge direction based fast parameter estimation algorithm for sample adaptive offset in hevc," in *2014 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2014, pp. 3749–3752.

[23] Z. Zhengyong, C. Zhiyun, and P. Peng, "A fast sao algorithm based on coding unit partition for hevc," in *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)*. IEEE, 2015, pp. 392–395.

[24] J. Lee, S. Kim, K. Lim, and S. Lee, "A fast CU size decision algorithm for HEVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 3, pp. 411–421, 2015.

[25] J. Vanne, M. Viitanen, and T. D. Hämäläinen, "Efficient mode decision schemes for HEVC inter prediction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 9, pp. 1579–1593, 2014.

[26] K. Choi and E. S. Jang, "Fast coding unit decision method based on coding tree pruning for high efficiency video coding," *Optical Engineering*, vol. 51, no. 3, p. 030502, 2012.

[27] K. Choi, S.-H. Park, and E. S. Jang, "Coding tree pruning based cu early termination," *JCT-VC document, JCTVC-F092*, 2011.

[28] J. Yang, J. Kim, K. Won, H. Lee, and B. Jeon, "Early skip detection for hevc," *document JCTVC-G543*, 2011.

[29] R.-h. Gweon and Y.-L. Lee, "Early termination of CU encoding to reduce HEVC complexity," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 95, no. 7, pp. 1215–1218, 2012.

[30] J. Xiong, H. Li, Q. Wu, and F. Meng, "A fast HEVC inter cu selection method based on pyramid motion divergence," *IEEE transactions on multimedia*, vol. 16, no. 2, pp. 559–564, 2013.

[31] Z. Pan, S. Kwong, M.-T. Sun, and J. Lei, "Early MERGE mode decision based on motion estimation and hierarchical depth correlation for HEVC," *IEEE Transactions on Broadcasting*, vol. 60, no. 2, pp. 405–412, 2014.

[32] F. Sampaio, S. Bampi, M. Grellert, L. Agostini, and J. Mattos, "Motion Vectors Merging: Low Complexity Prediction Unit Decision Heuristic for the Inter-prediction of HEVC Encoders," in *2012 IEEE international conference on multimedia and expo*. IEEE, 2012, pp. 657–662.

[33] W.-J. Hsu and H.-M. Hang, "Fast coding unit decision algorithm for HEVC," in *2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*. IEEE, 2013, pp. 1–5.

[34] Y. Li, Z. Liu, X. Ji, and D. Wang, "Cnn based cu partition mode decision algorithm for hevc inter coding," in *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 993–997.

[35] T. Li, M. Xu, and X. Deng, "A deep convolutional neural network approach for complexity reduction on intra-mode hevc," in *2017 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2017, pp. 1255–1260.

[36] T. Laude and J. Ostermann, "Deep learning-based intra prediction mode decision for hevc," in *2016 Picture Coding Symposium (PCS)*. IEEE, 2016, pp. 1–5.
[37] I. Rish *et al.*, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, no. 22, 2001, pp. 41–46.
[38] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE transactions on systems, man, and cybernetics*, vol. 21, no. 3, pp. 660–674, 1991.
[39] G. P. Zhang, "Neural networks for classification: a survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 30, no. 4, pp. 451–462, 2000.
[40] I. Steinwart and A. Christmann, *Support vector machines*. Springer Science & Business Media, 2008.
[41] D. G. Kleinbaum, K. Dietz, M. Gail, M. Klein, and M. Klein, *Logistic regression*. Springer, 2002.
[42] J. Shawe-Taylor and N. Cristianini, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press Cambridge, 2000, vol. 204.
[43] R. M. Haralick, K. Shanmugam, and I. H. Dinstein, "Textural features for image classification," *IEEE Transactions on systems, man, and cybernetics*, no. 6, pp. 610–621, 1973.
[44] Y.-W. Chen and C.-J. Lin, "Combining SVMs with Various Feature Selection Strategies," in *Feature extraction*. Springer, 2006, pp. 315–324.
[45] C. Goutte and E. Gaussier, "A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation," in *European Conference on Information Retrieval*. Springer, 2005, pp. 345–359.
[46] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," *ACM transactions on intelligent systems and technology (TIST)*, vol. 2, no. 3, p. 27, 2011.
[47] C.-W. Hsu, C.-C. Chang, C.-J. Lin *et al.*, "A practical guide to support vector classification," 2003.
[48] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intelligent data analysis*, vol. 6, no. 5, pp. 429–449, 2002.
[49] B. Erabadda, T. Mallikarachchi, G. Kulupana, and A. Fernando, "Machine Learning Approaches for Intra-Prediction in HEVC," in *2018 IEEE 7th Global Conference on Consumer Electronics (GCCE)*. IEEE, 2018, pp. 206–209.
[50] G. Bjontegarrd, "Calculation of average PSNR differences between RD-curves," *ITU - Telecommunications Standardization Sector STUDY GROUP 16 Video Coding Experts Group (VCEG)*, 2001.
[51] B. Bross, P. Helle, H. Lakshman, and K. Ugur, "Inter-picture prediction in HEVC," in *High Efficiency Video Coding (HEVC)*. Springer, 2014, pp. 113–140.
[52] F. Bossen *et al.*, "Common test conditions and software reference configurations," *JCTVC-L1100*, vol. 12, 2013.

BUDDHIPRABHA ERABADDA (S'16) received her B.Sc. (Eng) degree with honours (2014) and M.Sc. degree (2017) in Computer Science and Engineering from the University of Moratuwa, Sri Lanka. She is currently pursuing her Ph.D. degree in Electronic Engineering at the Centre for Vision, Speech, and Signal Processing, University of Surrey, United Kingdom.

From 2014-2015 she was a Software Engineer at a finance product company in Colombo, Sri Lanka. From 2015-2017 she was a Research Assistant at the Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka. Her research interests include data science and application of machine learning techniques for video coding.

THANUJA MALLIKARACHCHI (S'12-M'17) received his B.Sc. (Eng.) degree with honours in Electronic and Telecommunication Engineering from University of Moratuwa, Sri Lanka in 2011. From 2011 to 2013, he was a Senior Engineer at Virtusa (pvt) Ltd., Colombo, Sri Lanka. He received his Ph.D. degree in Electronic Engineering from the Centre for Vision, Speech and Signal Processing (CVSSP) at the University of Surrey, United Kingdom, in 2017. From 2017 to 2018, he was a Research Fellow in Multimedia Communication at CVSSP. Currently, he is a Lecturer in Data Science and Informatics at the Cardiff School of Technologies, Department of Applied Computing and Engineering, Cardiff Metropolitan University, Cardiff, United Kingdom.

His research interests are in the areas of video coding, video communication and video processing.

GOSALA KULUPANA received his B.Sc. (Eng.) degree with honours from University of Moratuwa, Sri Lanka in 2011. From 2011 to 2014, he was an Engineer at Mobitel (pvt) Ltd., Sri Lanka. He completed his Ph.D. in HEVC video coding in 2017 at the Centre for Vision, Speech and Signal Processing (CVSSP), University of Surrey, United Kingdom. Then he worked as a Research Fellow at the CVSSP, before joining the BBC R&D team, UK as a Research Engineer in 2018.

His research interests are in the areas of Intra Prediction, 360° video coding, VVC, HEVC and resource optimisation.

ANIL FERNANDO (S'98-M'01-SM'03) received the B.Sc. Engineering degree (First class) in Electronic and Telecommunications Engineering from the University of Moratuwa, Sri Lanka in 1995 and the MEng degree (Distinction) in Telecommunications from Asian Institute of Technology (AIT), Bangkok, Thailand in 1997. He completed his PhD in video coding at the Department of Electrical and Electronic Engineering, University of Bristol, UK in Feb. 2001.

Currently, he is a reader in signal processing at the University of Surrey, UK. Prior to that, he was a senior lecturer in Brunel University, UK and an assistant professor in AIT. His current research interests include cloud communications, video coding, Quality of Experience (QoE), intelligent video encoding for wireless systems and video communication in LTE with more than 290 international publications on these areas. He is a senior member of IEEE and a fellow of the HEA, UK. He is also a member of the EPSRC College.