LEARNING-BASED COMPLEXITY REDUCTION AND SCALING FOR HEVC ENCODERS

Mateus Grellert[*], *Guilherme Correa*[†], *Bruno Zatt*[†], *Sergio Bampi*[*], *Luis A. da Silva Cruz*[‡]

[*]Graduate Program in Computing, Federal University of Rio Grande do Sul, Brazil
[†]Video Technology Research Group (ViTech), Federal University of Pelotas, Brazil
[‡]Dept. of Elect. and Comp. Engineering / Inst. de Telecomunicações, University of Coimbra, Portugal

ABSTRACT

This article proposes a fast Coding Unit (CU) partition decision for use in HEVC encoders based on Decision Tree classifiers. The trees are employed in a modified low-complexity encoder that implements a fast CU partition decision algorithm. Using the proposed method, an average complexity reduction of 47.8% is achieved with a Bjontegaard Delta bitrate (BD-BR) loss of 0.24% in the Random Access coding configuration, and a 42.8% complexity reduction with a 0.19% BD-BR loss in the Low Delay B configuration. A decision threshold analysis is also presented to assess the rate-distortion-complexity trade-off of the proposed method at different complexity points, varying the complexity reduction from 28% (with a 0.04% loss in BD-BR) up to 60% (with a 3.6% BD-BR loss) using the Random Access configuration. A comparison with related works shows that the proposed method outperforms competing solutions in terms of both rate-distortion efficiency and complexity reduction.

***Index Terms*—** HEVC, machine learning, fast decision, decision trees

1. INTRODUCTION

The High Efficiency Video Coding (HEVC) standard [1], developed by a group of video-coding experts known as the Joint Collaborative Team on Video Technology (JCT-VC), outperforms the coding efficiency of the established H.264/AVC by 39.3% in terms of Bjontegaard Delta bitrate (BD-BR) [2][3]. However, to accomplish that the computational requirements of HEVC are also higher. As stated in [4], the HEVC encoder requires from 20% to 50% more computations to compress videos when compared to H.264/AVC.

In HEVC, frames are divided into blocks called Coding Tree Units (CTU), which can be further divided into Coding Units (CUs). The CU partition decision follows a recursive four-fold subdivision, forming what is called a CTU quadtree. In the root node of this quadtree (depth 0), the 64×64 CU block is evaluated and then it is divided into four 32×32 sub-blocks. This process is repeated until 8×8 blocks are formed (i.e., depth 3 is reached). The prediction mode decision is performed inside each quadtree node, which are candidate CUs. HEVC defines up to eight partitions (called Prediction Units – PUs) to further increase the flexibility of mode decision. Motion Estimation (ME), Skip/Merge mode, and intra prediction are all evaluated for each possible PU belonging to a CU. In the next encoding stage, where transforms are applied to convert prediction residues from the spatial to the frequency domain, the residue blocks are once again recursively divided, from 32×32 down to 4×4 blocks. This introduces a Residual quadtree (RQT) of transform blocks, which is nested inside each node of the CTU quadtree. Fig. 1 displays a CTU block and its selected partitions represented as blue-filled blocks.

The Rate-Distortion Optimization (RDO)-based mode decision is one of the main contributors to HEVC complexity, as ME and transforms are computed several times in each CTU quadtree node. Therefore, several works found in the literature address this problem by designing fast mode decision algorithms. Most of these references make use of statistics-based heuristics for fast mode decision, including [5], [6] and [7]. In recent studies, machine learning techniques have also been employed to speed up the mode decision process, using Decision Trees (DT) [8], Bayesian Decision Rule [9], and Support Vector Machines [10][11]. When both encoding time reduction and compression efficiency are considered, the results achieved by learning-based methods outperform statistics-based approaches.

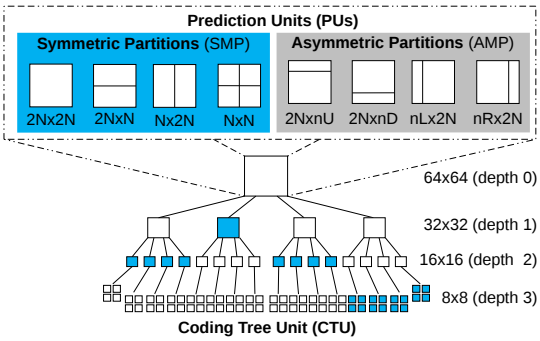


Fig. 1: HEVC mode decision for CU/PU partitioning. White-filled blocks are evaluated, but not used to encode the data.

This work is part of the project FAVIETOL supported by Instituto de Telecomunicações, FCT UID/EEA/50008/2013.

This paper presents a fast algorithm for CU partitioning decision using Decision Tree classifiers. The proposed classifiers are trained offline and employed in a modified HEVC encoder that uses the decision function to determine if the current CU should be further partitioned or if the partitioning can be terminated prematurely. This study follows after the work presented in [8], showing that better features can improve the efficiency and the generalization of classifiers. The following contributions are presented in this paper:

- A low-complexity HEVC encoder based on classifiers that employ new coding features relevant to the CU partition decision, allowing rate-distortion-complexity trade-off beyond current state-of-the-art techniques;
- A complexity scalability strategy for HEVC that uses classifiers' thresholds to achieve optimized rate-distortion-complexity trade-off for different operation points.

The remainder of this paper is as follows. Section 2 shows an analysis of the features used in this work. Section 3 explains how machine learning was used to reduce HEVC complexity and the decision threshold technique for complexity scalability. Section 4 presents results and comparisons with related work. Finally, section 5 concludes the paper.

2. HEVC FEATURE ANALYSIS

The encoding process yields much information that can be used as input for a CU partition classifier, but at the end only part of it will be useful and relevant to the partitioning decision. In [8], a set of 10 features was used to build a set of models, including values that can be obtained during encoding, such as the skip flag, the rate-distortion (RD) costs, and the PU mode decision in the current CU. In addition, new features were designed by combining two or more values, which are presented in the following equations:

$$Ratio_{m1,m2} = \frac{RDCost_{m1}}{RDCost_{m2}} \quad (1)$$

$$NormDiff_{m1,m2} = \frac{|RDCost_{m1} - RDCost_{m2}|}{RDCost_{m2}} \quad (2)$$

$$\Delta NeighDepth = depth_{curr} - AvgDepthCtx \quad (3)$$

$$AvgDepthCtx = \frac{\sum^{N} AvgDepth(N)}{N} \quad (4)$$

$$AvgDepth = \frac{\sum^{M} DepthCU(M)}{M} \quad (5)$$

In (1), the Ratio between the RD costs obtained from modes $m1$ (e.g., 2N×2N) and $m2$ (e.g., Merge/SKIP mode – MSM) is computed, and its normalized version is computed in (2). In (3), the average depth of the neighboring CTUs is first computed with (4), using the depth of its *M* constituent CUs, as in (5). Then, the average of these averages is computed and subtracted from the depth of the CU being currently encoded, yielding the Δ*NeighDepth* value. The neighboring CTUs include four spatial neighbors and up to two temporal neighbors (one from each reference list), so up to 6 neighbors are used in the calculation of Δ*NeighDepth*.

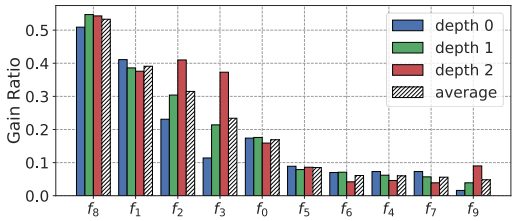
Table 1: Features introduced in this work

Id.	Description	Id.	Description
f_{11}	Best prediction mode	f_{28}	Coded Block Flag (CBF)
f_{12}	Total encoded bits	f_{40}	co-located CU split flag
f_{17}	Total distortion	f_{41}	upper CU split flag
f_{22}	NormDiff$_{Best,2N\times2N}$	f_{42}	left CU split flag
f_{23}	Ratio$_{Best,MSM}$	f_{51}	up-right CU split flag
f_{24}	NormDiff$_{Best,MSM}$	f_{52}	AvgDepthCtx$_{CU}$
f_{27}	RQT depth	f_{53}	ΔNeighDepth$_{CU}$

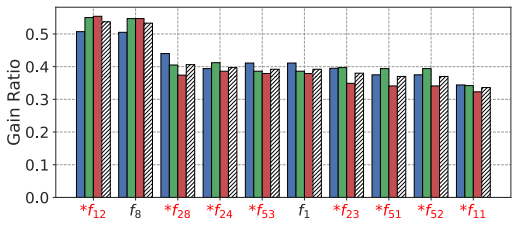
In this work, 44 new features were introduced, yielding a set of 54 features with the ones used in [8]. The most relevant of these 44 new features in terms of Gain Ratio (GR) are presented in Table 1. In the remainder of this paper, the features with indices between 0 and 9 will refer to those of [8], whereas features with indices above 9 represent the ones introduced in this work. Most features listed in Table 1 are extracted directly from the encoding process, except for f_{22}, f_{23} and f_{24}. The *AvgDepthCtx*$_{CU}$ value is a variation of (4) that uses only the depth of neighboring CUs instead of the average CTU depth. Similarly, Δ*NeighDepth*$_{CU}$ is computed using *AvgDepthCtx*$_{CU}$ instead of *AvgDepthCtx*.

The importance of each extracted feature was assessed in terms of GR, because the C5.0 algorithm uses this metric to rank features by their importance during the tree building phase. Given a set of feature vectors X and their respective labels Y, the GR of a feature f is the ratio between two values: (1) the Information Gain, which measures the entropy reduction of Y given the information of X_f, and (2) the Intrinsic Value, which measures the potential information generated by splitting the training data into each value of X_f. Fig. 2(a) shows the GR obtained using the set of features from [8] and Fig. 2(b) shows the features with the ten highest GRs using the extended set proposed in this work. It is possible to see that the features introduced in this work are more relevant than most of the features used by [8]. The total encoded bits (f_{12}), introduced in this work, achieved the highest GR. This value is used to compute the RD cost, so it is tightly related to the final RDO decision. The third best feature (CBF, f_{28}), also introduced in this work, is a flag that is set to false when the residual block has no significant coefficients (increasing the compression rate), so it is also reasonable that this feature is important for the split/unsplit prediction.

The split probability distributions of the two best features (in terms of GR) introduced in this work are presented in Fig. 3. In Fig. 3(a), it is possible to perceive that the unsplit CUs are much more frequent when the CBF flag is off, whereas the opposite occurs when it is on. In Fig. 3(b), it is possible to see that the probability of a CU not being split is very high only when the amount of encoded bits is close to zero. The greater GR of the introduced features indicates that the classifiers trained with the proposed features will perform better. The training results and the implementation of the obtained models are discussed in the following sections.

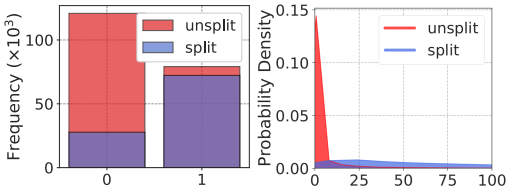


(a) Previous features



(b) Proposed features

Fig. 2: Gain Ratio of the (a) previous [8] and (b) proposed features for each depth.



(a) Coded Block Flag (f_{28}) (b) Encoded Bits (f_{12})

Fig. 3: Split probability distributions of the two best features in the proposed set (depth = 2).

3. CU SPLITTING BASED ON DECISION TREES

The decision tree models that employ the features derived in the previously section were trained with data collected from the HEVC Model (HM) encoder (version 16.8) for ten video sequences with various resolutions, namely: *Traffic*, *NebutaFestival* (2560×1600 pixels); *BasketballDrive*, *Parkscene* (1920×1080); *SlideShow*, *Vidyo1* (1280×720); *BQMall*, *PartyScene* (832×480); and *BlowingBubbles*, *RaceHorses* (416×240). The sequences were encoded with QP values of 22, 27, 32, 37 and 42, using the Random Access (RA) configuration, and the number of encoded frames per sequence was limited to 150. The features described in section 2 were collected for each CU, forming feature vectors that were labeled as split, when the cumulative RD cost of the four sub-CUs was smaller than the current one, or unsplit otherwise. After all data were collected, three training sets were built, one for each quadtree depth (0, 1 and 2). Then, each set was balanced to ensure the same distribution of split and unsplit cases using random under-sampling.

Table 2 shows the training results using the C5.0 algorithm [12], including accuracy, true positive rate (TPR), and max tree depth of each classifier, using 200,000 training vectors. To ascertain the efficiency of the proposed set of features, Table 2 compares the training accuracy of the classifiers trained with both data sets. Note that using more features increases the depth of the trees, but it represents a very small overhead when compared with the enormous amount of computations that are saved every time an early termination happens, as section 4 shows.

Table 2: Train accuracy, true positive rate, and tree depth of the trained classifiers (200,000 vectors per data set)

Feature set	depth	Accuracy	TPR	Tree Depth
previous [8]	d0	90.9%	89.5%	13
	d1	91.2%	89.1%	13
	d2	91.2%	88.8%	19
proposed	d0	92.7%	91.9%	20
	d1	93.6%	92.2%	24
	d2	94.2%	92.3%	21

3.1. Low-Complexity HEVC Encoder

The trained classifiers are used as input to a modified HEVC encoder that implements an early CU termination scheme, depicted in Fig. 4. Two methods were implemented in the HM software: (1) the feature extraction and (2) the decision function using the classifiers and the extracted features as input.

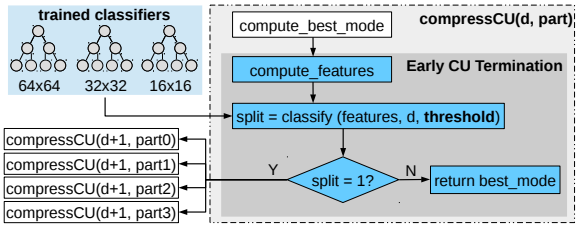


Fig. 4: Early CU termination scheme.

In Fig. 4, the best mode for the current CU is computed and then the feature extraction routine is called to generate the input vector for classification. The decision function is called in the next step, using the classifier trained for the corresponding CU size, and its output is tested. If the classifier outcome is *split*, the CU partitioning evaluation is continued. Otherwise, the splitting evaluation is terminated prematurely and the best mode found for the current CU is chosen.

3.2. Complexity-Scalable HEVC Encoder

The complexity reduction strategy presented in the previous paragraphs was employed in a complexity-scalable encoder implementation. This was achieved by including a split threshold ($Split_{th}$) parameter in the decision function, which leads to more or less split outcomes. First, the original decision is computed ($split$) along with its confidence factor (C). The confidence of a decision is always computed along with it in the C5.0 algorithm. Then, $split'$ is computed as:

$$split' = \begin{cases} 1 & \text{if } Split_{th} < 0.5 \text{ and } C < (1 - Split_{th}) \\ 0 & \text{if } Split_{th} > 0.5 \text{ and } C < Split_{th} \\ split & \text{otherwise} \end{cases}$$

When $Split_{th}$ is set to 0.5, the final output is exactly the same as the original classifier decision. When it is set below

0.5, any decision with a confidence below $1 - Split_{th}$ is reset to split, which increases the compression efficiency while decreasing the time savings. The opposite behavior occurs when the threshold is set above 0.5, favoring time savings over compression efficiency. A split threshold analysis and other comparisons are presented in the following section.

4. RESULTS AND DISCUSSION

To assess the RD performance of the proposed method, 13 sequences with varying resolutions were used, following the Common Test Conditions defined by the JCT-VC group [13]. To reduce the risk of bias, the set of sequences in this analysis is complementary to the one used in the training phase. The RD and complexity results of the test sequences are presented in Table 3. For these experiments, the default threshold was used in the decision function (i.e., $Split_{th}$ = 0.5). Time savings (TS) were computed as $TS = 1 - T_{test}/T_{ref}$, where T_{ref} is the encoding time of the original reference software (HM 16.8), and T_{test} is the encoding time with the proposed method, including the decision function calls. On average, the results in Table 3 show that the method is capable of reducing the HEVC encoding time in 47.8%, at the cost of a negligible compression efficiency loss of 0.24%. The rightmost column shows the ratio between RD efficiency and TS (BDTS), which is a commonly used metric to measure rate-distortion-complexity efficiency. On average, for every 1% in TS, a BD-rate increase of 0.0051% is required.

A comparison with related works is presented in Table 4, where *Previous* represents the work of [8], which was reimplemented in HM 16.8. The results prove that the new strategy outperforms [8] for all configurations. The performance difference is smaller for the Random Access (RA) configuration because the authors in [8] used RA data to train their models, but when other configurations (Low Delay P – LP, and Low Delay B – LB) are considered, the difference is more significant. These results show that the proposed classifiers generalize better than [8], since the training sets also use only RA data. When compared to the remaining related works, the results outperform them in BD-BR, TS and BDTS.

The final analysis presents the effect of the $Split_{th}$ parameter on rate-distortion-complexity efficiency. Several values for this parameter were tested, using the same sequences listed in Table 3. The results are summarized in Fig. 5. The chart shows that the $Split_{th}$ has significant impacts on both time savings and BD-BR. TS increases in a quasi-linear trend, whereas the BD-BR barely changes with a $Split_{th}$ up to 0.8, but grows rapidly with larger values. The smallest time savings achieved are 28% with negligible 0.04% increase in BD-BR, whereas the highest time savings of 60% are achieved at the cost of 3.6% in BD-BR. However, a smaller BD-BR increase of 0.61% is observed for an average savings of 51%, which is a much better trade-off. These results ultimately let us conclude that several complexity points can be achieved with minimum RD loss using the $Split_{th}$ threshold, which is useful for applications that require complexity scaling whereas still maintaining the encoding efficiency.

Table 3: Rate-distortion and complexity results of the proposed method (Random Access, QP = 22, 27, 32, 37)

Class	Sequence	BD-BR (%)	TS (%)	BDTS (×100)
A	PeopleOnStreet	0.28	23.0	1.21
	SteamLocomot.	0.32	53.9	0.59
B	Kimono	0.42	43.7	0.96
	BQTerrace	0.47	53.1	0.88
C	RaceHorsesC	0.14	20.8	0.68
	BasketballDrill	0.17	40.7	0.41
D	BasketballPass	0.16	43.1	0.37
	BQSquare	0.22	45.5	0.48
E	KristenAndSara	0.12	69.1	0.18
	Johnny	0.19	69.1	0.27
F	SlideEditing	0.01	73.3	0.01
	ChinaSpeed	0.44	37.8	1.17
Average		**0.24**	**47.8**	**0.51**

Table 4: Comparison with the related work

Cfg.	Method/Reference	BD-BR (%)	TS (%)	BDTS (×100)
RA	Bayesian [9]	0.71	34.9	2.03
	SVM [10]	1.35	44.7	3.02
	DT Previous [8]	0.32	47.2	0.67
	DT Proposed	**0.24**	**47.8**	**0.51**
LB	Bayesian [9]	0.63	32.6	1.93
	SVM [10]	1.66	41.9	3.02
	DT Previous [8]	0.46	40.8	1.12
	DT Proposed	**0.19**	**42.8**	**0.45**
LP	Bayesian [9]	0.62	32.7	1.89
	SVM [11]	2.66	59.9	4.44
	DT Previous [8]	0.39	38.1	1.03
	DT Proposed	**0.22**	**41.2**	**0.53**

5. CONCLUSION

This paper presented a complexity reduction method for HEVC encoders that employs a set of classifiers trained with the most relevant features extracted from the encoding process to the CU splitting decision. The features were chosen after an extensive Gain Ratio analysis and used to train three decision tree models with the C5.0 machine learning algorithm. The obtained trees were implemented in the HEVC encoder and allowed a complexity reduction of 47.8%, at the cost of a negligible BD-rate increase of 0.24%. Finally, the use of a confidence threshold for complexity scaling was also analyzed and the experimental results showed that the complexity reductions can be adjusted between 28% and 60%, with BD-rate increases varying between 0.04% and 3.6%.

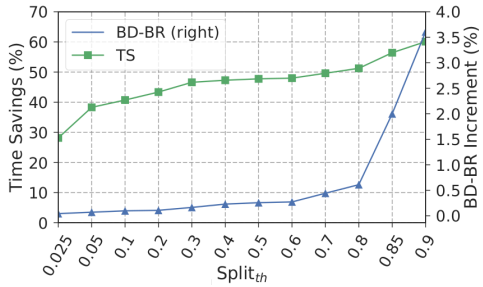


Fig. 5: TS and BD-BR for different values of $Split_{th}$

6. REFERENCES

[1] ITU-T, *Recomm. ITU-T H.265: High efficiency video coding*, 2016.

[2] D. Grois, D. Marpe, A. Mulayoff, B. Itzhaky, and O. Hadar, "Performance comparison of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC encoders," in *2013 Picture Coding Symposium (PCS)*, Dec 2013, pp. 394–397.

[3] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," Tech. Rep. VCEG-M33, Video Coding Experts Group from ITU-T, 2001.

[4] J. Vanne, M. Viitanen, T. D. Hamalainen, and A. Hallapuro, "Comparative rate-distortion-complexity analysis of HEVC and AVC video codecs," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1885–1898, Dec 2012.

[5] J. Kim, J. Yang, K. Won, and B. Jeon, "Early determination of mode decision for HEVC," in *2012 Picture Coding Symposium*, May 2012, pp. 449–452.

[6] X. Shen, L. Yu, and J. Chen, "Fast coding unit size selection for HEVC based on bayesian decision rule," in *2012 Picture Coding Symposium*, May 2012, pp. 453–456.

[7] 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, Feb 2014.

[8] G. Correa, P. A. Assuncao, L. V. Agostini, and L. A. da Silva Cruz, "Fast HEVC encoding decisions using data mining," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 4, pp. 660–673, April 2015.

[9] Hyo-Song Kim and Rae-Hong 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, 2016.

[10] Xiaolin Shen and Lu Yu, "CU splitting early termination based on weighted SVM," *EURASIP Journal on Image and Video Processing*, vol. 2013, no. 1, pp. 4, 2013.

[11] L. Zhu, Y. Zhang, Z. Pan, R. Wang, S. Kwong, and Z. Peng, "Binary and multi-class learning based low complexity optimization for HEVC encoding," *IEEE Transactions on Broadcasting*, vol. PP, no. 99, pp. 1–15, 2017.

[12] R. Quinlan, "Data mining tools see5 and c5.0," 2015.

[13] F. Bossen, "Common test conditions and software reference configurations," Tech. Rep. JCTVC-J1100, JCT-VC, 2012.