Fast CU Size Decisions for HEVC Inter-Prediction Using Support Vector Machines

Buddhiprabha Erabadda, Thanuja Mallikarachchi, Gosala Kulupana and Anil Fernando
Centre for Vision Speech and Signal Processing, University of Surrey, United Kingdom
Email: {e.harshani, d.mallikarachchi, g.kulupana, w.fernando}@surrey.ac.uk

***Abstract*—The brute force rate-distortion optimisation based approach used in the High Efficiency Video Coding(HEVC) encoders to determine the best block partitioning structure for a given content demands an excessive amount of computational resources. In this context, this paper proposes a novel algorithm to reduce the computational complexity of HEVC inter-prediction using Support Vector Machines. The proposed algorithm predicts the Coding Unit (CU) split decision of a particular block enabling the encoder to directly encode the selected block, avoiding the unnecessary evaluation of the remaining CU size combinations. Experimental results demonstrate encoding time reductions of ~58% and ~50% with 2.27%, and 1.89% Bjøntegaard Delta Bit Rate (BDBR) losses for Random Access and Low-Delay B configurations, respectively.**

***Index Terms*—High Efficiency Video Coding (HEVC), inter-prediction, Coding Unit (CU), Support Vector Machine (SVM), encoding complexity reduction**

I. Introduction

The recent advancements in multimedia technologies and mobile hand-held devices have contributed immensely towards the increasing mobile consumption of video. As such, the demand for video streaming services such as Netflix, Amazon, YouTube, etc., is expected to exceed 75% of the overall mobile data traffic by 2021 [1]. The increasing popularity of High Definition (HD) and Ultra-High Definition (UHD) video contents demand efficient video coding algorithms with greater compression efficiency in order to reduce the strain on the already congested communication networks.

In this case, High Efficiency Video Coding (HEVC) achieves ≈50% more coding efficiency when compared with its immediate predecessor H.264/AVC [2]. However, this increased coding efficiency of HEVC comes at the cost of high computational complexity. This is mainly due to the brute-force Rate-Distortion (RD) optimisation based approach used to find the optimal block partitioning structure for a given video content. The excessive demand of computational resources for HEVC has become a compelling challenge for resource and energy constrained consumer electronic devices such as mobile phones and tablets.

Addressing the aforementioned phenomenon, this paper proposes a Support Vector Machine(SVM) based algorithm that predicts the optimal block partitioning structure for a given content. The proposed approach replaces the time-consuming brute force evaluation of partitioning structures using RD optimisation, thereby significantly reduces the encoding time complexity with minimal impact on the coding efficiency.

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

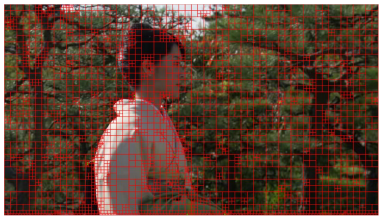

Fig. 1. Partitioning structure for a frame in *Kimono*

The remainder of this paper is organised as follows. Section II provides an overview of the quadtree based block partitioning structure employed in HEVC. Section III describes the related work on computational complexity reduction in HEVC inter-prediction. Section IV introduces the proposed algorithm followed by Section V, presenting the experimental results and discussion. Section VI concludes the paper.

II. Quadtree Block Partitioning Structure in HEVC

HEVC employs a block-based encoding scheme similar to that of H.264/AVC, where each frame is divided into square-shaped blocks. However, the main processing block of HEVC, also known as Coding Tree Unit (CTU), uses a block size of 64×64, which facilitates efficient encoding of high resolution video content compared to H.264/AVC. A CTU can be further sub-divided recursively into four equally sized blocks known as Coding Units (CU). The size of a CU can vary from 64×64 to 8×8, corresponding to depth levels from 0 to 3, respectively.

During the compression loop of the encoder, for a given CTU, all possible CU depths are analysed to compute their respective RD costs. The depth levels that yield the minimum RD cost are selected as the best CU size combination for the content. For example, when analysing a CU of the size 32×32 (corresponding to depth 1), the cost of encoding the CU as a single CU and the cost of encoding the CU as four equally divided sub-CUs of size 16×16 (corresponding to depth 2) are calculated. The CU is labelled as a "split" CU if the RD cost of the four sub-CUs at depth 2 is smaller than RD cost of the depth 1 CU. Otherwise, the CU is identified as a "non-split" CU and encoded with the depth level 1 (i.e., CU size = 32×32). This partitioning takes place for all CTUs in a given

978-9-0827-9703-9/19/$31.00 ©2019 IEEE

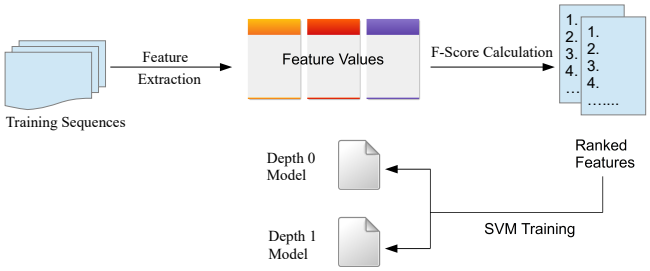

Fig. 2. Process of data collection and model building

frame. Fig. 1 depicts an example partitioning structure for a frame in the *Kimono* video sequence.

III. Related Work

To address the problem of high computational complexity in HEVC, recent literature has proposed alternative algorithms that avoid the brute-force RD optimisation approach. These algorithms can be broadly categorised into two: statistical approaches and learning-based approaches. Statistical approaches are based on conclusions drawn from statistics-based models, whereas learning-based approaches use machine learning techniques to predict the optimal coding tree structure.

Statistical model based approaches are widely used in the encoding complexity reduction, due to the simplicity of the prediction models. For instance, Lee *et al.* [3] propose an algorithm based on RD cost statistics of *Inter* $2N \times 2N$ mode and the status of *SKIP* and *merge* modes. Furthermore, Mallikarachchi *et al.* [4] propose a content-adaptive approach for fast low-delay video encoding that makes use of *Inter* $N \times N$ mode motion features and RD cost thresholds. Despite the reported encoding complexity reductions in the range of 55%, the inadequacies in the amount of data accumulated for certain features make the split decisions less optimal for certain video contents. The coding tree pruning algorithm proposed by Choi and Jang [5] terminates CU splitting if the current CU level has selected *SKIP* mode as the best prediction mode. Despite the low BDBR losses, the encoding time reductions that can be achieved is minimal (40%) due to the large number of unnecessary evaluations of CU depth levels. Similar algorithms that make use of various statistical parameters available in the encoding loop are reported in [6]–[8]. However, the common drawbacks of these methods can be identified as rigid models and redundant calculations that limit the time complexity reductions that can be achieved.

Recent advancements of machine learning have enabled the successful use of supervised learning methods in video coding research. As opposed to statistics-based approaches that rely on extensive human involvement to create rule-based methods, machine learning-based models learn from existing data with minimal human involvement.

Grellert *et al.* [9] propose a flexible SVM based approach to early terminate the CU evaluation process in the encoding loop. Features calculated after encoding the current depth level are used to determine when the evaluations for the next CU depth should be continued or not. The authors report a 52.4% time complexity reduction for *Random Access* configuration, when compared with the HM16.8 reference software. However, calculations for the current depth level are carried out for feature calculations, leading to redundant calculations when the CU is decided to be split. Following a similar approach, Shen and Yu [10] propose a complexity reduction algorithm that uses weighted SVMs. The RD costs of *SKIP/MERGE* and *Inter* $2N \times 2N$ modes are considered as features for the off-line trained weighted SVMs which are used to determine the early termination of the CU evaluation process. This method uses rigid models for prediction of CU and requires encoding information from the current CU depth. The decision tree based CU size selection algorithm proposed by Correa *et al.* [11] suffers from content adaptation issues due to the rigid decision trees. Similarly, the use of weighted SVMs in the algorithm proposed by Zhang *et al.* [12] achieves an average encoding time reduction of approximately 51%. However, this method also needs encoding information of the current depth level as features to the models. In addition, different machine learning techniques have been heavily used in [13]–[16] to reduce the encoding time complexity in HEVC.

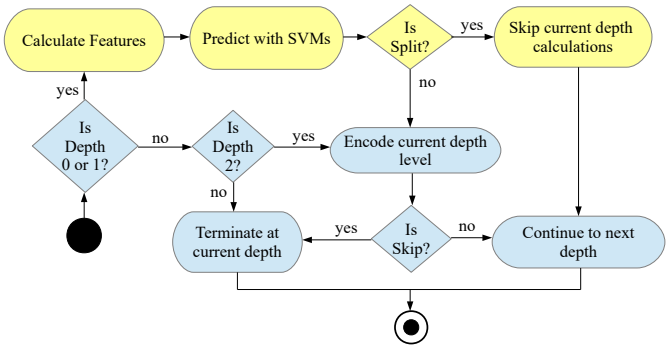

Fig. 3. Two-stage prediction model for a given CU depth

IV. Proposed Method

A. Problem Formulation

As explained in Section II, at each CU depth level it is decided either to split the CU or to encode the CU in the current size. This formulates a *binary classification* problem with classes *split* and *non-split* at each CU depth level: 0, 1, and 2. Since the minimum CU size is 8×8 that corresponds to depth 3, there is no requirement to take the split decision at CU depth 3.

There are many classification algorithms available in machine learning, such as SVMs, decision trees, and neural networks. The problem of time complexity reduction in HEVC inter-prediction relies on the fact that the model prediction time to be considerably less than the brute-force RD optimisation time. SVMs can handle binary classification problems with significant computational advantages [17]. Hence, in this work, SVMs were used.

Fig. 2 and Fig. 3 depict the model building and the two-stage prediction process, respectively. The following sub-sections explain the components in detail.

TABLE I
F-SCORE VALUES FOR DEPTH 0

Feature	F-Score
CTU A-avg. depth	0.1862
CTU A-max. depth	0.1842
Coloc. CU-max. depth	0.1833
Coloc. 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 II
F-SCORE VALUES FOR DEPTH 1

Feature	F-Score
Coloc. CU-max. depth	0.1932
Coloc. 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

B. Early Split and Early Termination of CUs

As depicted in Fig. 3, the proposed algorithm operates in two stages: (1) prior to (blocks in *yellow*) and; (2) after (blocks in *blue*) the encoding of the current depth level. In the first stage, offline-trained SVMs are used, whereas in the second stage, it is checked if the *SPLIT* mode is selected as the best mode in the current depth level. Our initial analysis showed that using offline models at depth 2 significantly degraded the encoding efficiency. Hence, the first stage offline SVMs operate only for depths 0 and 1, while the second stage operates in all three levels, i.e., 0, 1, and 2.

1) Training Data Collection: To decide on the optimal feature set for the SVMs, a feature analysis was carried out. The sample data were gathered from a set of training sequences that have resolutions ranging from 416×240 to 2560×1600. These training sequences include *BlowingBubbles(*416×240*)*, *BQMall(*832×480*)*, *Kimono(*1920×1080*)*, and *Traffic(*2560×1600*)*. Data were gathered by encoding twenty frames from each sequence for $\mathrm{QP} \in \{22, 27, 32, 37\}$.

All training and cross-validation sets were randomly taken and they included equal numbers of data from each sequence in order to ensure that the data samples were representative of all the sequences used in training. Since SVMs perform poorly for unbalanced data [18], equal numbers of instances for *split* and *non-split* classes were taken.

2) Feature Selection: A multitude of possible features were analysed in order to find the best features for the split decision prediction for CU depth levels 0 and 1.

The set of features used for the analysis included information from the neighbouring CTUs (CTU above, CTU left, CTU above right, and CTU above left), colocated CU, and features calculated from the current CU. Specifically, the set of features considered include:

- Depth information from the neighbouring CTUs
- Depth information from the colocated CU
- current quantization parameter (QP) value
- current CU RD-cost for the size $2N \times 2N$
- texture information from current CU (homogeneity, contrast, and energy)
- local range values (mean, median, and standard deviation)
- Encoding information from the neighbouring CTUs

The F-score was calculated for all these features, in order to select the best features. F-score is given by;

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

where $\bar{x}_i$, $\bar{x}_i^+$, $\bar{x}_i^-$, P, and N refer to the average of the ith feature, average of the positive instances, average of the negative instances, the number of positive instances, and the number of negative instances, respectively. $x_{n,i}^+$ and $x_{n,i}^-$ refer to the values of feature i in nth positive and negative instances, respectively.

Tables I and II depict the F-score values for the depths 0 and 1, respectively. In the tables, CTU A, CTU L, and CTU AR refer to CTUs Above, Left, and Above Right to the current CU, respectively.

Upon the analysis of the F-score values of the aforementioned features, the following observations were made.

- for the two depth levels, different sets of features scored highest f-score values
- texture information of the current CU contributed less in the inter-prediction CU split decision
- depth values of the neighbouring and colocated CUs scored high in both depth levels, indicating its importance in inter-prediction CU split decision
- QP value was among the top features for both depths (fifth feature in depth 0 and third feature in depths 1)

From the above observations, features were selected to train one SVM per depth level. The number of features selected for each model was limited to 4, in order to minimise the additional overhead required to obtain the feature values for the SVMs.

The features that gave highest F-score values were selected for both depth levels. When both average and maximum depths of a given neighbour were among the top features, only the feature that scored higher was taken. Table III shows the features that were selected for the SVMs.

3) Model Training and Hyper Parameter Tuning: Radial Basis Function(RBF) can handle non-linear decision boundaries and it works well with small numbers of feature [19]. Therefore, RBF was selected as the kernel function for SVMs in this work.

The associated hyper-parameters of SVM with RBF kernel; cost parameter(C) and gamma parameter(G), were tuned using a grid search. Values for the multiples of 2, ranging from 2^{-7} to 2^7 were analysed to arrive at the optimal values.

The number of training samples(N_{Tr}) was also regarded as a hyper-parameter. A range of values was checked for the N_{Tr} at both depth levels. The number of support vectors increase

TABLE III
SELECTED FEATURES FOR THE DEPTH LEVELS

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

TABLE IV
COMPLEXITY REDUCTION PERFORMANCE CODING EFFICIENCY (LOW DELAY B)

Sequence	Proposed (t=0.7) vs HM			Proposed (t=0.6) vs HM			Grellert *et al.* [9] ($threshold$=0.5) vs HM			Grellert *et al.* [9] ($threshold$=0.7) vs HM			Mallikarachchi *et al.* [4] 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)
PeopleOnStreet	26.99	0.37	-0.02	43.63	1.51	-0.06	21.77	0.93	-0.04	29.29	2.91	-0.13	25.07	0.86	-0.03
BQTerrace	45.43	0.42	-0.01	52.28	2.46	-0.03	47.84	0.57	-0.01	51.68	1.08	-0.02	41.38	0.92	-0.01
Cactus	40.28	0.74	-0.02	50.65	3.26	-0.06	44.38	1.51	-0.03	51.12	3.38	-0.07	31.75	0.99	-0.02
ParkScene	47.28	0.51	-0.02	55.34	2.21	-0.07	46.60	0.76	-0.02	48.95	1.81	-0.05	42.53	1.07	-0.03
BasketBallDrill	30.71	0.32	-0.01	40.62	1.98	-0.08	37.09	1.04	-0.04	44.29	3.31	-0.12	34.32	0.88	-0.03
RaceHorses(C)	24.67	0.37	-0.01	38.69	1.82	-0.07	22.83	0.67	-0.02	31.83	2.24	-0.08	30.10	0.81	-0.03
PartyScene	30.10	0.26	-0.01	42.33	1.24	-0.05	29.21	0.51	-0.02	37.27	1.48	-0.06	32.36	0.89	-0.03
BasketBallPass	29.07	0.32	-0.01	36.01	0.84	-0.04	25.63	0.90	-0.04	32.34	2.48	-0.12	27.93	0.77	-0.04
BQSquare	29.84	0.21	-0.01	38.93	1.14	-0.04	34.26	0.50	-0.02	40.03	1.53	-0.05	31.05	1.08	-0.03
Johnny	66.35	0.22	-0.00	67.81	1.97	-0.03	68.73	0.94	-0.02	70.18	1.67	-0.04	56.56	1.69	-0.04
FourPeople	60.71	0.42	-0.01	64.25	2.43	-0.07	62.75	1.05	-0.04	65.38	2.56	-0.08	52.26	1.40	-0.04
KristenAndSara	61.75	0.16	-0.01	66.39	1.86	-0.05	65.13	0.42	-0.01	67.76	1.41	-0.04	52.81	1.78	-0.05
Average	**41.10**	**0.36**	**-0.01**	**49.74**	**1.89**	**-0.05**	**42.19**	**0.82**	**-0.03**	**47.51**	**2.16**	**-0.07**	**38.18**	**1.10**	**-0.03**

TABLE V
COMPLEXITY REDUCTION PERFORMANCE CODING EFFICIENCY (RANDOM ACCESS)

Sequence	Proposed (t=0.7) vs HM			Proposed (t=0.6) vs HM			Grellert *et al.* [9] ($threshold = 0.7$) vs HM			Mallikarachchi *et al.* [4] vs HM		
	ΔT(%)	BD-Rate (%)	BD-PSNR(dB)	ΔT (%)	BD-Rate (%)	BD-PSNR(dB)	ΔT (%)	BD-Rate (%)	BD-PSNR(dB)	ΔT (%)	BD-Rate (%)	BD-PSNR(dB)
PopleOnStreet	36.10	0.84	-0.04	56.62	1.28	-0.02	23.85	2.23	-0.09	25.66	0.79	-0.03
BQTerrace	56.62	1.28	-0.02	62.37	3.44	-0.04	51.43	1.11	-0.02	41.82	0.62	-0.01
Cactus	49.35	1.03	-0.02	57.43	3.71	-0.07	51.81	1.80	-0.04	40.09	0.58	-0.01
ParkScene	53.38	0.82	-0.02	60.03	2.85	-0.08	54.66	0.91	-0.03	43.33	0.73	-0.02
BasketBallDrill	49.54	0.68	-0.03	57.32	2.84	-0.11	46.28	1.59	-0.06	46.28	1.59	-0.06
RaceHorses(C)	32.54	0.77	-0.03	46.69	2.50	-0.09	34.25	1.77	-0.06	30.20	0.95	-0.03
PartyScene	41.64	0.75	-0.03	48.42	1.50	-0.06	42.97	1.17	-0.05	31.63	0.56	-0.02
BasketBallPass	30.66	0.67	-0.03	42.54	1.24	-0.06	35.60	1.84	-0.08	30.93	0.61	-0.03
BQSquare	46.44	0.56	-0.02	52.96	1.16	-0.04	49.55	0.78	-0.03	45.28	0.57	-0.02
Johnny	72.83	0.49	-0.01	75.12	2.50	-0.06	72.22	0.45	-0.01	56.12	0.33	-0.01
FourPeople	69.67	0.43	-0.02	69.79	1.82	-0.06	68.40	0.86	-0.03	52.44	0.35	-0.01
KristenAndSara	68.89	0.52	-0.02	71.07	2.42	-0.07	70.76	0.74	-0.02	54.56	0.48	-0.01
Average	**50.64**	**0.74**	**-0.02**	**58.36**	**2.27**	**-0.06**	**50.15**	**1.27**	**-0.04**	**41.53**	**0.68**	**-0.02**

TABLE VI
HYPER-PARAMETER VALUES OF THE SVMS

Parameter	**Depth 0**	**Depth 1**
Cost Parameter (C)	4	64
Gamma Parameter (G)	2^{-7}	0.5
Number of training Data	400	600

with the size of the training data, which leads to higher SVM prediction time. In order to ensure this does not affect the speed of SVM prediction, a maximum of 1200 training samples were tested for N_{Tr}. A separate cross-validation set consisting of 600 data samples was used at each depth level. The model accuracy was used to determine the best hyper-parameter values and the selected hyper-parameters following the grid search are shown in Table VI.

4) Predicting the CU size using Support Vector Machines and SKIP *mode:* When encoding a CU at either depth level 0 or 1, the corresponding features are retrieved during the encoding process. These feature values are sent to the SVM to predict the decision for the CU. If the model predicts to *split* the CU, current level calculations are skipped.

If the model prediction decision is *non-split* or when the CU depth is 2, the current depth computations are performed. If the best mode for the current level is *SKIP*, further processing of the CU is terminated. This is because, *SKIP* mode corresponds to static image regions with no residual [20].

5) Computational complexity and coding efficiency trade-off : The proposed method allows to tune the decision threshold of SVMs(t) to trade-off the computational complexity with the coding efficiency. This gives the flexibility to achieve the required levels of complexity reduction and coding efficiency depending on the application.

V. EXPERIMENTAL RESULTS

The proposed algorithm was implemented in HM 16.8 [21] using the optimised SVM library *LibSVM* [19] for SVM implementations. Computational complexity reductions for *Low-Delay B* and *Random Access* configurations were analysed to present the performance of the proposed method, following the HEVC common test configurations [22].

The experimental results are presented for the test sequences identified in [22] that cover a range of texture and motion complexities. However, the sequences used for SVM training are not presented in the evaluation results. The performance of the algorithm is compared with two state-of-the-art methods.

The algorithm proposed in [9] represents a machine learning based approach whereas the algorithm in [4] represents a statistical approach in the recent literature. The time complexity reduction $\Delta T(\%)$ is evaluated with,

$$\Delta \mathrm{T}(\%) = \frac{(T_{HM} - T_i)}{T_{HM}} \times 100, \tag{2}$$

where T_{HM}, and T_i are the encoding times of the HM reference encoder and the corresponding algorithm, respectively, for QP $\in \{22, 27, 32, 37\}$. The coding efficiency loss is measured using the Bjøntegaard Delta Bit Rate(BDBR) increase [23].

Table IV gives results for two values of the complexity control parameter, $t \in \{0.6, 0.7\}$ for *Low-Delay B* configuration. It can be observed that the proposed method with t=0.7 gives 41.1% time complexity reduction with a negligible BDBR loss of 0.36%. When t=0.6, it gives higher time complexity reduction of 49.74% with an increased BDBR loss of 1.89%. As it can be observed, the parameter t gives the flexibility to the user to trade-off the computational complexity with the coding efficiency. The proposed method with t=0.7 outperforms [9] with less BDBR loss at approximately equal time complexity reduction, when threshold for [9] is set at 0.5. With t=0.6, the proposed method has increased time gain, while [9](with threshold=0.7) has less time gain at higher BDBR loss than those of the proposed method. When t=0.6, the proposed method outperforms [4] both in terms of complexity reduction and BDBR. Further, the method in [4] does not offer flexibility to trade-off computational complexity and coding efficiency.

Table V gives results for the *Random Access* configuration. It can be observed that the proposed method with t=0.7 outperforms both state-of-the-art methods with a negligible coding loss. Furthermore, when t=0.6 the proposed method reports a significant time complexity reduction of 58.36% at a negligible BDBR loss of 2.27%.

From the overall performance evaluation, it can be observed that the proposed method can achieve higher time complexity reductions at less BDBR losses when compared with the benchmark methods.

VI. Conclusion

This paper introduced a novel complexity reduction algorithm for HEVC inter-prediction, replacing the brute-force RD optimisation approach. The experimental results based on *Random Access* and *Low Delay B* configurations show significant time complexity reductions outperforming the state-of-the-art methods with negligible BDBR losses. Additionally, the proposed method also allows the user to trade-off the computational complexity with the coding efficiency by adjusting the SVM decision threshold t. This gives flexibility to the user to achieve required levels of time complexity reduction and BDBR loss depending on the application.

The future work will focus on exploiting the possibility to add a model to depth 2 for increased time gains, while maintaining the coding efficiency. Further, possibility of extending the work for other coding structures such as Prediction Units and Transform Units will also be considered.

References

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

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

[3] J. 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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. 63, no. 3, pp. 547–561, 2017.

[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] L. Zhu, Y. Zhang, S. Kwong, X. Wang, and T. Zhao, "Fuzzy SVM-based coding unit decision in HEVC," *IEEE Transactions on Broadcasting*, vol. 64, no. 3, pp. 681–694, 2018.

[16] A. Mercat, F. Arrestier, M. Pelcat, W. Hamidouche, and D. Menard, "Machine Learning Based Choice of Characteristics for the One-Shot Determination of the HEVC Intra Coding Tree," in *2018 picture coding symposium (PCS)*. IEEE, 2018, pp. 263–267.

[17] 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.

[18] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intelligent data analysis*, vol. 6, no. 5, pp. 429–449, 2002.

[19] 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.

[20] 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.

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

[22] F. Bossen *et al.*, "Common test conditions and software reference configurations," *JCTVC-L1100*, vol. 12, 2013.

[23] G. Bjontegarrd, "Calculation of average PSNR differences between RD-curves," *ITU - Telecommunications Standardization Sector STUDY GROUP 16 Video Coding Experts Group (VCEG)*, 2001.