ACCELERATION OF KVAZAAR HEVC INTRA ENCODER WITH MACHINE LEARNING

Alexandre Mercat, Ari Lemmetti, Marko Viitanen and Jarno Vanne

Tampere University
Korkeakoulunkatu 10, Tampere, 33720, Finland
{alexandre.mercat, ari.lemmetti, marko.viitanen, jarno.vanne}@tuni.fi

ABSTRACT

The complexity of High Efficiency Video Coding (HEVC) poses a real challenge to HEVC encoder implementations. Particularly, the complexity stems from the HEVC quad-tree structure that also has an integral part in HEVC coding efficiency. This paper presents a Machine Learning (ML) based technique for pruning the HEVC quad-tree without deteriorating coding gain. We show how ML decision trees can be used to predict a depth interval for a quad-tree before the Rate-Distortion Optimization (RDO). This approach limits the number of RDO candidates and thus speeds up encoding. The proposed technique works particularly well with high-quality video coding and it is shown to accelerate the veryslow preset of practical Kvazaar HEVC intra encoder by 1.35× with 0.49% bit rate increase. Compared with the corresponding preset of x265 encoder, Kvazaar is 2.12× as fast at a cost of under 1.21% bit rate overhead. These results indicate that the optimized Kvazaar is the leading open-source encoder in high-quality HEVC intra coding.

Index Terms— High Efficiency Video Coding (HEVC), Intra Encoder, Machine Learning (ML), Complexity Reduction, Quad-Tree

1. INTRODUCTION

Nowadays, numerous applications encode and stream video content. Cisco [1] reports that 75% of total IP traffic is dedicated to video in 2017 and estimates it to grow to 82% by 2022. The latest international video coding standard, High Efficiency Video Coding (HEVC) [2], is developed to address this growth. HEVC is published as twin text by ITU, ISO, and IEC as ITU-T H.265 | ISO/IEC 23008-2. When compared with the previous MPEG AVC standard, HEVC Main profile reduces the bit rate by 50% on average for similar objective video quality [3, 4] but at a cost of over five times as high encoding complexity [5]. This overhead stems mostly from the new quad-tree block partitioning scheme, which exponentially increases the execution time of Rate-Distortion Optimization (RDO) process in HEVC.

As shown in our previous work [6], quad-tree partitioning of Coding Tree Unit (CTU) has a potential to reduce energy up to 78% in a practical software HEVC intra encoder. Previous studies on low complexity quad-tree partitioning can be classified into two categories: 1) the early termination mechanisms which dynamically terminate processing during the RDO process when future gains are unlikely; and 2) the prediction-based complexity reduction techniques which are applied before the RDO process to predict the quad-tree partitioning with lower complexity than the full RDO process. In this paper, we focus on the latter techniques.

Authors in [7,8] proposed to reduce the complexity of the HEVC encoder by skipping some depth levels of the quad-tree partitioning. The skipped depths were selected based on the correlation between the minimum depths of the collocated CTUs in the current and previous frames. Results in [8] showed an average time savings of 45% for a Bjøntegaard Delta Bit Rate (BD-BR) increase of 1.9%.

Works in [9–14] used CTU texture complexities to predict the quad-tree partitioning. Authors in [10] classified a Coding Unit (CU) as split, non-split, or undetermined. They used global and local edge complexities in four different directions (horizontal, vertical, 45°, and 135° diagonals) of CU and sub-CUs. This method provided a complexity reduction of 52% for a BD-BR penalty of 0.8%. Feng et al. [11] used information entropy of CUs and sub-CUs saliency maps to predict the CU sizes. The method reduced the complexity by 37.9% for a BD-BR cost of 0.62%.

For the time being, several Machine Learning (ML) based solutions have been proposed to reduce the complexity of the HEVC encoding. Authors in [15, 16] presented an intra CU size classifier based on data-mining with an offline classifier training. The classifier was a three-node decision tree that used mean and variance of CUs and sub-CUs as features. This algorithm reduced coding time by 52% at the expense of BD-BR increase of 2%. Duanmu et al. [17] presented a fast CU partitioning scheme using ML for screen content coding. They used many features such as CU luma variances, color Kurtosis of CU, and gradient Kurtosis of CU. Shen and Yu [18] proposed an early termination algorithm for CU splitting. It was based on weighted Support Vector Machine (SVM). The Rate-Distortion (RD)-cost losses due to the misclassification are used as features (weights) in SVM training. In [19], authors modeled the CU depth decision process in HEVC with a three-level hierarchical decision problem using SVM classifiers. Liu et al. [20] presented Convolution Neural Network (CNN) based CTU partitioning prediction scheme that infers CU and Prediction Unit (PU) split decision. The presented solution reduced the coding time by 61.1% at the expense of BD-BR increase of 2.67%.

In this paper, we propose a new method to predict HEVC quad-tree partitioning in order to reduce the complexity of the practical Kvazaar HEVC encoder. The proposed complexity reduction technique makes use of an ML algorithm to predict an adaptive quad-tree partitioning interval for a CTU before starting the RDO process. The existing complexity reduction techniques in the literature worked on the HEVC Test Model (HM) [21] software encoder and their relative performance figures do not necessarily scale to practical encoders due to the inherent compelxity of HM. Unlike them, our work focuses on complexity reduction under a practical framework.

The rest of this paper is organized as follows. Section 2 presents a brief overview of the Kvazaar HEVC intra encoder. Section 3 details our ML based algorithm for quad-tree partitioning prediction. Performance of the proposed complexity reduction technique is presented in Section 4. Finally, Section 5 concludes the paper.

2. KVAZAAR HEVC INTRA ENCODER

Kvazaar [22] is an academic, cross-platform software HEVC encoder. It is open-sourced under the LGPLv2.1 license. Unlike the reference encoder HM, Kvazaar takes advantage of multiple processor cores and Single Instruction Multiple Data (SIMD) instructions [23]. Kvazaar is also supported by FFmpeg and Libav projects where it can be used as an external library. The veryslow preset of Kvazaar intra encoder is described next at a high level as it is used in this work.

2.1. Coding Tree Unit Partitioning

The best CTU structure is the result of a recursive depth-first search in the quad-tree. Progressing top-down, CU split decisions are made by comparing the RD-costs of the CU at current depth versus the combined RD-cost of the 4 sub-CUs, where sub-CUs may be split even further. The RD-cost J is computed as $J = D + \lambda \cdot R$, where D is the distortion, R is the bit rate, and λ Lagrange multiplier [24]. Distortion D is computed by Sum of Squared Differences (SSD).

Two early termination mechanisms are implemented to speed up coding tree partitioning. The first one terminates the search when all transformed coefficients of the current CU are equal to zero. The second one prevents further search of the sub-CUs if their accumulated RD-cost (1 to 3 sub-CUs) is higher than that of their parent CU at the previous depth.

2.2. Intra Mode Decision

The intra search algorithm consists of two stages. First, a logarithmic search is performed to find the minimum distortion among angular intra modes. In a rough step, luma distortion is computed by Sum of Absolute Transformed Differences (SATD) between luma samples of the source image and prediction blocks. Luma mode bits are multiplied by square root of Lagrange multiplier λ and then added to the distortion for an estimated RD-cost. Most probable angular modes, planar mode, and DC mode are also considered.

In the RDO stage of the search, previously estimated modes are sorted according to their costs. Depending on the current block size, at most 2 or 3 best modes are forwarded to the RDO search. Most probable modes are again added to the list of modes if they are not present. Rate-distortion optimized quantization is performed for the selected modes which are then completely reconstructed. SSD between the reconstructed and source image samples is computed by adding up luma and both chroma channels. The number of used bits is calculated through CABAC, including transform tree and transformed coefficient bits. The mode with a minimum RD-cost [25] is selected as the best mode.

3. PROPOSED COMPLEXITY REDUCTION TECHNIQUE

The aim of the proposed technique is to replace the brute force scheme usually employed in HEVC encoders with a low-complexity algorithm that predicts a depth interval of CTU partitioning in which the HEVC encoder is constrained to apply the RDO process. In general, limiting the search to a certain interval reduces encoding complexity. The proposed technique is divided into two stages: 1) the ML-based one-shot prediction of quad-tree partitioning and 2) the interval prediction for quad-tree partitioning.

3.1. ML-Based One-shot Prediction of Quad-Tree Partitioning

The quad-tree prediction is called *one-shot* as the prediction is applied only once, before starting the RDO process of the CTU.

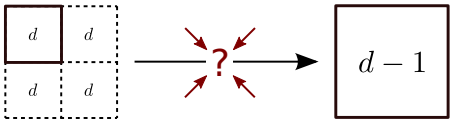

Fig. 1. Classification problem between CUs at depths d and $d-1$.

3.1.1. Quad-Tree Partitioning as a Classification Problem

Following a bottom-up approach (from CU size of 4×4[1] to 32×32), the main idea is to determine the best partitioning of a given CU between $2N\times2N$ pixels and $N\times N$ pixel sub-blocks at each depth. Fig. 1 illustrates the classification problem which predicts whether the CU at depth d has to be merged with CU at depth $d-1$.

At each depth d, the classification problem is solved by a ML approach across data-mining classifiers. The aforementioned state-of-the-art studies gather many characteristics used to predict the coding tree decomposition of a CTU. To predict the coding tree in one-shot, only characteristics independent from the encoding process with a limited overhead of computation are considered.

To avoid overfitting, i.e., overspecializing a model to a training set, the sequences are split in two data sets: the training set composed of one sequence per class and the experimental set composed of the other sequences. A training data pool is extracted from a fixed number of CTUs of each sequence of the training data set. For each depth d, 80 000 instances are randomly sampled from the previous defined data pool, composed of 40 000 instances of each prediction decision. The training setup of the learning algorithm and the choice of the features is detailed in [26]. The features have been deduced from an extensive study of two factors: the *information gain* provided by the Waikato Environment for Knowledge Analysis (WEKA) software and the overhead of computation under a practical encoder. The set of features is composed of the following 12 features:

- **CU var** [9, 13–17] : the variance of the CU luma samples at depth d (1 feature).
- **Lower-CU var** [9, 13, 15–17]: the variances of the 4 sub-CU luma samples at depth $d+1$ (4 features).
- **Upper-CU var** [9, 13–16]: the variances of the upper CU luma samples at depth $d-1$ (1 features).
- **Nhbr-CU var** [13, 17]: the variances of the neighboring CU luma samples at depth d in the Z-scan order (3 features).
- **Var of lower-CU mean** [15, 16]: the variance of the mean of the 4 sub-CU luma samples at depth $d+1$ (1 feature).
- **Var of lower-CU var** [15, 16]: the variance of the variance of the 4 sub-CU luma samples at depth $d+1$ (1 feature).
- **Quantization Parameter (QP)**: the Quantization Parameter (QP) of the frame (1 feature).

The training of the decision trees is performed with the C4.5 algorithm [27]. As the *information gain*, the C4.5 algorithm uses Kullback-Leibler Divergence (KLD) to select the best features for each decision. The C4.5 algorithm is iterating among all training instances and searches the threshold that achieves the best classification for each feature, i.e., with the highest *information gain*. Then, the features and their corresponding thresholds are used to divide the training instances into two subsets. To finish, the process is recursively iterated on the two different subsets of training instances.

Table 1 summarizes the trained tree sizes, number of leaves, and the Percentage of Correctly Classify Instances (PCCI) of the 4 decision trees, where PCCI (given by the 10-fold cross-validation) de-

[1] 8×8 CU split into 4×4 prediction units is considered as 4×4 CUs.

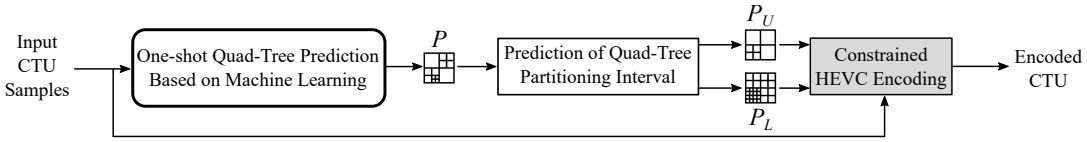

Fig. 2. High-level diagram of the proposed complexity reduction scheme.

Table 1. Dimensions and accuracy of decision trees (PCCI)

	Decision Trees			
Depth	$d = 4$	$d = 3$	$d = 2$	$d = 1$
Leaves	18	15	10	9
Size	15	13	13	15
PCCI	81.39%	80.52%	80.19%	81.26%

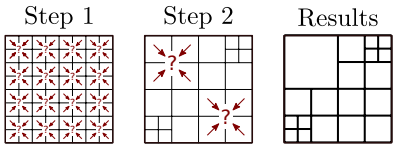

Fig. 3. CTU partitioning prediction, flowing a bottom-up approach.

scribes the accuracy of the decision trees. The results show that the accuracy of decision trees is high, i.e., over 80% of the decisions are valid.

3.1.2. Bottom-Up Approach for Quad-Tree Partitioning Prediction

Fig. 3 describes the proposed bottom-up algorithm that predicts the CTU partitioning using a ML approach. The algorithm takes as inputs the set of previously computed features (Section 3.1.1) and uses them to predict the CTU quad-tree partitioning.

First of all, the prediction (P) for CTU partitioning is initialized with the maximum depth value of 4. Then, the algorithm explores the CTU partitioning with a bottom-up approach: from $d = 4$ to $d = 1$. At each depth d, the algorithm browses the CTU prediction by taking the block size into account. Afterwards, the algorithm tests if the 4 neighboring blocks in the Z-scan order have the same depth. The algorithm does not try to merge neighboring blocks of different depths as illustrated in Fig. 3. If the former condition is true, the algorithm uses the prediction of decision trees (Section 3.1.1) to test whether the blocks can be merged or not and updates P accordingly.

After testing each depth, the one-shot quad-tree prediction is ready and the obtained P is delivered to the next stage (Section 3.2) for interval prediction.

3.2. Interval Prediction for Quad-Tree Partitioning

The goal of this stage is to relax the predicted CTU partitioning by generating a prediction interval around the input P (Section 3.1.2). The interval is specified between the upper prediction P_U and the lower prediction P_L. The algorithm is split into three steps:

1. The first step applies the algorithm called *Upper Expansion* on the input P. The algorithm merges the group of four neighboring blocks (in the Z-scan order) if they are at the same depth. The number of merged blocks depends on P and is not predictable. The merged blocks are indicated in red in Fig. 4.
2. The second step applies the "complementary" algorithm, called *Lower Expansion*, to generate the lower prediction P_L from P by splitting all blocks that have not been merged during the step 1. The split blocks are in blue in Fig. 4.
3. Finally, the third step applies a second pass of the Upper Expansion algorithm on the prediction yielded in the step 1 to generate the upper prediction P_U.

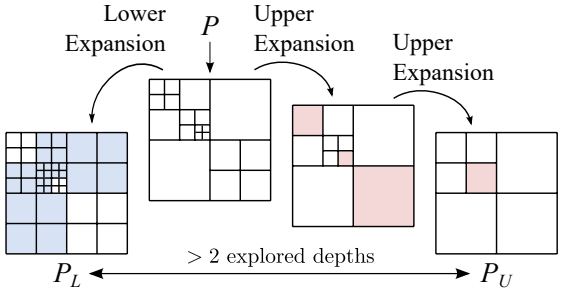

Fig. 4. Interval prediction algorithm for quad-tree partitioning.

When constrained between the P_L and P_U, the encoder tests 2 to 3 depth levels spread around the input P depending on the number of blocks merged by the third step.

3.3. Complexity Reduction Scheme

Fig. 2 presents a high-level diagram of our resulting ML complexity reduction scheme. Thanks to the offline training of the decision trees, all the frames are constrained and no learning frame is needed.

First, the features (Section 3.1.1) are computed for a CTU of interest. Secondly, the ML based algorithm (Section 3.1.2) uses the computed features to generate a coarse quad-tree prediction for the CTU. Thirdly, the interval prediction algorithm (Section 3.2) uses the coarse prediction to compute a prediction interval between the upper prediction P_U and the lower prediction P_L. Finally, the HEVC encoder is forced to limit RDO between the interval formed by P_U and P_L.

4. PERFORMANCE ANALYSIS

The impact of the proposed complexity reduction techniques on Kvazaar performance is measured with the 18 8-bit common test sequences encoded on an Intel Core i7-5960X Extreme (8 × 3.0 GHz) processor with 32 GB of RAM memory.

Table 2 shows the BD-BR and speedup results of the proposed complexity reduction scheme over the anchor version of Kvazaar [23], version 2.8 of x265 [28], and version 16.8 of HM [21]. The benchmarking is performed with command line parameters listed in Table 3. The veryslow presets of Kvazaar and x265 are close to each other in functionality, so they are selected for the comparison. HM is set to use default configuration of All-intra coding.

Table 2. Coding speed and efficiency of the optimized Kvazaar over the original Kvazaar, x265, and HM.

Format	Sequences	Proposed technique		Kvazaar vs. x265		Kvazaar vs. HM		
		Speedup 16 threads	BD-BR	Speedup 16 threads	BD-BR	Speedup 1 thread	Speedup 16 threads	BD-BR
Class A 2560×1600 (1600p)	PeopleOnStreet	1.32×	0.27%	1.84×	1.48%	8.41×	68.69×	2.01%
	Traffic*	1.36×	0.39%	1.89×	1.98%	8.66×	70.19×	2.32%
Class B 1920×1080 (1080p)	BasketballDrive	1.47×	0.84%	2.22×	1.62%	10.69×	86.86×	2.47%
	BQTerrace	1.33×	0.63%	1.95×	2.17%	8.09×	65.75×	2.14%
	Cactus*	1.40×	0.42%	2.09×	1.77%	8.83×	71.79×	2.21%
	Kimono	1.48×	0.82%	2.33×	0.93%	11.54×	93.13×	1.55%
	ParkScene	1.36×	0.45%	2.08×	1.19%	8.38×	68.28×	1.56%
Class C 832×480 (WVGA)	BasketballDrill	1.40×	1.02%	2.03×	2.75%	7.98×	63.37×	3.54%
	BQMall*	1.33×	0.42%	1.97×	1.11%	7.78×	63.59×	2.05%
	PartyScene	1.27×	0.10%	1.89×	1.08%	5.70×	47.01×	1.64%
	RaceHorses	1.38×	0.38%	2.10×	1.13%	7.48×	60.12×	1.71%
Class D 416×240 (WQVGA)	BasketballPass*	1.38×	0.28%	2.46×	-0.39%	7.51×	60.44×	1.38%
	BlowingBubbles	1.26×	0.10%	2.23×	0.55%	5.39×	43.90×	1.55%
	BQSquare	1.31×	0.13%	2.38×	0.45%	5.84×	47.20×	1.65%
	RaceHorses	1.35×	0.23%	2.39×	0.05%	6.67×	52.70×	1.45%
Class E 1280×720 (720p)	FourPeople	1.31×	0.83%	1.94×	1.62%	9.86×	79.67×	2.54%
	Johnny*	1.31×	0.95%	2.27×	1.20%	12.70×	99.55×	3.21%
	KristenAndSara	1.35×	0.60%	2.14×	1.19%	11.84×	92.09×	2.68%
	Average	**1.35×**	**0.49%**	**2.12×**	**1.21%**	**8.52×**	**68.57×**	**2.09%**

**test sequences used for the training phase*

Table 3. Encoding parameters of Kvazaar, HM, and x265

Encoder	Parameters
Kvazaar	--preset=veryslow --rd=2 --threads=16 --gop=0 --wpp --no-info -p=1 --no-rdoq-skip -n=(frames) -q=(qp)
HM	-c encoder_intra_main.cfg
x265	--tune=psnr --psnr --no-info --preset=veryslow --no-progress --hash=3 -q=(qp) -f=(frames) -I=1 --ipratio=1 --no-scenecut --pools=(threads) --fps=(fps) --loglevel=debug

4.1. Speedup over the original Kvazaar

The proposed complexity reduction technique speeds up the encoding process by 1.35× on average for an BD-BR increase of 0.49%. First of all, it is noticeable in Table 2 that sequences used to constitute the training data set (marked by *) do not achieve better results compared with other sequences, which show the non-overfitted behavior of the decision trees. Therefore, training sequences are also included in Table 2.

The results also show that the class D has less degradation in terms of BD-BR (+0.18% in average) than the other classes. This stems from the selected strategy for CTU partitioning prediction (Section 3.2), where neighboring blocks of different depths cannot be merged. This approach tends to result in finer-grained CTU partitioning which favors smaller resolutions.

4.2. Speedup over x265 and HM

Kvazaar and x265 were run with 16 threads using all available optimizations. Both Kvazaar and x265 support multi-threading and SIMD optimizations for 8-bit content. Kvazaar encodes each test sequence faster for similar RD performance in a majority of cases. The average speedup of Kvazaar over x265 is 2.12× with a 1.21% increase in BD-BR. The average complexity overhead of the proposed technique is around 2% in Kvazaar.

As HM does not implement multi-threading, Table 2 includes Kvazaar results with a single thread for the sake of more straightforward algorithm level comparison. Kvazaar is shown to be 8.52× as fast as HM, when using only one thread, and more than 68.5× as fast as HM with 16 threads. In spite of the large speedup, the BD-BR is deteriorated by only 2.09% against HM.

5. CONCLUSION

This paper presented a complexity reduction technique that makes use of an ML algorithm to predict adaptive quad-tree partitioning interval for a CTU before the RDO process. The proposed technique used a one-shot quad-tree partitioning prediction based on decision trees. It accelerates the practical Kvazaar HEVC encoder by 1.35× with 0.49% BD-BR overhead. The optimized Kvazaar is 2.12× as fast as the veryslow preset of x265 encoder with a BD-BR increase of 1.21%. These rate-distortion-complexity results show that Kvazaar is currently the front-runner among the existing open-source HEVC intra encoders when complexity aspect is taken into account.

6. ACKNOWLEDGEMENT

This work was supported in part by the European Celtic-Plus Project VIRTUOSE and the Academy of Finland (decision no. 301820). The authors would also like to thank all contributors of the Kvazaar open-source project [22].

7. REFERENCES

[1] "Cisco Visual Networking Index: Forecast and Trends, 2017–2022," p. 38, 2018.

[2] Vivienne Sze, Madhukar Budagavi, and Gary J. Sullivan, Eds., *High Efficiency Video Coding (HEVC)*, Integrated Circuits and Systems. Springer International Publishing, Cham, 2014.

[3] Thiow Keng Tan, Rajitha Weerakkody, Marta Mrak, Naeem Ramzan, Vittorio Baroncini, Jens-Rainer Ohm, and Gary J. Sullivan, "Video Quality Evaluation Methodology and Verification Testing of HEVC Compression Performance," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 76–90, Jan. 2016.

[4] Jarno Vanne, Marko Viitanen, Timo D. Hamalainen, and Antti 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] Guilherme Correa, Pedro Assuncao, Luciano Agostini, and Luis A. da Silva Cruz, "Performance and Computational Complexity Assessment of High-Efficiency Video Encoders," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1899–1909, Dec. 2012.

[6] Alexandre Mercat, Florian Arrestier, Wassim Hamidouche, Maxime Pelcat, and Daniel Menard, "Energy Reduction Opportunities in an HEVC Real-Time Encoder," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. 2017, pp. 1158–1162, IEEE.

[7] Liquan Shen, Zhaoyang Zhang, and Ping An, "Fast CU size decision and mode decision algorithm for HEVC intra coding," *IEEE Transactions on Consumer Electronics*, vol. 59, no. 1, pp. 207–213, 2013.

[8] Michele Belotti Cassa, Matteo Naccari, and Fernando Pereira, "Fast rate distortion optimization for the emerging HEVC standard," in *Picture Coding Symposium (PCS), 2012*. 2012, pp. 493–496, IEEE.

[9] Alexandre Mercat, Florian Arrestier, Maxime Pelcat, Wassim Hamidouche, and Daniel Menard, "Prediction of Quad-Tree Partitioning for Budgeted Energy HEVC Encoding," in *Signal Processing Systems (SiPS), 2017 IEEE Workshop on*. 2017, pp. 1–6, IEEE.

[10] Biao Min and Ray C. C. Cheung, "A Fast CU Size Decision Algorithm for the HEVC Intra Encoder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 5, pp. 892–896, May 2015.

[11] Lei Feng, Ming Dai, Chun-lei Zhao, and Jing-ying Xiong, "Fast prediction unit selection method for HEVC intra prediction based on salient regions," *Optoelectronics Letters*, vol. 12, no. 4, pp. 316–320, July 2016.

[12] Xuanjing Wang and Yonglin Xue, "Fast HEVC intra coding algorithm based on Otsu's method and gradient," in *Broadband Multimedia Systems and Broadcasting (BMSB), 2016 IEEE International Symposium on*. 2016, pp. 1–5, IEEE.

[13] Muhammad Usman Karim Khan, Muhammad Shafique, and Jörg Henkel, "An adaptive complexity reduction scheme with fast prediction unit decision for HEVC intra encoding," in *Image Processing (ICIP), 2013 20th IEEE International Conference on*. 2013, pp. 1578–1582, IEEE.

[14] Kuan-Kai Peng, Jui-Chiu Chiang, and Wen-Nung Lie, "Low Complexity Depth Intra Coding Combining Fast Intra Mode and Fast CU Size Decision in 3d-HEVC," 2016, pp. 1126–1130, IEEE.

[15] Damián Ruiz, Gerardo Fernández-Escribano, Velibor Adzic, Hari Kalva, José Luis Martínez, and Pedro Cuenca, "Fast CU partitioning algorithm for HEVC intra coding using data mining," *Multimedia Tools and Applications*, pp. 861–894, Nov. 2015.

[16] Damián Ruiz-Coll, Velibor Adzic, Gerardo Fernández-Escribano, Hari Kalva, José Luis Martínez, and Pedro Cuenca, "Fast partitioning algorithm for HEVC Intra frame coding using machine learning," in *Image Processing (ICIP), 2014 IEEE International Conference on*. 2014, pp. 4112–4116, IEEE.

[17] Fanyi Duanmu, Zhan Ma, and Yao Wang, "Fast CU partition decision using machine learning for screen content compression," in *Image Processing (ICIP), 2015 IEEE International Conference on*. 2015, pp. 4972–4976, IEEE.

[18] 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.

[19] Yun Zhang, Sam Kwong, Xu Wang, Hui Yuan, Zhaoqing Pan, and Long Xu, "Machine Learning-Based Coding Unit Depth Decisions for Flexible Complexity Allocation in High Efficiency Video Coding," *IEEE Transactions on Image Processing*, vol. 24, no. 7, pp. 2225–2238, July 2015.

[20] Zhenyu Liu, Xianyu Yu, Yuan Gao, Shaolin Chen, Xiangyang Ji, and Dongsheng Wang, "CU Partition Mode Decision for HEVC Hardwired Intra Encoder Using Convolution Neural Network," *IEEE Transactions on Image Processing*, vol. 25, no. 11, pp. 5088–5103, Nov. 2016.

[21] JCT-VC, "HEVC reference software," 2016, https://hevc.hhi.fraunhofer.de/.

[22] UltraVideoGroup, "Kvazaar HEVC Encoder," 2017, http://ultravideo.cs.tut.fi/#encoder.

[23] Ari Lemmetti, Eemeli Kallio, Marko Viitanen, Jarno Vanne, and Timo D. Hämäläinen, "Rate-Distortion-Complexity Optimized Coding Scheme for Kvazaar HEVC Intra Encoder," in *Data Compression Conference (DCC)*. 2018, p. 419, IEEE.

[24] Gary J. Sullivan and Thomas Wiegand, "Rate-Distortion Optimization for Video Compression," vol. IEEE Signal Process, no. 15, pp. 74–90, 1998.

[25] Marko Viitanen, Ari Koivula, Ari Lemmetti, Jarno Vanne, and Timo D. Hamalainen, "Kvazaar HEVC encoder for efficient intra coding," in *Circuits and Systems (ISCAS), 2015 IEEE International Symposium on*. 2015, pp. 1662–1665, IEEE.

[26] Alexandre Mercat, Florian Arrestier, Maxime Pelcat, Wassim Hamidouche, and Daniel Menard, "Machine Learning Based Choice of Characteristics for the One-Shot Determination of the HEVC Intra Coding Tree," in *2018 Picture Coding Symposium (PCS)*, San Francisco, CA, USA, June 2018, pp. 263–267, IEEE.

[27] John Ross Quinlan, *C4. 5: Programs for machine learning*, Elsevier, 2014.

[28] MulticoreWare, "x265 HEVC Encoder / H.265 Video Codec," 2017, http://x265.org/.