JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 39, 339-352 (2023)
DOI: 10.6688/JISE.202303_39(2).0006

Fast Intra Coding Partition Decisions for Feature Video Coding

JIUNN-TSAIR FANG[1,+], YU-LIANG TU[2] AND PAO-CHI CHANG[2]
[1]*Department of Electronic Engineering*
Ming Chuan University
Taoyuan City, 333 Taiwan
E-mail: fang@mail.mcu.edu.tw
[2]*Department of Communication Engineering*
National Central University
Taoyuan City, 320 Taiwan

Future video coding (FVC) can support 4K high-resolution videos to replace the previous coding standard, high efficiency video coding (HEVC). In particular, FVC adopts quadtree with binary tree (QTBT) to improve coding efficiency; however, encoding time increases heavily. Thus, we propose fast partition algorithms for FVC intra coding. Fast partition algorithms include spatial correlation method and deep learning method. The spatial correlation method uses gradient variances of pixels to determine early skip and early termination for QT depths 0 and 1, respectively. The deep learning method uses convolutional neural networks (CNNs) to predict the QTBT coding structure at QT depth 2 with its corresponding coding trees. Experimental results show that the proposed method can reduce encoding time by 12.49% on average but increases Bjontegaard delta bit rate (BDBR) by about 0.54%.

Keywords: future video coding, intra coding, quadtree with binary tree, coding unit, convolutional neural networks

1. INTRODUCTION

With rapid growth of Internet and multimedia technology, high-resolution videos in daily life have become indispensable. Recently, there have been many 4K high-resolution or even higher-resolution videos on the market. It can be expected that high-resolution video will become even more popular in the future, and the video compression standard, high efficiency video coding (HEVC), has gradually become inadequate. The Joint Video Exploration Team (JVET), which comprises International Telecommunication Union-Telecommunication Standardization Sector Video Coding Experts Group (ITU-T VCEG) and International Organization for Standardization/International Electrotechnical Commission Moving Picture Experts Group (ISO/IEC MPEG), has developed a new generation of video compression standard called future video coding (FVC) [1]. FVC applies quadtree plus binary tree (QTBT) coding structure, revising the quadtree coding structure of HEVC, to improve the coding efficiency. QTBT coding structure is more adapted to the texture characteristics of pictures sizes, but its encoding time increases heavily. Therefore, how to reduce the coding time of QTBT coding structure has become an important research issue.

FVC not only supports 4K resolution video, but also supports aerial photography, HDR/SDR, 360-degree video and so on [2-4]. The application of FVC in daily life is wider

Received October 1, 2021; revised January 13 & May 15, 2022; accepted May 22, 2022.
Communicated by Jen-Hui Chuang.

than HEVC can possibly support. The reference software of FVC, JEM, modified based on the HEVC reference software (HM) can improve the coding performance of HM, but its coding complexity increases heavily. Table 1 presents the comparative coding performance between the reference software JEM7.0 [5] of FVC and the reference software HM16.6 [6, 7] of HEVC. The coding time of JEM7.0 is 36 times longer than that of HM16.6 due to all intra coding configuration.

Table 1. Comparative performance of JEM7.0 [5], and HM16.6 [7].

Test configuration	BDBR			Time	
	Y	U	V	Enc.	Dec.
All intra	–20%	–28%	–27%	×36	×2
Random access	–29%	–35%	–34%	×10	×7
Low delay-B	–22%	–28%	–29%	×9	×7
Low delay-P	–26%	–31%	–32%	×7	×5

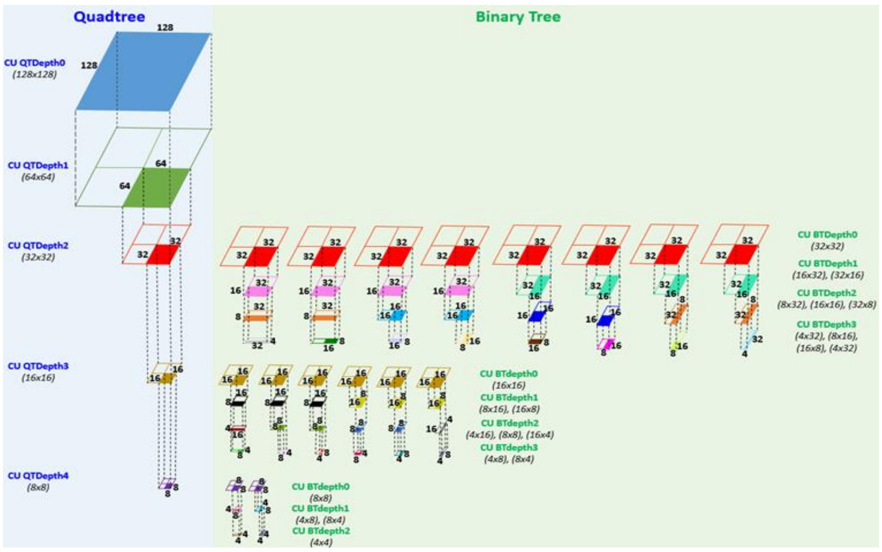


Fig. 1. QTBT coding structure [11].

FVC uses QTBT coding structure for prediction coding. According to QTBT, a coding tree unit (CTU) is composed of 128×128 pixels, which allows recursive splitting into four equal-sized sub-CUs, and each sub-CU can be further split into four smaller CUs (QT) or two smaller CUs (BT) until the smallest CU size is reached. A rate distortion optimization (RDO) procedure can determine the best CU size for coding prediction by choosing the minimum RD cost. Different from quadtree structure of HEVC, QTBT not only removes the hierarchical structure of CU, PU, and TU, but also adopts square or rectangular coding blocks according to the characteristic of the video picture. In other words, QTBT uses a variety of flexible coding shapes to improve coding performance [8]. Fig. 1 plots the QTBT coding structure [11]. For one CTU, the system performs QT partition decision at CU depths 0 and 1. At CU depths 2 to 4, this CU also performs BT partition decision, in addition to QT. BT coding structure can be separated by vertical split and horizontal split, which is the rectangular shape of CUs. Each BT has its own coding tree split. From the QTBT coding structure, the CU is no longer limited to a square shape which HEVC adopts. The square shape and rectangular shape for CUs allows more flexibility for prediction coding.

The parameters of QTBT coding structure are listed in Table 2 [9]. The quadtree node varies from 128×128 to 8×8, and the corresponding CU depth is from 0 to 4. If the leaf quadtree node does not exceed MaxBTSize, it can be further partitioned by the binary tree from BT Depths 0 to 3. If the width of binary tree node is equal to MinBTSize, it implies no further vertical partition. Similarly, if the height of binary tree node is equal to MinBTSize, it implies no further horizontal partition [9].

Table 2. Parameters for QTBT [9].

QTBT parameters	I slice
CTU Size	128×128
MinQTSize	8×8
MaxBTSize	32×32
MaxBTDepth	3
MinBTSize	4

Fast algorithms for intra coding are summarized as follows. Wang *et al.* [10] proposed to use the depth of neighbor CUs, including left, above, and the collocated CU in the previous frame, to estimate the depth for QTBT. A fast algorithm was proposed for QTBT intra coding based on spatial features wherein the gradient variances, differences of gradient variances, and edges were used to design the fast CU partition decision [11]. Spatial features for binary tree and quadtree were employed for early splitting or termination of CUs encoding process. Zouidi *et al.* [12] presented fast algorithms relying on statistical analysis. To accelerate the intra prediction mode selection, they listed prediction modes for each binary depth of the QTBT structure. From their analysis, higher correlation between QP parameter and the distribution of frequently used intra modes were used for each binary depth prediction. Chen *et al.* [13] used the coding mode of neighboring CUs to effectively eliminate unnecessary RDO operations. They analyzed the correlation of BT depth between the current block and neighboring ones. A threshold was set up to determine early CU split or skip. The Sobel operator was used to extract gradient features to decide the current encoding CU split [14]. From their analysis, homogeneous areas tended to select larger CUs while complicated texture tended to be divided into small sub-CUs. They also calculated the variance of variance of each sub-CU to decide the CU partition. Lin *et al.* [15] used CNNs to predict 67 intra modes. An input image was set up by 20×20 blocks to train CNNs. They finally selected the best five modes as candidates for RDO mode decision. Jin *et al.* [16] broke the CU depth coding structure and proposed CNNs to classify the 32×32 and 16×16 CU depths together. Finally, they classified 5 categories from these two QT depths and their corresponding BT coding structures. CNNs were applied as classifier to predict the depth of the CU [17]. They first created a database [18] to store different CU depths, and then used the subsampling to create a hierarchical CU partition map, which could handle blocks with different CU depths and sizes. Finally, a hierarchical CNN learning was proposed to predict the partition map for early termination. Tissier *et al.* [19] used the ResNets [20] structure of CNNs for classification. The input data is 65×65, which is the original CU sizes plus the left and above line pixels. After convolution, pooling, and fully connecting, the final output could determine whether or not the CU split with probability.

In this work, we propose fast partition algorithms to reduce the encoding time of intra coding for FVC. Fast partition prediction algorithms include deep learning method and spatial correlation method. The deep learning method uses CNNs to predict CU partition at QT depth 2 with its corresponding coding trees. The spatial correlation method uses gradient variances of pixels to determine early skip and early termination for QT depths 0 and 1. This paper is organized as follows. Section 2 describes the proposed fast algorithm. Experimental results are described in Section 3. Finally, Section 4 presents conclusions.

2. FAST ALGORITHMS FOR QTBT INTRA CODING

The proposed fast algorithms are designed to reduce the coding time of FVC intra coding while preserving the high quality of coding performance. Fast algorithms in this research include deep learning and spatial correlation method. The following two sub-sections describe the proposed method in detail.

2.1 Deep Learning for QT Depth 2 Split Prediction

CNNs are deep learning networks which have been proved to perform well in image classification [21, 22]. In Fig. 2, we plot training and testing procedures of CNNs. CNNs build network model and parameters through a training process. Four different resolutions from CPIH dataset [23] are the input sequences for training. They are 4928×3264, 2880×1920, 1536×1024, and 768×512. The ratio of training data and validating data is about 8:2. These sequences are encoded by JEM to determine the best CU split of each frame. Then, each frame is partitioned into 32×32 CUs, and these 32×32 CUs, whose final CUs split has been determined, can be used as labeling by supervised learning to train network model and parameters. The training procedure is plotted in the left part of Fig. 2.

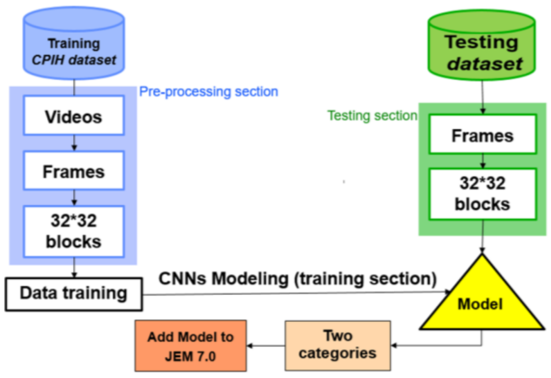


Fig. 2. Training and testing procedures of CNNs.

Some CUs may not be suitable to become labels because these CUs may be hard to be distinguished. For a given 32×32 CU, if its RD cost difference between the optimal and sub-optimal partitions is too small, this image is not distinguished, and is not suitable to become a label for training a CNNs model. The RD cost difference between the optimal and sub-optimal partitions is defined by Eq. (1) [16],

$$\Delta RDCost = \left|\frac{RDCost_{best} - RDCost_{sub}}{RDCost_{best} + RDCost_{sub}}\right| \tag{1}$$

where notation |·| is an absolute value, and $RDCost_{best}$ and $RDCost_{sub}$ are the best and the second best RD values, respectively. From our experiment result, RD cost difference is set at 0.02.

In this work, we use CNNs to classify CUs at QT depth 2. According to QTBT coding structure, plotted in Fig. 1, QT depth2 is a 32×32 CU and can be split into four sub-QTs or BTs coding trees. In [16], the authors used class depth to categorize final partitioning at QT depth2. The class depth is defined according to the maximum QTBT partition depth within a 32×32 CU, formulated by Eqs. (2), and (3),

$$Class\_depth = \text{argmax}(QTBTDepth_{CU_i}) \tag{2}$$

$$QTBTDepth_{CU_i} = 2 \times QTDepth_{CU_i} + BTDepth_{CU_i} \tag{3}$$

where $QTBTDepth_{CU_i}$ represents the final QTBT partitioning depth [16].

Fig. 1 shows all sub-CU coding trees of QTBT coding structure. According to Eq. (3), the sub-CU coding trees from CU depth 2 can be classified into 7 class depths, from 4 to 10. Following extensive simulations, we finally decided to merge these 7 classes into two categories so that CNNs can have better classification performance. Label 0 is for class depths 4-6, and label 1 is for class depths 7-10. Table 3 lists the classification of CUs in Labels 0 and 1. In general, Label 0 can be treated as CUs with smooth content and label 1 can be treated as CU with complicated content. Table 4 lists the number of training images in each label.

In the testing process, as shown in the right part of Fig. 2, JEM encodes each frame as original procedure. While encoding each 32×32 CU, JEM exports CU image data into CNNs. CNNs uses the already built network model to classify the image data into two categories, and passes the classification results back to JEM. After two labels of classification, JEM continues the remaining encoding process within each category to find the best CU for prediction coding.

Table 3. The classification of CUs in Labels 0 and 1.

Label	CU depths	
Labe 0	QT2	BT0
		BT1
		BT2
	QT3	BT0
Label 1	QT2	BT3
	QT3	BT1
		BT2
		BT3
	QT4	BT0
		BT1
		BT2

Table 4. The number of training CUs (32×32) in each label.

	Class_depth	
QP	4~6 (Label 0))	7~10 (Label 1)
22	68,339	74,765
27	140,103	80,788
32	173,051	71,357
37	196,308	55,204
Total	577,801 (63.7%)	328,779 (36.2%)

In Fig. 3, we plot the architecture of proposed CNNs, in which the dense convolutional networks (DenseNets) [24] are adopted. The DenseNets include convolution layers, batch normalization, pooling, and fully connected layers. Input data are 32×32 CUs; there are three convolution layers, with 3×3 kernels, and the stride equals to 1. Batch normalization can make neural networks faster and more stable through normalization of each layer inputs [25]. After 3 dense layers, maximum pooling is used to remove noise points. Finally, three layers of full connection are for classification, and the result of final classification is sent to the output layer.

In CNNs, the parametric rectified linear unit (PReLU) is used for the activation function, which can preserve a negative value after convolution operation. Since there are only two categories, the binary cross entropy (BCE) method is used for classification. To avoid overfitting problem, regularization operation is added in the CNNs model. After several experimental tests, we used both L_1 and L_2 regularization functions [26]. L_1 is robust to error. However, its derivative is not continuous. L_2 is less robust to error, but is easier to get the best solution. L_1 and L_2 have different advantages and disadvantages. Applying L_1 and L_2 can solve the over-fitting problem and make the network model more flexible. Parameters for setting up CNNs models are listed in Table 5. For L_1, λ_1 sets at 0.03. For L_2, λ_2 sets at 0.07, 0.05, 0.03, or 0.01 for QP equals to 22, 27, 32, or 37, respectively.

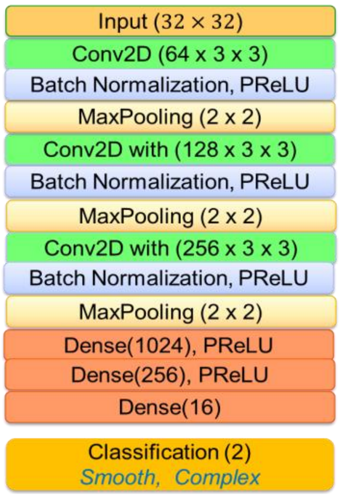


Fig. 3. The CNNs architecture.

Table 5. Parameters for CNNs.

Learning rate	0.001
Kernel size	(3, 3)
Max pooling	(2, 2)
Epoch	100
Batch size	256
Number input	(32, 32)
Classification	2
Parameter of PReLU	0.3

2.2 Spatial Correlation for Split Prediction at QT Depths 0 and 1

Spatial correlation has been used to effectively predict texture smoothness of CU blocks [11-13]. To further reduce the coding time of QTBT intra coding, we use spatial correlation for split decision at QT depths 0 and 1, separately. However, fast algorithms in these two CU depths should avoid error decisions because error decisions could greatly affect the video quality.

For QT depths 0 and 1, Sobel filters are applied. Horizontal and vertical gradients, G_{x_i} and G_{y_i}, of a pixel are defined by Eqs. (4) and (5), separately,

$$G_{x_i} = P_{i+1,j-1} + 2 \times P_{i+1,j} + P_{i+1,j+1} - P_{i-1,j-1} - 2 \times P_{i-1,j} - P_{i-1,j+1}, \quad (4)$$

$$G_{y_i} = P_{i-1,j-1} + 2 \times P_{i,j-1} + P_{i+1,j-1} - P_{i-1,j+1} - 2 \times P_{i,j+1} - P_{i+1,j+1}. \quad (5)$$

where (i, j) is the coordinate of pixel $P_{i,j}$. The gradient of pixel $P_{i,j}$ is defined by $GRAD(P_{i,j}) = |G_{x_i}| + |G_{y_i}|$. Gradient on a CU is defined by the summation of all gradients of pixels within this CU. The gradient of a CU in QT depth 0 can be calculated by Eq. (6),

$$\sum CU_{GRAD} = \sum_{i=0}^{127} \sum_{j=0}^{127} GRAD(P_{i,j}) \cdot \quad (6)$$

For QT depth 0, the gradient of a CU can be used to predict whether or not this CU needs to split into sub-CUs. If its gradient is larger than a threshold, this CU will be split into its four sub-CUs directly without processing RDO. Otherwise, it will be determined by RDO procedure. With extensive experiments, the threshold is set at 200 [11].

Difference of gradient variance can be used for QT depth 1 split determination. QT depth 1 has four sub-CUs. The gradient variance of each sub-CU is calculated, individually, and expressed by Eq. (9),

$$Variance_{GRAD,k} = \tfrac{1}{N \times N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} (GRAD_{x,y} - \mu_{4,k}), k = 0,1,2,3 \quad (7)$$

where N equals to 32, and the gradient mean is defined by Eq. (10),

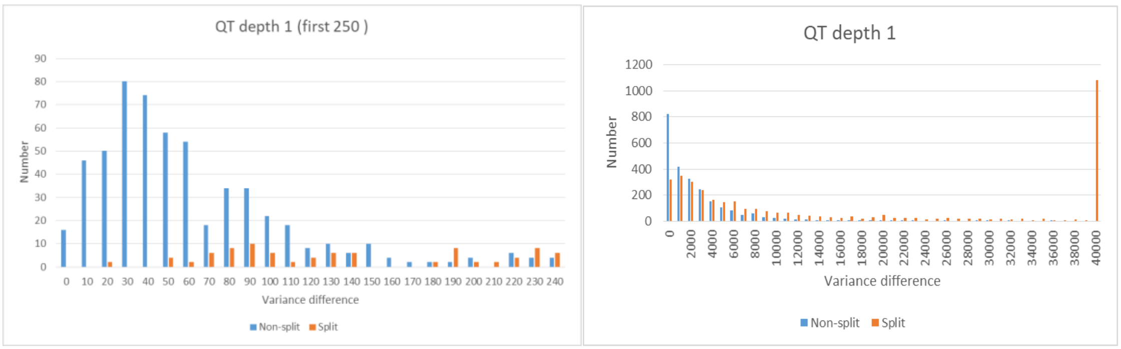
$$\mu_4 = \tfrac{1}{N \times N} \sum_{q=0}^{N-1} \sum_{r=0}^{N-1} GRAD_{q,r} \,. \quad (8)$$

The difference of gradient variance can be calculated by Eq. (11),

$$Diff_{Variance}^{GRAD} = \frac{1}{6}\sum_{k \neq r} | Variance_{GRAD,k} - Variance_{GRAD,r} | \tag{9}$$

where notation |·| is an absolute value. There are 6 differences between two sub-CUs. If the difference of gradient variance is large, it means the variance difference between four sub-CUs is large. This CU has a high chance to split into four sub-CUs (early split). In contrast, if the difference of gradient variance is small, it means the variance difference between four sub-CUs is small. This CU has a high chance to stay in CU depth 1 for prediction coding (early termination). If the difference of gradient variance is not large or small, then its CU split is determined by RDO procedure.

To determine thresholds for setting difference of gradient variance of 32×32 CUs at QP depth 1, three test sequences, ClassB_Kimono, ClassC_RaceHorsesC, ClassE_Kristen AndSara were used, and four QPs (22, 27, 32, and 37) were tested, separately. More than 2,400 CUs were non-split, and more than 3,700 CUs were split. Fig. 4 shows distributions of difference of gradient variances for split and non-split CUs at QP depth 1 under QP 27. To have better representation, we separate the result into two figures. Fig. 4 (a) shows the distributions of the first 250 bins whose difference of gradient variances are small. If the difference of gradient variance is less than 40, most CUs are determined to be non-split, plotted by blue lines. The distributions of all differences of gradient variances are plotted in Fig. 4 (b). If the difference of gradient variances are greater than 30, 000, most CUs are determined to be split into deeper CU depths. The last orange line in Fig. 4 (b) means that 376 CUs whose difference of gradient variances are greater than 40,000 are split into deeper depths. After experiments, the thresholds for early skip and early termination are set at about 3%, which are 25 and 330,000, respectively.



(a) Less than 250 (bin interval 10). (b) All (bin interval 1000).

Fig. 4. Distributions of difference of gradient variances for split and non-split CUs at QP depth 1 at QP 27.

Proposed fast algorithms are summarized as follows. A CTU with 128×128 pixels is first calculated by the summation of gradient. If the summation of gradient is greater than the threshold, then this CTU is directly split into four 64×64 of QT depth 1. Otherwise, RDO process determines if it is split or not. For a CU in QT depth 1, the difference of gradient variance is calculated. If the difference of gradient variance is greater the threshold of early split, then this CU is split into four sub-CUs of QT depth 2 directly. The difference of gradient variance is then compared with the threshold of early termination. If greater than the threshold, then this CU is not split. Otherwise, whether this CU if split or not is determined by RDO procedure. For a CU in QT depths 2 and 3, CNNs are used to classify CUs to be Labels 0 or Label 1. The RDO continues the decision to search the best CU within each Label. Fig. 5 plots the flow chart of proposed fast algorithms.

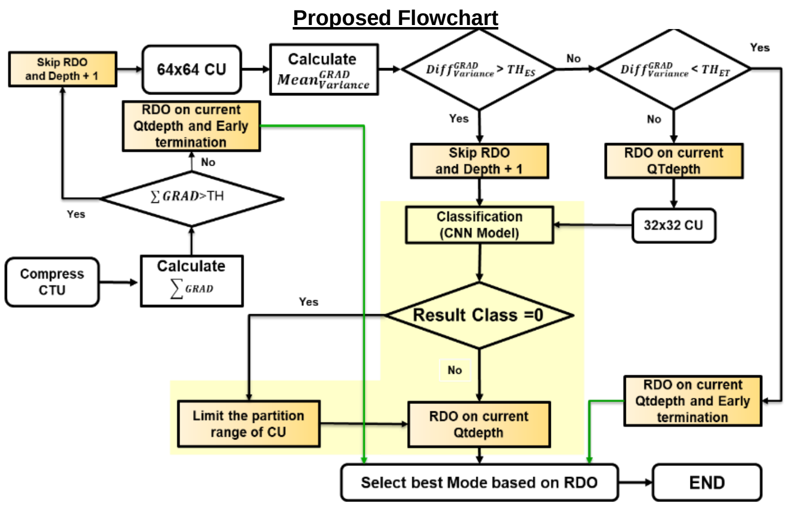

Proposed Flowchart
Skip RDO and Depth + 1
64x64 CU
Calculate $Mean^{GRAD}_{Variance}$
$Diff^{GRAD}_{Variance} > TH_{ES}$
No
$Diff^{GRAD}_{Variance} < TH_{ET}$
Yes
RDO on current Qtdepth and Early termination
Yes
No
Skip RDO and Depth + 1
RDO on current QTdepth
No
$\sum GRAD$>TH
Yes
Classification (CNN Model)
32x32 CU
Compress CTU
Calculate $\sum GRAD$
Yes
Result Class =0
No
RDO on current Qtdepth and Early termination
Limit the partition range of CU
RDO on current Qtdepth
Select best Mode based on RDO
END

Fig. 5. Flow chart of proposed fast algorithms.

3. EXPERIMENTAL RESULTS

Experiments are designed to show the performance of the proposed system. The confusion matrices of CNNs under different four QPs are shown in Fig. 6. The diagonal of each matrix shows the correction rates for label classification. For example, in the first matrix of QP 27, there is 84% chance of correctness if CNNs determine a CU to be Label 0. In contrast, there is 16% of chance CNNs may make a mistake for Label 0 under QP 27. The average of CNN prediction accuracy rates for CNNs under four different QP is about 75%.

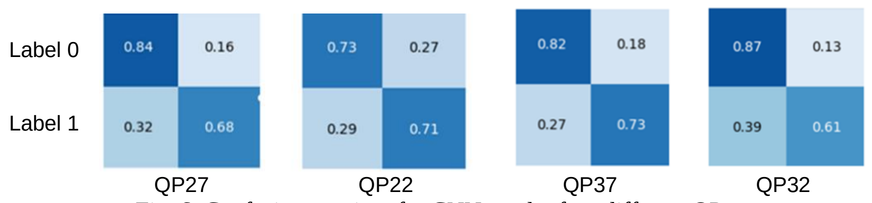


Fig. 6. Confusion matrices for CNNs under four different QPs.

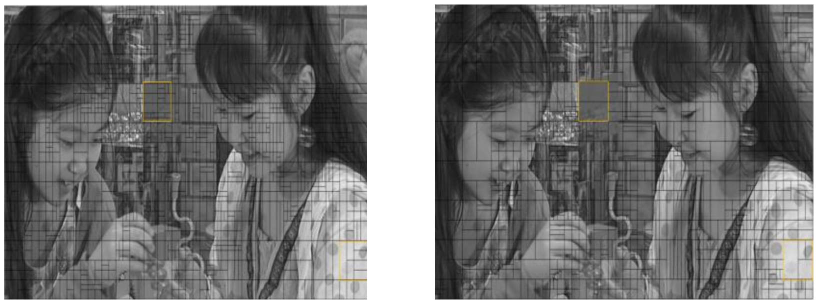
Experiment environment is listed in Table 6. The reference software of FVC is JEM version 7.0, and the test sequences refer to common test conditions (CTC) of FVC. All intra configuration is adopted for FVC, and four QPs are used. They are 22, 27, 32, and 37.

Table 6. Experiment environment.

Hardware		
CPU	Intel(R) Core(TM) i7-6800 @3.40 GHz	
RAM	32.0 G bytes	
GPU	NVIDIA GeForce GTX 2070	
Software		
Reference software	JEM 7.0	
Video Coding tool	Visual Studio 2015, Tensorflow [27], Keras [28]	
Test sequences	ClassB_Kimono	ClassD_BQSquare
	ClassB_Cactus	ClassD_BlowingBubbles
	ClassC_RaceHorsesC	ClassE_Johnny
	ClassC_PartyScene	ClassE_KristenAndSara
Configuration file	Intra_main_10	
QP	22, 27, 32, 37	

In Fig. 7, we show the final CU splits of ClassD_BlowingBubbles for the proposed method and JEM 7.0 method under QP 37. There are some differences of CU splits between these two methods. In particular, we use yellow color to mark two CU splits for comparisons, where they are located in the middle and the bottom right of the frame. The proposed method uses fewer CUs splits for these regions, which may be result from CNNs classification ability.

The bitrate and PSNR for JEM 7.0 and the proposed method to encode Class D BlowingBubbles under QP 37 are 202.1 kbps and 32.28 dB, and 200 kbps and 32.12 dB, respectively. The encoding time for JEM 7.0 and the proposed method is 892 seconds and 845 seconds, respectively. The proposed method sacrifices some video quality but can save encoding time.



(a) JEM 7.0 (b) Proposed method.

Fig. 7. Final CU splits of Class D BlowingBubbles between JEM 7.0 and the proposed method.

Performance comparisons with state-of-the-art work are listed in Table 7. The Bjøntegaard delta bit rate (BDBR) is calculated based on [29]. Time saving is calculation by Eq. (10),

$$\Delta T(\%) = \frac{1}{4}\sum_{i=1}^{QP_i} \frac{Encoding\ time_{JEM\,7.0}^{QP_i} - Encoding\ time_{proposed}^{QP_i}}{Encoding\ time_{JEM\,7.0}^{QP_i}} \times 100 \tag{10}$$

where *Encoding time*$_{JEM7.0}$ represents the encoding time for JEM 7.0 and *Encoding time* $_{Proposed}$ represents the encoding time by the proposed method. Four different QPs are tested, including 22, 27, 32, and 37.

In Table 7, the proposed method can reduce encoding time by 12.49% on average but increases BDBR by about 0.54%. In [11], the encoding time is reduced by 12.19% on average but the BDBR increases by about 0.34%. Both proposed methods can maintain the video quality and efficiently reduce the encoding time. In [11], fast algorithms are applied in each QT or BT partition. By contrast, the proposed method classifies the QT and BT partitions only into two labels. For future work, more labels or more classifications can be applied to reduce more encoding time.

Table 7. Comparative performance of proposed method and other method.

Class	Sequence	Porposed method		[11]	
		BDBR (%)	ΔT	BDBR (%)	ΔT
B	Kimono	0.16	1.19	0.53	5.16
(1920×1080)	Cactus	0.61	8.25	0.34	16.01
C	RaceHorses	0.48	11.63	0.17	12.90
(832×480)	PartyScene	0.59	26.14	0.21	10.11
D	BQSquare	0.39	21.20	0.33	12.71
(416×240)	BlowingBubbles	0.60	16.80	0.19	13.16
E	Johnny	0.75	7.37	0.64	13.25
(1280×720)	KristenAndSara	0.71	7.33	0.29	14.24
Average		0.54	12.49	0.34	12.19

4. CONCLUSION

In this work, we propose fast algorithms to determine the QTBT coding structure from FVC intra coding. Two methods are proposed. The first method is to apply CNNs for classification of QT depth 2 with its corresponding coding trees. The second method is using spatial correlation to determine the QT depths 0 and 1. By setting the thresholds for early skip and early termination, some RDO procedures can thus be skipped. This proposed method can yield coding time savings with low BDBR increases as compared to JEM-7.0. By combining both spatial correlation and deep learning, the proposed method can achieve an effective method to save coding time in FVC video coding.

REFERENCES

1. J. Chen, E. Alshina, G.-J. Sullivan, J.-R. Ohm, and J. Boyce, “Algorithm description of joint exploration test model 1,” *JVET-A1001*, 1st Meeting, Geneva, CH, 2015.
2. F. Bossen, J. Boyce, X. Li, V. Seregin, and K. Sühring, “JVET common test conditions and software reference configurations for SDR video,” *JVET-K1010*, 11th JVET Meeting, Ljubljana, SI, 2018.

3. A. Segall, E. François, and D. Rusanovskyy, "JVET common test conditions and evaluation procedures for HDR/WCG video," *JVET-K1011*, 11th JVET Meeting, Ljubljana, SI, 2018.
4. P. Hanhart, J. Boyce, and K. Choi, "JVET common test conditions and evaluation procedures for 360° video," *JVET-K1012*, 11th JVET meeting: Ljubljana, SI, 2018
5. M. Karczewicz and E. Alshina, "JVET AHG report: Tool evaluation (AHG1)," *JVET-H0001*, 8th Meeting, China, 2017.
6. G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 22, 2012, pp. 1649-1668.
7. HEVC reference software, https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.6/.
8. Z. Wang, S. Wang, J. Zhang, and S. Ma, "Probability decision based block partitioning for future video coding" *IEEE Transactions on Image Processing*, Vol. 27, 2018, pp. 1475-1486.
9. J. Chen, E. Alshina, G. J. Sullivan, J. R. Ohm, and J. Boyce, "Algorithm description of joint exploration test model 7 (JEM7)," *JVET-G1001*, 2017.
10. Z. Wang, S. Wang, J. Zhang, and S. Ma, "Effective quadtree plus binary tree block partition decision for future video coding," in *Proceedings of Data Compression Conference*, 2017, pp. 23-32.
11. T. L. Lin, H. Y. Jiang, J. Y. Huang, and P. C. Chang, "Fast intra coding unit partition decision in H.266/FVC based on spatial features," *Journal of Real-Time Image Processing*, Vol. 27, 2018, pp. 493-510.
12. N. Zouidi, F. Belghith, A. Kessentini, and N. Masmoudi, "Fast intra prediction decision algorithm for the QTBT structure," in *Proceedings of IEEE International Conference on Design & Test of Integrated Micro & Nano-Systems*, 2019, pp. 1-6.
13. J. Chen, Y. Chiu, C. Lee, and Y. Tsai, "Utilize neighboring LCU depth information to speedup FVC/H.266 intra coding," in *Proceedings of International Conference on System Science and Engineering*, 2019, pp. 308-312.
14. Y. Fan, J. Chen, H. Sun, J. Katto, and M. Jing, "A fast QTMT partition decision strategy for VVC intra prediction," *IEEE Access*, Vol. 8, 2020, pp. 107900-107911.
15. T. L. Lin, K. W. Liang, J. Y. Huang, Y. L. Tu, and P. C. Chang, "Intra mode prediction for H.266/FVC video coding based on convolutional neural network," *Journal of Visual Communication and Image Representation*, Vol. 71, 2020, pp. 1-5.
16. Z. Jin, P. An, C. Yang, and L. Shen, "Fast QTBT partition algorithm for intra frame coding through convolutional neural network," *IEEE Access*, Vol. 6, 2018, pp. 54660-54673.
17. M. Xu, T. Li, Z. Wang, X. Deng, and R. Yang, "Reducing complexity of HEVC: a deep learning approach," *IEEE Transactions on Image Processing*, Vol. 27, 2018, pp. 5044-5059.
18. D.-T. Dang-Nguyen, C. Pasquini, V. Conotter, and G. Boato, "RAISE: A raw images dataset for digital image forensics," in *Proceedings of the 6th ACM Multimedia Systems Conference*, 2015, pp. 219-224.
19. A. Tissier, W. Hamidouche, J. Vanne, F. Dalpin, and D. Menard, "CNN oriented complexity reduction of VVC intra encoder," in *Proceedings of IEEE International Conference on Image Processing*, 2020, pp. 3139-3143.

20. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* 2016, pp. 770-778.
21. K. Alex, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, 2012, pp. 1097-1105.
22. S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: a convolutional neural-network approach," in *Proceedings of IEEE Transactions on Neural Networks*, Vol. 8, 1997, pp. 98-113.
23. T. Li, M. Xu, and X. Deng, "A deep convolutional neural network approach for complexity reduction on intra-mode HEVC," in *Proceedings of IEEE International Conference on Multimedia and Expo*, 2017, pp. 1255-1260.
24. Z. L. G. Huang, L. V. D. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2261-2269.
25. I. Sergey and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *Proceedings of International Conference on Machine Learning*, 2015, pp. 448-456.
26. C. M. Bishop and N. M. Nasrabadi, *Pattern Recognition and Machine Learning*, Springer, NY, 2006.
27. TensorFlow: an open source Python package for machine intelligence, https://www.tensorflow.org, 2016.
28. keras: https://keras.io/.
29. G. Bjontegaard, "Calculation of average PSNR difference between RD-curves," ITU-T Q.6/SG16 VCEG, 13th Meeting, Doc. VCEG-M33, 2001.

Jiunn-Tsair Fang (方俊才) received the BS degree in Physics from National Taiwan University, and Ph.D. degree in Electrical Engineering from National Chung-Cheng University, Taiwan. Currently, he is an Associate Professor in the Department of Electronic Engineering in Ming Chuan University, Taiwan. His researching interest includes video coding (HEVC and VVC), audio processing, and deep learning techniques on multimedia applications.

Yu-Liang Tu (涂育良) received the MS degree in Communication Engineering from National Central University, Taiwan, in 2021. His research interest is video coding and deep learning. Now he works on Egis Technology Inc. in Taiwan.

Pao-Chi Chang (張寶基) received the BS and MS degrees from National Chiao Tung University, Taiwan, and the Ph.D. degree from Stanford University, California, all in Electrical Engineering. From 1986 to 1993, he was a research staff member of the Department of Communications at IBM T. J. Watson Research Center, Hawthorne, New York. At Watson, his work centered on high speed switching systems, efficient network design algorithms, and multimedia conferencing. In 1993, he joined the faculty of National Central University, Taiwan, where he is presently a Professor in the Department of Communication Engineering. In 1994, Dr. Chang established and has headed the Video-Audio Processing Laboratory (VAPLab) in the Electrical Engineering Department and Communication Department of National Central University since. Dr. Chang has been the principle investigator for many joint projects with Ministry of Science Council (MoST), Institute of Information Industry (III), Chung Hwa Telecommunication Laboratories (TL), and many other companies. His research interests include speech/audio coding, video/image compression, and deep learning techniques on multimedia applications.