Fast mode decision for CU coding based on CNN for the VVC standard

Bouthaina Abdallah,[a,*] **Fatma Belghith**,[a] **Mohamed Ali Ben Ayed**,[b] **and Nouri Masmoudi**[a]

[a]University of Sfax, National Engineering School of Sfax, Electronics and Information Technology Laboratory, Sfax, Tunisia

[b]University of Sfax, National School of Electronics and Communication (ENET'COM), New Technologies and Telecom Systems Laboratory (NTS'COM), Sfax, Tunisia

Abstract. Versatile Video Coding (VVC) is the latest generation of the video coding standard. In VVC, the advanced quadtree with a nested multitype tree (QTMT) partition structure provides more flexible coding unit (CU) partition sizes compared with the quadtree (QT) decision tree structure applied in the previous High Efficiency Video Coding (HEVC) standard. This flexibility, achieved by the new QTMT partitioning improvement, considerably improves the coding performance while increasing the coding computational complexity caused mainly by the rate distortion optimization processing. To overcome the complexity issue, a fast deep intra QTMT decision tree approach based on a convolution neural network (CNN) is adopted to determine the QTMT depth decision of each 128×128 Coding Tree Unit (CTU). The proposed algorithm predicts both the BT depths at 32×32 CUs and the QT depths at 64×64 using trained CNNs designed for each structure instead of processing the RDcost. Experimental results prove that the suggested deep QTMT approach achieves an important complexity reduction of up to 55.51% compared with the original reference software VTM3.0, with an average of about 35% encoding time reduction accompanied by an insignificant loss in encoding performance. [DOI: 10.1117/1.JEI.31.5.053004]

Keywords: Versatile Video Coding; quadtree with nested multitype tree; computational complexity; convolution neural network; complexity reduction.

Paper 220045G received Jan. 15, 2022; accepted for publication Aug. 17, 2022; published online Sep. 5, 2022.

1 Introduction

With the huge increase of video traffic on the internet and the emergence of new video content, such as virtual reality, high frame rate, and 360 video with resolutions of up to 4K, 8K, and even 16K, several advanced video compression standards have been developed, mainly by the VCEG and MPEG groups. This collaboration aims to identify the future generation of video coding. Therefore, the joint video experts team, established by the VCEG and MPEG groups, standardized a new video codec called Versatile Video Coding (VVC)[1] in July of 2020. The VVC standard is used to improve the coding efficiency of the previous codec High-Efficiency Video Coding (HEVC)[2] by providing a significant gain of about 50% in bitrate with the same video quality. The latest VVC standard enhances the RD performance at the cost of an increase in computational complexity caused by the improvement tools added to the VTM reference software. The new advanced quadtree with nested multitype tree (QTMT) partition structure is one of the most complicated improvement techniques compared with the quadtree plus binary tree (QTBT) decision tree used in the JEM reference software and the quadtree (QT) structure employed in HEVC.[3] In addition to the different square blocks supported by the QT scheme (128×128, 64×64, 32×32, 16×16, and 8×8), the VVC encoder provides further rectangular blocks in the horizontal and vertical splitting directions for the BT and TT structures (32×16, 16×32, 16×8, 8×16, 32×8, 8×32, 32×4, 4×32, 8×4, and 4×8) to improve the coding

*Address all correspondence to Bouthaina Abdallah, Boutheina.abdallah@enis.tn

units (CUs) partitioning shapes' flexibility. Thus, a fast QTMT decision algorithm has been developed to reduce the computational complexity at all intra (AI) configuration in the face of the large CU partition sizes. Several works on video coding have exploited the popularity of the deep learning (DL) field to achieve a trade-off between complexity reduction and compression efficiency. In the literature, the DL-based approach results outperform those of the machine learning-based methods and the statistical-based approaches in video processing. Recently, the DL algorithms based on the convolution neural network (CNN) have shown significant coding results in terms of complexity reduction while maintaining almost the RD performance. This work focuses on developing a fast CU partition method while reducing the QTMT structure complexity. Therefore, a deep QTMT partitioning algorithm based on the CNN is designed in this study to minimize the VVC complexity at AI configuration.

In this paper, five parts are introduced as follows. Section 2 studies the related works on QTBT and QTMT complexity reduction for the VVC standard. Section 3 reviews the QTMT partition decision process. In Sec. 4, the proposed intra QTMT algorithm applying several CNNs is described, with different network architectures being designed for the QT and BT structures in the horizontal and vertical directions. The experimental results of the current approach are discussed in Sec. 5. Finally, Sec. 6 concludes the present work.

2 Related Works

Several works have been developed in recent years to overcome the VVC complexity and HEVC standards by providing a trade-off between bitrate, video quality, and encoding time using ML techniques and deep methods.

For HEVC, an effective block partition decision algorithm in intra-coding[4] was implemented to accelerate the encoding process by developing an early determination of the CU mode decision and a bypass strategy for a large block size partitioning. This approach reached a maximum of 67% complexity reduction with almost the same bitrate and quality compared with the original HM 10.0 software. Another work based on the determination of the CU depth range and on early termination algorithms was presented by Shen et al.[5] to remove the motion estimation (ME) on unnecessary block sizes. The results demonstrated that the introduced approach reduces the computational complexity with almost the same coding efficiency compared with the original HM 2.0 reference software. In Ref. 6, an adaptive inter-mode decision approach was presented to reduce the encoding time of the HEVC standard by integrating three strategies including the early SKIP mode decision, prediction size correlation-based mode decision, and RD cost correlation-based mode decision. The proposed algorithm reduced the computational complexity by 49% to 52% on average with negligible RD performance degradation.

For the VVC standard, the work presented in Ref. 7 introduced a fast intra multitype tree (MTT) algorithm based on machine learning. The advanced method applied the random forest models to predict the intra MTT partition and skip the unnecessary ones using visual perception. The achieved results prove that the algorithm saved an important encoding time while keeping an effective coding performance. Dong et al.[1] suggested an intra-fast mode partition approach for VVC using both mode selection and prediction termination called adaptive mode pruning (AMP) and mode-dependent termination (MDT). In fact, the AMP stage deletes nonpromising prediction modes for the mode selection using various classifiers, and the MDT step skips the unnecessary prediction of the remaining CU levels. Experimental results demonstrate that the approach reduced the encoding time by 52% on average with a Bjontegaard bitrate (BDBR) increase from 0.93% to 1.08%. Yang et al.[2] introduced a fast intra-QTMT partitioning algorithm. A process called "early termination" was developed to determine the QTMT modes using the decision tree. The algorithm was utilized to control the transition from the given depth to the next using the different characteristics of each CU. The proposed algorithm reduced computational complexity by 63% on average with a BDBR increase of 1.93% and a Bjontegaad peak signal-to-noise ratio (BDPSNR) decrease of 0.11 dB. Another fast decision algorithm was developed at different CU sizes by Chen et al.[8] The approach implemented different SVM classifiers to determine the QTMT structure. In fact, the CU features were selected to identify the partitioning directions. Then, these features were applied to train the SVM models at different CU sizes.

Finally, the trained models were implemented to predict the QTMT partition structure. The proposed algorithm provided a significant gain in coding time, reaching 51.01% with a loss of 1.54% in BDBR. In Ref. 9, Zhang et al. elaborated a fast partition decision approach for a given CU based on the directed acyclic graph-support vector machine (DAG-SVM) model to reduce the partition complexity. The features were first extracted by encoding the video sequences on VTM4.0. These features were then used to train the DAG-SVM model. Finally, the classifier was implemented on VTM4.0 to predict the optimal partitioning modes. The results indicated that the proposed method saved 54.74% in coding time with a loss of 0.93% in BDBR. A fast method based on the CU partition and the intra-mode prediction was developed by Zhang et al.[10] This method included two strategies: a fast CU partition using the random forest classifier (RFC) model and a fast intra-prediction mode using the texture of regions. The algorithm was based on splitting a given CU with the RFC and analyzing its energy with four predefined prediction modes using the region features to skip unnecessary intra-prediction modes. The results indicated that the presented algorithm achieved a gain of 54.91% in encoding time accompanied by a loss of 0.93% in BDBR. In Ref. 11, Wu et al. introduced an SVM classifier for the intra-partition module. The algorithm removed the redundant partitions using the different textures of each block with several SVM classifiers being developed for different block sizes. Threshold values were set for each classifier to balance the coding complexity with the RD performance. This proposed approach helped to reduce the coding time from 30.78% to 63.16% with an increase from 1.10% to 2.71% in BDBR.

For the deep methods, several DL algorithms have been developed at the partitioning module using the neural network. Jin et al.[12] generated an intra QTBT partitioning approach based on the CNN. The classifier was implemented to predict the QTBT partitioning decision of each block sized 32×32, minimizing the RD cost computational complexity. The presented method offers a gain of about 42% in coding time with a loss of about 0.69% in BDBR. Also, Amna et al.[13] suggested a fast QT partition algorithm based on a CNN model by predicting the decision tree of each block instead of applying the rate distortion optimization (RDO) process. This approach developed a CNN architecture using the three partition levels of the QT structure for a better intra QTBT structure performance. Experimental results showed that this method decreased the coding time by 35% on average with a BDBR rise of 1.7%. Another work based on determining the adaptive CU partition was introduced in Ref. 14 using a neural network called pooling-variable CNN. The approach computed the residual CU gradient to select the appropriate QTMT decision. Three decisions were used: dividing the given CU, not dividing it, or applying the pooling CNN to make the partitioning decision tree. This algorithm reduced the computational complexity by 33% with an increase of 0.99% in BDBR. Tissier et al. developed another work based on a CNN to overcome the complexity of the partitioning module. The CNN architecture model was inspired by the ResNet network.[15] The CNN was used to analyze the CUs texture sized 64×64 and predict a vector of probabilities for the different boundaries of blocks 4×4. From these probabilities, a split probability was derived and compared with a predefined threshold value. The proposed method minimized the coding complexity by about 51% with an increase of 1.45% in bitrate. In Ref. 16, a network called multi-stage exit-CNN (MSE-CNN) was created to predict the intra QTMT partitioning structure. The MSE-CNN model predicted the CU decision tree by avoiding the Rdcost function through a multithreshold decision. The approach achieved a gain of 44.65% and 66.88% in encoding time with a BDBR increase of 1.322% and 3.188%, respectively, using two different threshold values. In Ref. 17, Tech et al. elaborated a fast intra-partition approach for the VVC encoder. A CNN was created to reduce the number of partition modes tested while maintaining the RD performance. The neural network estimated the parameters that limited the horizontal and vertical MTT partitions for each block sized 32×32. The proposed CNN was stimulated by the ResNet model[15] with a luminance block input sized 32×32. Experimental results showed a reduction in encoding time of about 50% with a loss of 0.9% in BDBR. In Ref. 18, a multibranch CNN approach based on intra QTMT partitioning was presented to deal with computational complexity. The approach aimed to predict the depth of the QTMT partitioning for each block of size 32×32 by applying two phases. Indeed, The CNN was used to predict the QT depth and determine the TT mode decision. Then, these prediction depths were adopted to ignore some QTMT partition combinations. The outcomes showed that the presented method reached a gain of 42.34% in coding time and a loss of

0.71% in BDBR compared with VTM6.1. In Ref. 19, Abdallah et al. developed a fast intra QTMT decision tree based on a neural network called early-terminated hierarchical CNN (ETH-CNN). The proposed algorithm predicted the QT partition depth of the CU 64×64 based on the partition decision of splitting or skipping the current CU. The current method reduced the encoding complexity by 32.96% accompanied by an increase in bitrate of 4.18% and a decrease of 0.18 dB in quality.

3 Overview of QTMT Partitioning Structure

In the VVC standard, each image is first split into square CTU blocks. The CTU within the QTMT structure is set to 128×128, which can be partitioned into different CU shapes. Each CU is presented under square blocks by the QT structure or rectangular blocks given by the added MTT partition structure.

The QT mode depths range from a 128×128 initial depth denoted QT0 to 8×8. In addition, the MTT decision mode includes BT and TT modes in horizontal and vertical splitting directions with block sizes ranging from 32×32 to 4×4. As detailed in Fig. 1, the CTU is first split into four square leaf nodes sized 64×64 corresponding to a QT depth of 1 denoted QT1. Then, each 64×64 CU is divided by the QT structure until reaching a 8×8 minimum QT shape, where the QT depth ranges from 1 to 4.

At the QT depth denoted QT2, the CU 32×32 can be partitioned by the BT and TT modes while respecting the maximum MTT depth fixed at 4. At this level, these modes have a depth equal to zero denoted BT0 and TT0, respectively. For the BT partition, the BT depth is varied from 0 to 4, producing a horizontal binary tree (BTH) and vertical binary tree (BTV) block partitioning and respecting the minimum MTT size 4×4 as presented in Fig. 1. Similarly, the TT depth range can reach TT4 for the TT partition, generating horizontal ternary tree (TTH) and vertical ternary tree (TTV) structures. In the case of the MTT partition structure starting, a rectangular block size is produced and the use of the QT structure is thus forbidden.

To select the optimal QTMT decision tree, the RDO process is applied for all CU sizes until the minimum size is obtained. The RDcost is first calculated for all block sizes. Then, the QTMT partition with the minimum RDcost is selected as an optimal decision tree.

4 Proposed Intra-QTMT Partition Algorithm

In this paper, an overall approach based on a deep QTMT partitioning decision for intra-coding is introduced to overcome computational complexity caused by the RDO processing. The overall approach focuses mainly on both the QT structure and the BT partition in the horizontal and vertical directions. Therefore, a CNN, named ETH-CNN, used from the state-of-the-art method is implemented at the QT mode, and two other CNNs called CNN-BTH and CNN-BTV are proposed to predict the BTH and BTV mode decisions, respectively.

4.1 *Intra-Partition Algorithm Based on QT Structure*

For the QT algorithm,[19] a deep network called ETH-CNN[20] is implemented in the QTMT processing to reduce the computational complexity caused by the QT structure.

As shown in Fig. 2, the ETH-CNN architecture is developed according to the three QT partitioning levels 64×64, 32×32, and 16×16. In each level, the luminance component 64×64 is preprocessed through two layers to decrease the variation of the input samples and to generate the corresponding QT partition level. Then, the preprocessing output is convoluted with three convolution layers to extract the various features of all CUs. In the first convolution layer, the low features of the splitting block are generated using 16 filters. The two following layers are designed to determine the high block features with 24 and 32 filters, respectively. Next, the high extracted features of the three levels are concatenated into one vector, which passes through a fully connected module to predict the decision tree map of each CU sized 64×64.

The ETH-CNN is retrained under our extracted database using the original VTM3.0 reference software to construct the partition map, which generates the QT decision depth of all 64×64 CUs as an output.

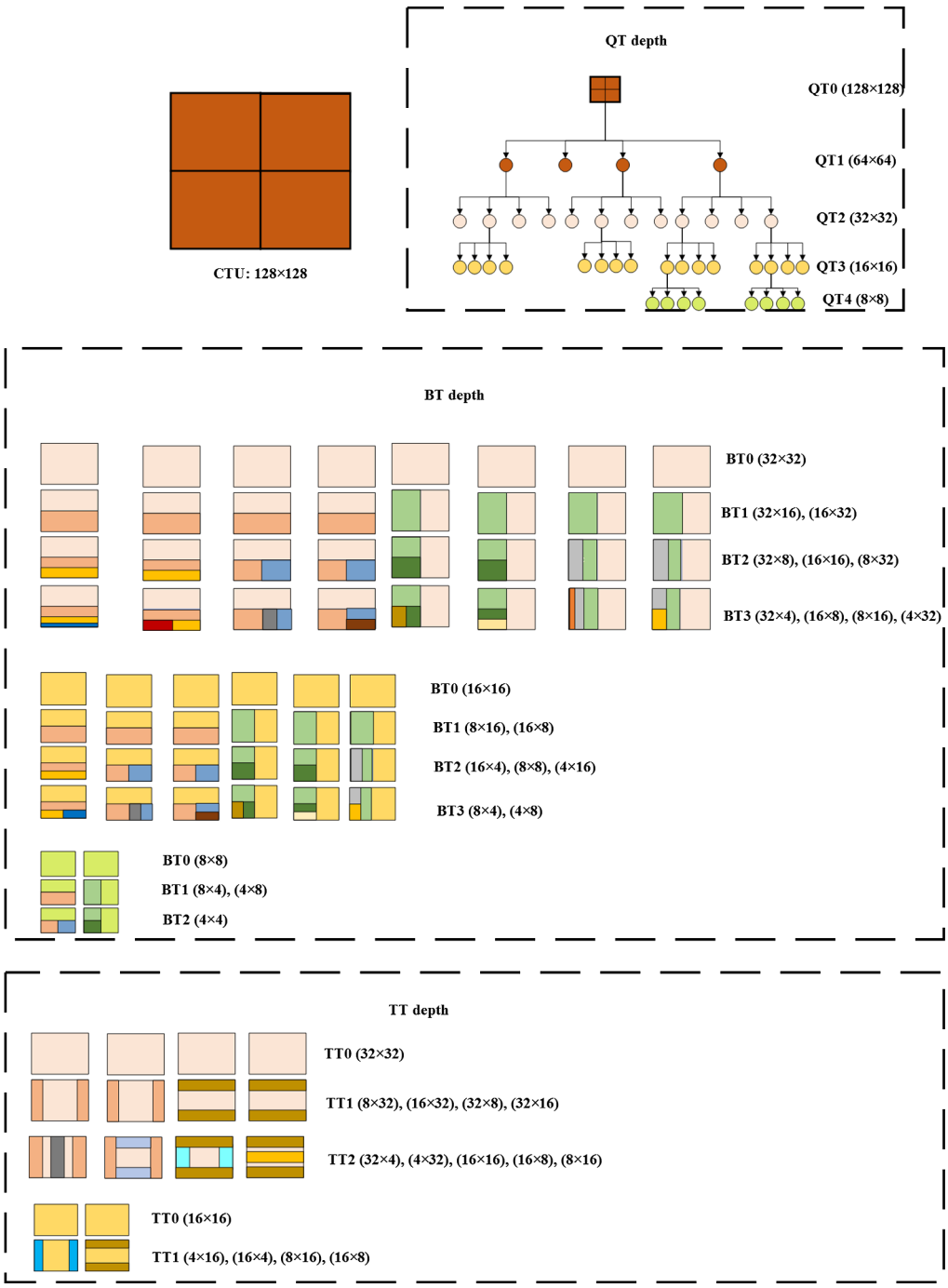

Fig. 1 QTMT partition structures.

At each level, a splitting probability is predicted to decide whether or not to split the current block into square CUs. To boost the algorithm performance, a bithreshold decision is used at each level to balance the encoding complexity with the compression performance. At the three CUs sized 64×64, 32×32, and 16×16, the predicted probability is compared with the predefined bithreshold as presented in Fig. 3. Two decisions are introduced: whether the current CU is divided into four square blocks directly without calculating the RDcost and checking the MTT modes or whether it skips the QT mode and allows for rectangular splitting. When the probability ranges between the bithreshold values, the RDcost metric is computed at the whole partition mode.

The proposed algorithm can reduce the encoding complexity of the QTMT module by applying the ETH-CNN to predict the QT depth instead of the RDO process.

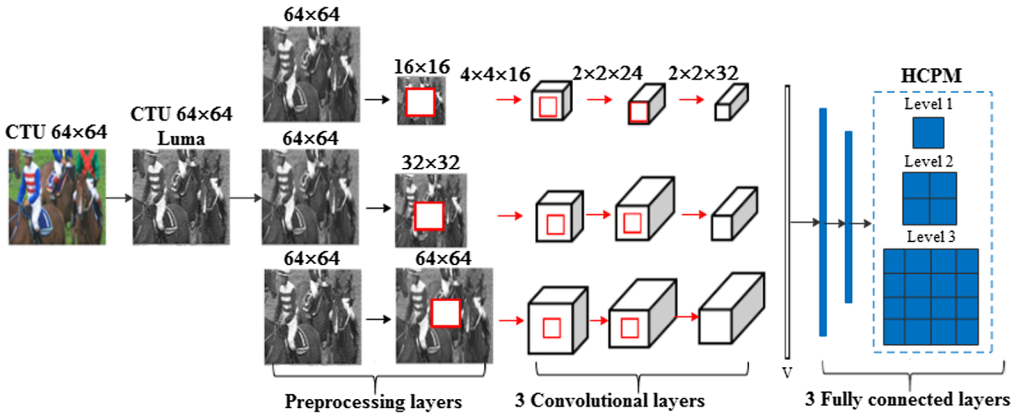

Fig. 2 ETH-CNN architecture.

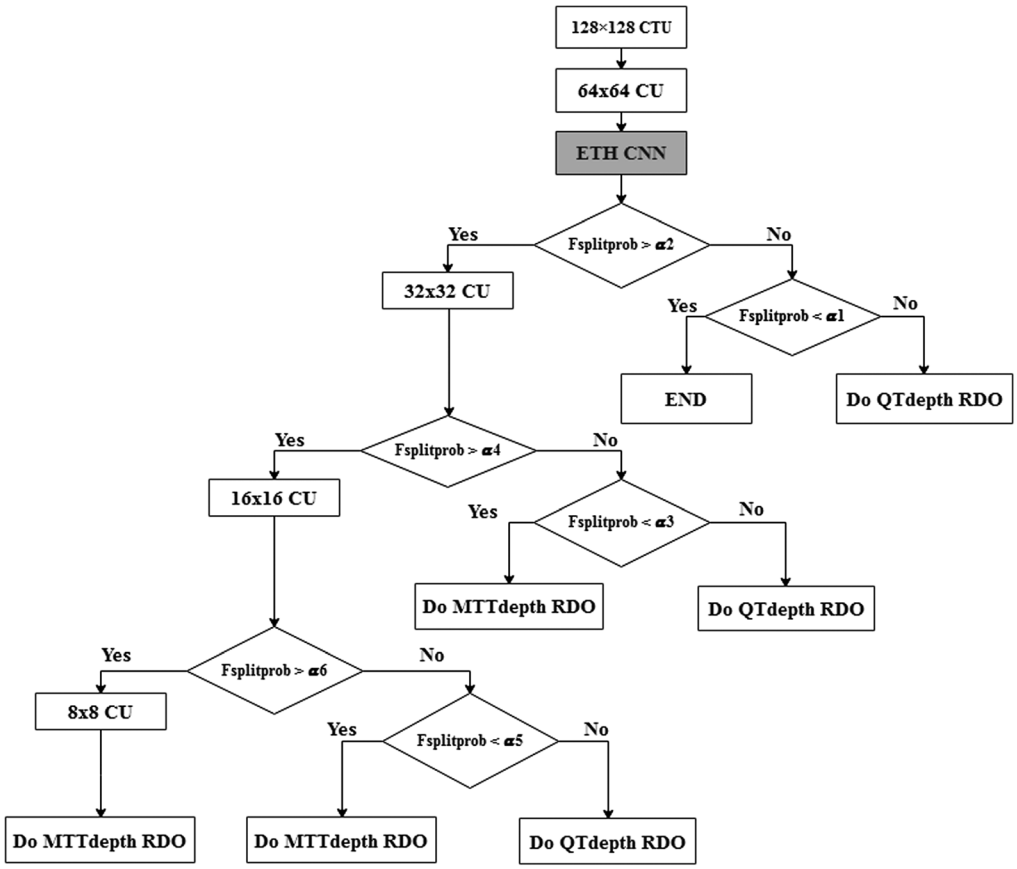

Fig. 3 Flowchart of QTMT partitioning based on ETH-CNN.

4.2 *Intra-Partition Algorithm Based on BT Structure*

For the BTH algorithm,[21] a CNN named CNN-BTH is proposed to optimize the intra-BTH splitting decision. In this part, we mainly focus on the three BTHs sized 32×32, 32×16, and 32×8 with BTH depths ranging from 0 to 2.

Therefore, our CNN-BTH architecture is developed under these three levels starting with an input block sized 32×32 as shown in Fig. 4. The input samples pass through three layers named preprocessing layers, convolutional layers, and fully connected layers. After the preprocessing step, the resulted 32×32, 32×16, and 32×8 BTH levels are convoluted with three layers to extract the different characteristics for all levels. After concatenating the generated features, three fully connected layers are applied to predict the BTH partition map of each 32×32 CU.

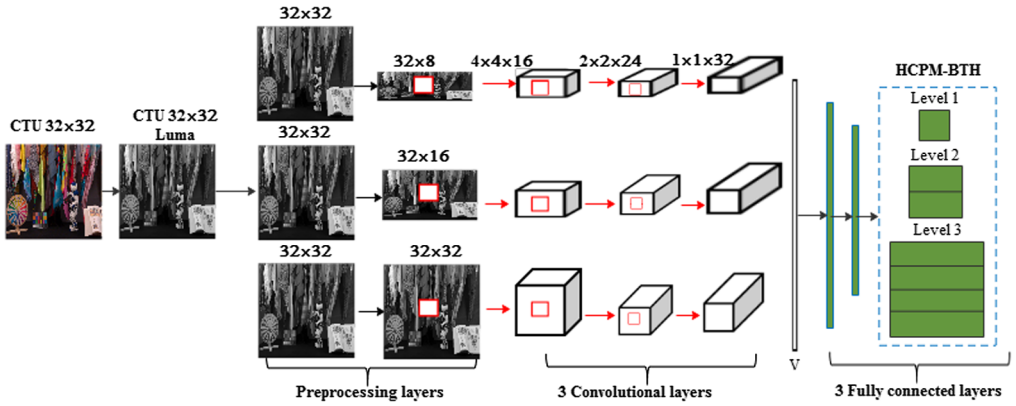

Fig. 4 CNN-BTH network architecture.

In the first stage, an image database is collected from an open database[22] and then coded with the original VTM3.0 reference software at the AI configuration to build the intended dataset. The image database is coded under four quantization parameters (QPs) with values 22, 27, 32, and 37 to generate the different BTH depths of each CU sized 32×32.

In the next stage, the proposed CNN-BTH network is trained under our database depths. Four models are created and then implemented in the developed algorithm to predict the BTH partition depth at each block sized 32×32. Figure 5 shows the BTH process at 32×32, 32×16, and 32×8, where a partition probability is predicted at each block. When the probability is higher than the fixed threshold, the current CU is split directly into two rectangular blocks in the horizontal direction, skipping both the RDcost computation and QT, BTV, TTH, and TTV modes verification.

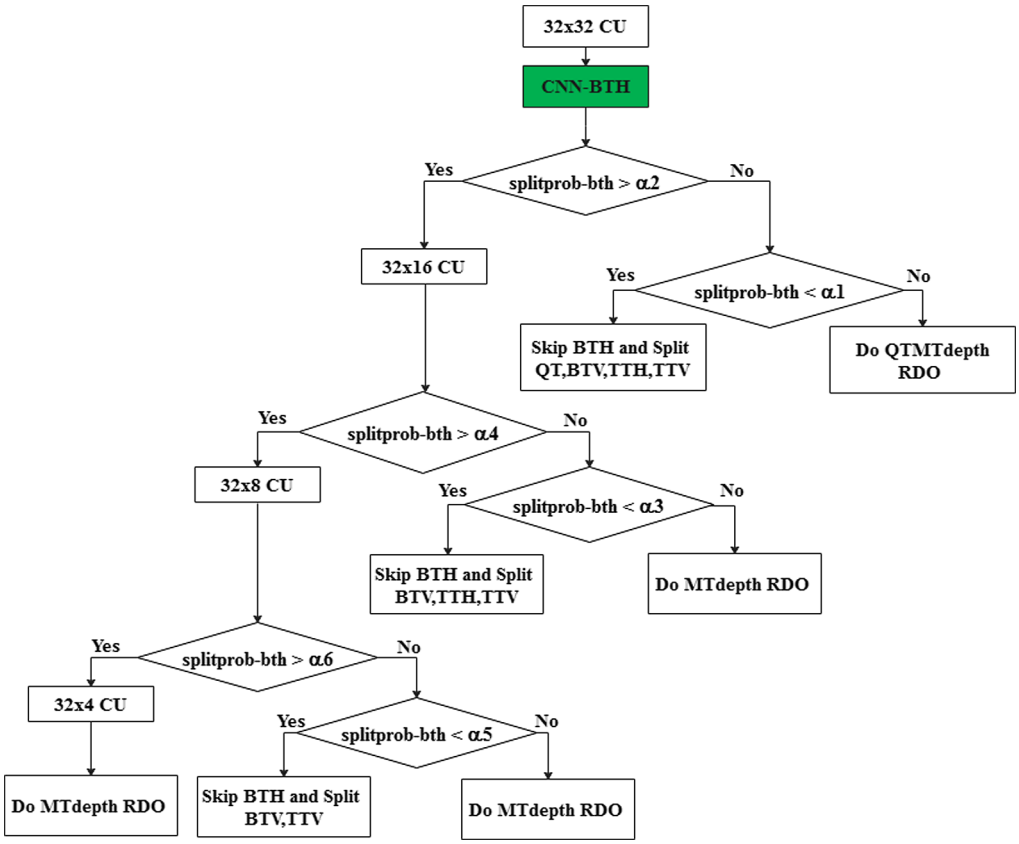

Fig. 5 Flowchart of QTMT partitioning based on CNN-BTH.

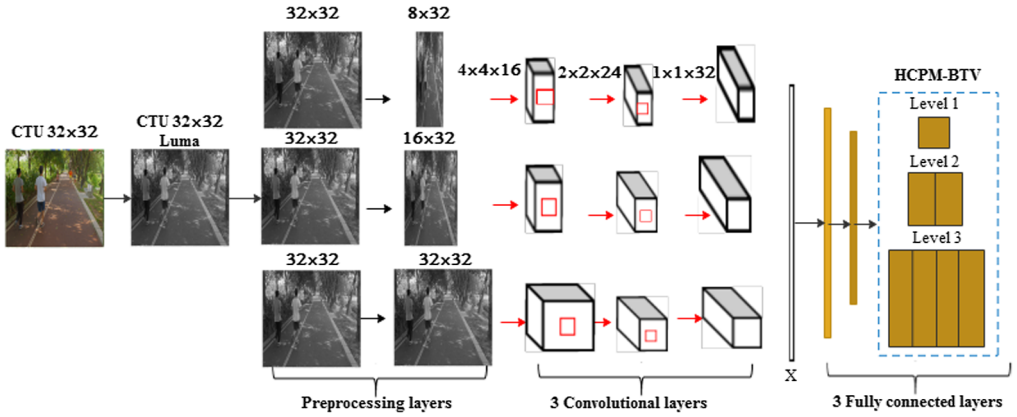

Fig. 6 CNN-BTV network architecture.

For the BTV algorithm,[21] a new convolution model called CNN-BTV is created according to the three partition levels sized 32×32, 16×32, and 8×32. Its architecture is similar to the CNN-BTH model as shown in Fig. 6, in which the output layer predicts the CU partition structure in the vertical direction producing 1, 2, and 4 partitioning probabilities at 32×32, 16×32, and 8×32 CUs, respectively. Before integrating the CNN-BTV, it is necessary to train the split or not split decision at each level from our generated database. The latter contains all of the 32×32 CUs BTV depths extracted from the original VTM3.0 algorithm. The trained CNN-BTV model is then implemented in the QTMT module to optimize the RDcost processing as shown in Fig. 7, for which a decision of whether splitting the current block directly or activating the RDO process is taken at the CUs sized 32×32, 16×32, and 8×32.

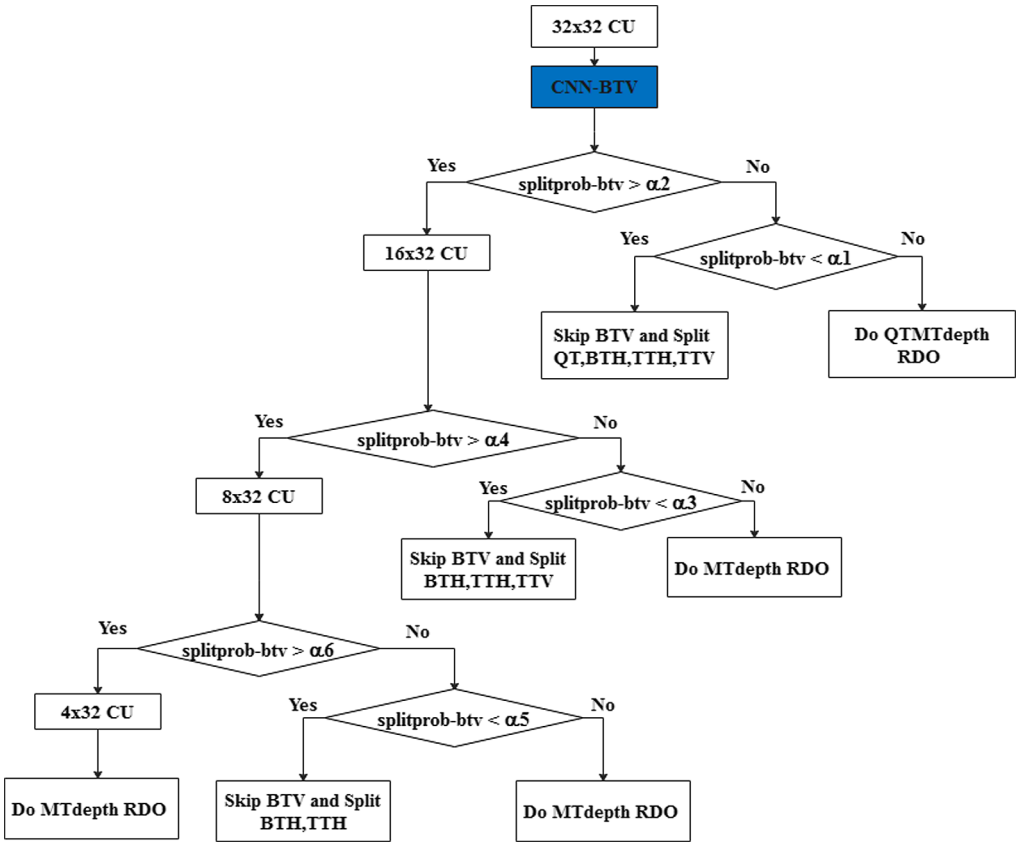

Fig. 7 Flowchart of QTMT partitioning based on CNN-BTV.

4.3 *Overall Proposed QTMT Algorithm Based on CNN*

In this work, an intra-partition decision tree algorithm is proposed to determine the QTMT structure based on the three CNNs called ETH-CNN, CNN-BTH, and CNN-BTV.

The used configuration of each deep neural network structure is detailed in Table 1, which presents the different input sizes, the number of filters, the kernel sizes for convolution, and the type of activation layers. As detailed in Table 1, each model includes three convolutional layers and three fully connected layers divided into two hidden fully connected layers and one output layer. It is noted that the activation function rectified linear units (ReLU)[23] is used at the convolutional and hidden fully connected processing, whereas the output fully connected unit is activated using the sigmoid function to generate the binary labels at each level. In fact, the ETH-CNN output contains 1, 4, and 16 binary labels at levels 1, 2, and 3, respectively, whereas the CNN-BTH and CNN-BTV outputs provide 1, 2, and 4 binary elements.

After training, these CNN models are implemented in the VTM3.0 reference software to handle the QTMT partitioning complexity at the QT and BT structures as presented in the following part.

At the CU sized 64×64, the ETH-CNN is implemented to predict the QT structure until arriving at the 8×8 size. In addition to the ETH-CNN model, the CNN-BTH and CNN-BTV networks are integrated at the 32×32 level to finally predict the QTMT decision tree at each CTU sized 128×128. The ultimate QTMT decision is based on a bithreshold used at each level o achieve a trade-off between the RD performance and the complexity reduction. Therefore, different bithresholds named [$\alpha 1$, $\alpha 2$], [$\alpha 3$, $\alpha 4$], and [$\alpha 5$, $\alpha 6$] are used at 64×64, 32×32, and 16×16, respectively, for the QT structure and at 32×32, 32×16, or 16×32 and 32×8 or 8×32 for the BT structure, respectively. The proposed intra-QTMT approach is described in Fig. 8 and explained as follows:

Table 1 Configuration of implemented networks ETH-CNN, CNN-BTH, and CNN-BTV.

Networks	Layers	Input size	Number of filters	Kernel size	Activation	
ETH-CNN	Convolution 1	Level 1	16 × 16	16	4 × 4	ReLU
		Level2	32 × 32	16	4 × 4	
		Level3	64 × 64	16	4 × 4	
	Convolution 2	Level 1	4 × 4	24	2 × 2	ReLU
		Level2	8 × 8	24	2 × 2	
		Level3	16 × 16	24	2 × 2	
	Convolution 3	Level 1	2 × 2	32	2 × 2	ReLU
		Level2	4 × 4	32	2 × 2	
		Level3	8 × 8	32	2 × 2	
	Fully connected 1	Level 1	2688			ReLU
		Level2	2688			
		Level3	2688			
	Fully connected 2	Level 1	64			ReLU
		Level2	128			
		Level3	256			
	Fully connected 3	Level 1	48			Sigmoid
		Level2	96			
		Level3	192			

Table 1 (*Continued*).

Networks	Layers	Input size	Number of filters	Kernel size	Activation	
CNN-BTH	Convolution 1	Level 1	32 × 8	16	4 × 4	ReLU
		Level2	32 × 16	16	4× 4	
		Level3	32 × 32	16	4 × 4	
	Convolution 2	Level 1		24	2 × 2	ReLU
		Level2		24	2 × 2	
		Level3		24	2 × 2	
	Convolution 3	Level 1	4 × 1	32	1 × 1	ReLU
		Level2	4 × 2	32	1 × 1	
		Level3	4 × 4	32	1 × 1	
	Fully connected 1	Level 1	1568	—	—	ReLU
		Level2	1568			
		Level3	1568			
	Fully connected 2	Level 1	64			ReLU
		Level2	128			
		Level3	256			
	Fully connected 3	Level 1	48			Sigmoid
		Level2	96			
		Level3	192			
CNN-BTV	Convolution 1	Level 1	8 × 32	16	4 × 4	ReLU
		Level2	16 × 32	16	4 × 4	
		Level3	32 × 32	16	4 × 4	
	Convolution 2	Level 1	—	24	2 × 2	ReLU
		Level2	—	24	2 × 2	
		Level3	—	24	2 × 2	
	Convolution 3	Level 1	4 × 1	32	1 × 1	ReLU
		Level2	4 × 2	32	1 × 1	
		Level3	4 × 4	32	1 × 1	
	Fully connected 1	Level 1	1568	—	—	ReLU
		Level2	1568			
		Level3	1568			
	Fully connected 2	Level 1	64			ReLU
		Level2	128			
		Level3	256			
	Fully connected 3	Level 1	48			Sigmoid
		Level2	96			
		Level3	192			

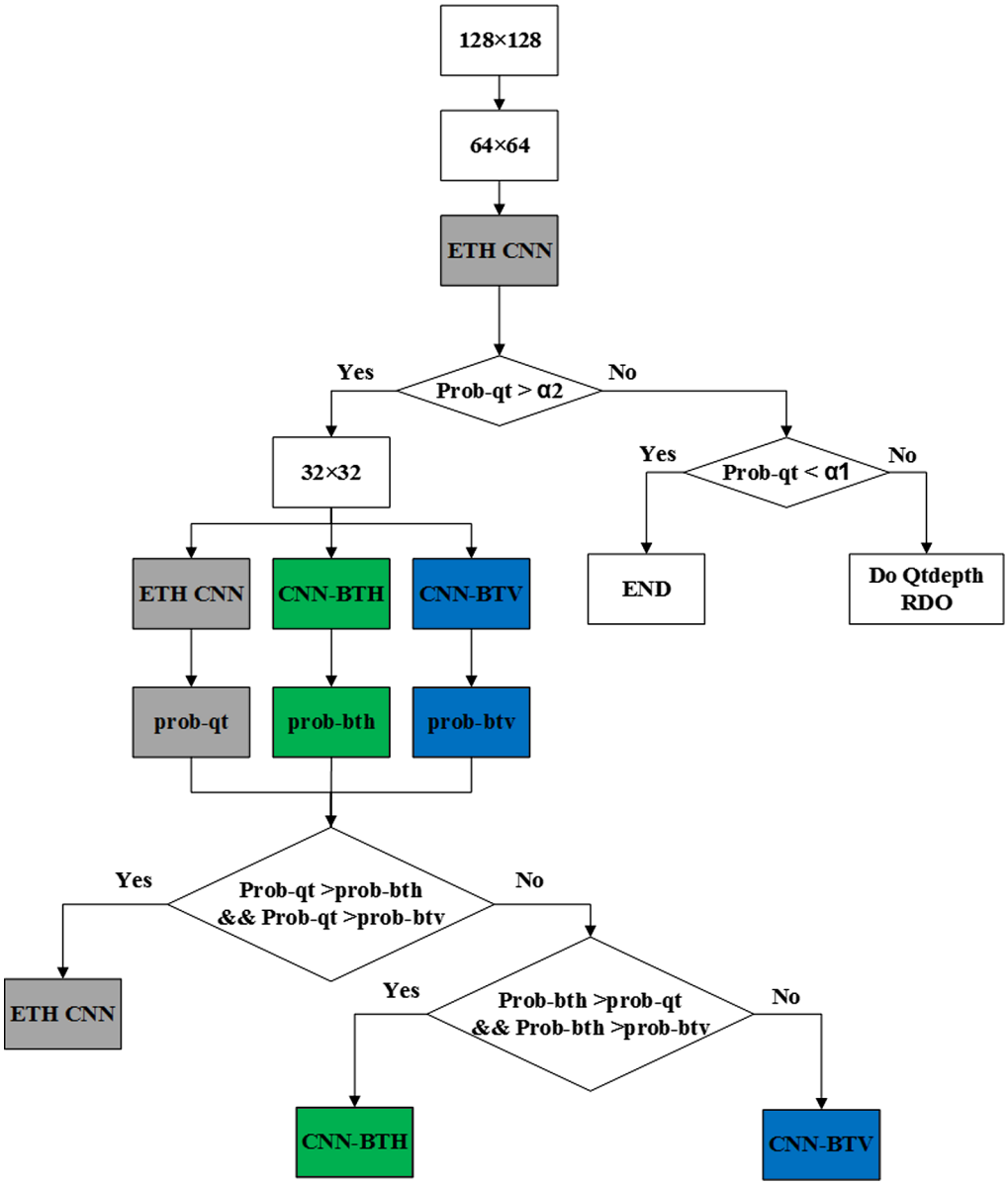

Fig. 8 Flowchart of CNNs-based QTMT partition.

Step 1: The current 128×128 CTU is split into four 64×64 sub-blocks directly without calculating the RDcost. At each 64×64 level, the ETH-CNN determines whether or not to divide the current CU by the QT structure according to a bithreshold [$\alpha 1$, $\alpha 2$].

Step 2: At the 64×64 level, if the probability predicted by the ETH-CNN named prob-qt $> \alpha 2$, the 64×64 CU is split directly into four 32×32 sub-blocks. If prob-qt $< \alpha 1$, the current CU is not divided via the QT structure. Because the maximum size of the MT structure is set to 32×32, the partition processing is completed at this level. If $\alpha 1 <$ prob $-$ qt $< \alpha 2$, the RDcost is computed at the CU 64×64.

Step 3: At the 32×32 level, the three networks ETH-CNN, CNN-BTH, and CNN-BTV are integrated to predict the partition probabilities and are denoted prob-qt, prob-bth, and prob-btv, respectively. In this case, a comparison is applied to determine the best partition scheme at the 32×32 CU. If the conditions prob-qt > prob-bth and prob-qt > prob-btv are verified, the partition processing is pursued through the ETH-CNN. If prob-bth > prob-qt and prob-bth > prob-btv, then the decision tree at 32×32 is determined by the CNN-BTH network.

If the previous conditions are not verified, the CNN-BTV network is applied to select the partition structure of the current CU.

5 Experimental Results

5.1 *Experimental Conditions*

To evaluate the efficiency of the proposed approach in terms of encoding time and RD performance, the simulations were executed under the VTM3.0 reference software.[24] The performance of our algorithm was evaluated on 22 videos with six different resolutions. Thus, six classes introduced by the joint collaborative team on video coding were tested[25] including classes A1 (3840×2160), A2 (3840×2160), B (1920×1080), C (832×480), D (416×240), and E (1280×720), where the ultra high definition (UHD) video sequences of classes A1 and A2 are defined by the VVC standard and classes B to E are previously used in the HEVC codec. The number of frames taken in each video sequence is equivalent to 161. These frames were coded with a QP value varying between 22, 27, 32, and 37 at the AI configuration. Experimental results were conducted on a Windows 7 OS platform with Intel ® E5-1620 v3 @ 3.50 GHz CPU and 32 GB RAM. The resulted coding performances are highlighted in Tables 2 and 3, where the encoding time is defined on ΔT, the RD performances are evaluated on ΔBR, BDBR,[26] ΔPSNR, and BDPSNR.[26] ΔT, ΔBR, and ΔPSNR are calculated as

Table 2 Evaluation of ΔBR, ΔPSNR, and ΔT of the proposed algorithm compared with the original software VTM3.0.

Class	Sequence	ΔBR (%)	ΔPSNR (dB)	ΔT (%)
A1	Tango2	3.27	−0.027	−55.51
	FoodMarket4	3.9	−0.027	−42.55
	Campfire	1.74	−0.073	−40.89
A2	Catrobot1	2.47	−0.048	−50.61
	DaylightRoad2	2.6	−0.051	−54.4
	ParkRunning3	2.23	−0.071	−21.45
	Average	**2.70**	**−0.049**	**−44.24**
B	BasketballDrive	2.83	−0.05	−48.12
	BQTerrace	1.62	−0.039	-30.7
	Cactus	1.48	−0.023	−30.78
	Kimono	2.39	−0.026	−35.94
	ParkScene	1.37	−0.032	−35.42
C	BasketballDrill	1.37	−0.049	−26.51
	BQMall	1.6	−0.048	−25.79
	PartyScene	0.9	0.033	−25.32
	RaceHorsesC	1.22	−0.024	−29.36
D	BasketballPass	2.21	−0.089	−31.34
	BlowingBubbles	0.2	−0.036	−24.08
	RaceHorses	1.04	−0.023	−21.95
E	FourPeople	2.07	−0.061	−37.77
	Johnny	2.45	−0.09	−33.4
	KristenAndSara	2.61	−0.053	−31.64
	Average	**1.69**	**−0.04**	**−31.21**
Average		**1.97**	**−0.043**	**−34.93**

Note: The bold characters are used to highlight the average results.

Table 3 Proposed algorithm performance BDBR and BDPSNR compared with VTM3.0.

Class	Sequence	BDBR (%)	BDPSNR (dB)
A1	Tango2	4.54	−0.06
	FoodMarket4	5.02	−0.14
	Campfire	3.51	−0.1
A2	Catrobot1	4.14	−0.1
	DaylightRoad2	3.65	−0.08
	ParkRunning3	2.75	−0.13
	Average	**3.93**	**−0.1**
B	BasketballDrive	4.5	−0.11
	BQTerrace	2	−0.09
	Cactus	1.98	−0.06
	Kimono	3.08	−0.1
	ParkScene	2.23	−0.09
C	BasketballDrill	2.38	−0.1
	BQMall	2.48	−0.13
	PartyScene	1.52	−0.11
	RaceHorsesC	1.73	−0.1
D	BasketballPass	4.07	−0.22
	BlowingBubbles	0.84	−0.04
	RaceHorses	1.55	−0.09
E	FourPeople	3.60	−0.19
	Johnny	4.96	−0.19
	KristenAndSara	3.8	−0.16
	Average	**2.71**	**−0.11**
Average		**3.06**	**−0.11**

Note: The bold characters are used to highlight the average results.

$$\Delta T(\%) = \frac{T(p) - T(o)}{T(o)} \times 100, \tag{1}$$

$$\Delta BR(\%) = \frac{BR(p) - BR(o)}{BR(o)} \times 100, \tag{2}$$

$$\Delta \mathrm{PSNR(dB)} = \mathrm{PSNR}(p) - \mathrm{PSNR}(o), \tag{3}$$

where $\mathrm{BR}(p)$, $\mathrm{PSNR}(p)$, and $T(p)$ represent the bitrate, the quality, and the coding time of the proposed algorithm, respectively. $\mathrm{BR}(o)$, $\mathrm{PSNR}(o)$, and $T(o)$ correspond to the bitrate, the quality, and the coding time of the original algorithm, respectively.

5.2 *Experimental Results*

Table 2 illustrates the coding performance of our proposed mode decision approach compared with the original VTM3.0 reference software in terms of ΔBR, ΔPSNR, and ΔT.

It can be observed that our deep QTMT algorithm brings a considerable gain in encoding time of about 35% on average accompanied by a negligible RD performance degradation of only 1.97% in ΔBR and 0.043 dB in ΔPSNR, which does not exceed 0.1%.

In addition, UHD classes A1 and A2 reduce the encoding complexity by 44.24% on average with a slight ΔBR increase of 2.7% and a negligible ΔPSNR degradation of 0.049 dB. For video sequences from classes B to E, 31.21% of complexity reduction is reached with only a loss of 1.69% and 0.04 dB in bitrate and quality, respectively.

In terms of encoding time saving, the video sequence Tango2 from class A1 results in a maximum complexity reduction of 55.51% accompanied by a rise of 3.27% in ΔBR and a decrease of 0.027 dB in ΔPSNR. The presented results can be explained by the high resolution of Tango sequence that tends to be partitioned into larger CUs, where less partition modes are checked due to the best decision taken by the used models.

In can be seen from the experimental results that the ParkScene video sequence reaches the best trade-off between complexity reduction and RD performance. In fact, it attains an important gain in coding time of 35.42% with an acceptable loss in bitrate and quality of only 1.37% and 0.032 dB, respectively. This result is confirmed by the training database resolutions, which are mainly composed of 1/2 images with HD resolutions. The video ParkScene also contains homogeneous blocks.

The worst case is observed for the Johnny video from class E, for which the ΔBR increase can reach 2.45% with a ΔPSNR decrease of 0.09 dB while it saves a significant encoding time of 33.4%. The Johnny video sequence is marked by its textured region that needs to be split into smaller CUs. Thus, the time saving with the quality degradation may be interpreted by the false prediction of the CNNs.

All of the presented results can be explained by the fixed bithreshold values at each level, where these thresholds are employed in the intra-partition process to determine whether the current CU should be partitioned or not. Multivariations of the bithreshold are tested under various videos to choose the best values according to the trade-off between the bitrate, the quality, and the computational complexity. So, experimental results are mainly based on the probabilities predicted by the CNNs and the bithreshold values.

The encoding performance of the implemented algorithm in terms of bitrate and quality is further evaluated through BDBR and BDPSNR as indicated in Table 3.

It is noted that an acceptable increase in BDBR of 3.06% on average is reached with a little quality degradation of 0.11 dB imperceptible by the human eye. In fact, in the best case, the video sequences from classes A1 and A2 show a negligible quality degradation of 0.1 dB on average, varying from 0.06 to 0.14 dB, and an adequate increase of 3.93% on average. Even for videos from classes B to E, the degradation is satisfactory by 2.71% in terms of BDBR and 0.11 dB in BDPSNR varying from 0.04 to 0.22 dB.

All of the previous results prove that our proposed QTMT partition decision for CU coding achieves a balance between the computational complexity and the RD performance.

5.3 *Performance Comparison with Previous Works in VVC*

To assess the coding efficiency of the proposed approach for intra-CU partitioning compared with the state-of-the-art methods, Table 4 compares the results obtained in this paper with those of the previous algorithms namely Amna et al.,[13] Tang et al.,[14] Amestoy et al.,[28] Liu et al.,[29] and Fu et al.[27]

The developed approach outperforms the VVC algorithms including Amna et al.,[13] Tang et al.,[14] Amestoy et al.,[28] and Fu et al.[27] in terms of computational complexity reduction. Compared with machine learning methods, namely Refs. 27 and 28, the proposed method minimizes the encoding time by 5.15% and 6.79, respectively. In addition, our algorithm saves about 35% of encoding time with a high BDBR increase compared with 33.85%[13] and 33.41%[14] in CNN works. It can be concluded that our algorithm achieves an important complexity reduction at the cost of an acceptable BDBR increase. Moreover, the developed work[2] implemented on VVC saves 56.49% on average at the expense of a 4.41% BDBR increase and a degradation of 0.17 dB in BDPSNR. In fact, the proposed QTMT partition decision outperforms the previous work in terms of video quality by 0.06 dB and encoding bitrate by 1.35% accompanied by less

Table 4 Evaluation of the proposed approach compared with the previous algorithms in VVC.

	BDBR (%)	BDPSNR (dB)	ΔT (%)	BDBR (%)	BDPSNR (dB)	ΔT (%)	BDBR (%)	BDPSNR (dB)	ΔT (%)
		Fu et al.[27]			Amna et al.[13]			Tang et al.[14]	
Class A1	0.33	−0.005	−22.76	—	—	—	1.05	—	−34.96
Class A2	0.40	−0.01	− 26.53	3.25	—	−37.65	—	—	—
Class B	0.30	−0.09	−31.67	2.12	—	−34.02	0.88	—	−34.41
Class C	0.44	−0.028	−27.84	0.77	—	−34.92	0.74	—	−29.37
Class D	0.36	−0.02	−30.76	0.65	—	−32.22	0.78	—	−32.92
Class E	0.57	−0.02	−27.02	2.26	—	−34.3	1.5	—	−36.42
Average	0.46	−0.04	−28.14	1.44	—	−33.85	0.99	—	−33.41
		Amestoy et al.[28]			Yang et al.[2]			Proposed algorithm	
Class A1	0.95	—	−29.3	4.39	−0.11	−49	4.35	−0.1	−46.31
Class A2	0.46	—	−30.9	5.47	−0.14	−53.61	3.51	−0.1	−42.15
Class B	0.59	—	−25.47	4.95	−0.15	−61.71	2.75	−0.09	−36.19
Class C	0.59	—	−25.47	3.63	−0.18	−58.89	2.02	−0.11	−26.75
Class D	0.35	—	−28.95	2.30	−0.13	−52.34	2.15	−0.11	−25.79
Class E	0.49	—	−31.05	5.98	−0.26	−62.59	4.12	−0.18	−34.27
Average	0.61	—	−29.78	4.41	−0.17	−56.49	3.06	−0.11	−34.93

encoding time saving. These results can be considered to be a compromise between the computational complexity and the coding efficiency. This comparison with the state-of-the-art methods proves that the proposed algorithm is efficient as it realizes an important encoding time with an acceptable decrease in coding efficiency.

6 Conclusion

This paper presents an intra-fast mode decision algorithm for the VVC standard based on CNN to reduce the computational complexity of the QTMT partitioning process. A QTMT partition method is introduced to predict the QT depth at each CU 64×64 and the BT depth at CU sized 32×32 in the horizontal and vertical directions instead of the RDO processing using different CNNs. This approach is able to save a significant encoding time of up to 55.51% compared with the original VTM3.0 reference software, with a gain of 34.93% on average accompanied by an acceptable BDBR increase and a negligible video quality degradation. Our algorithm shows good results in encoding efficiency and complexity reduction compared with the state-of-the-art methods. To enhance the RD performance and the time saving of the proposed approach, we will mainly focus on adjusting the choice of the bithreshold values in future work.

References

1. X. Dong et al., "Fast intra mode decision algorithm for versatile video coding," *IEEE Trans. Multimedia* **24**, 400–414 (2021).
2. H. Yang et al., "Low complexity CTU partition structure decision and fast intra mode decision for versatile video coding," *IEEE Trans. Circuits Syst. Video Technol.* **30**, 1668–1682 (2019).

3. H. Kibeya et al., "Fast intra-prediction algorithms for high efficiency video coding standard," *J. Electron. Imaging* **25**(1), 013028 (2016).
4. L. Shen, Z. Zhang, and Z. Liu, "Effective CU size decision for HEVC intracoding," *IEEE Trans. Image Process.* **23**(10), 4232–4241 (2014).
5. L. Shen et al., "An effective CU size decision method for HEVC encoders," *IEEE Trans. Multimedia* **15**(2), 465–470 (2012).
6. L. Shen, Z. Zhang, and Z. Liu, "Adaptive inter-mode decision for HEVC jointly utilizing inter-level and spatiotemporal correlations," *IEEE Trans. Circuits Syst. Video Technol.* **24**(10), 1709–1722 (2014).
7. M.-J. Chen et al., "Efficient partition decision based on visual perception and machine learning for H.266/versatile video coding," *IEEE Access* **10**, 42141–42150 (2022).
8. F. Chen et al., "A fast CU size decision algorithm for VVC intra prediction based on support vector machine," *Multimedia Tools Appl.* **79**(37), 27923–27939 (2020).
9. Q. Zhang et al., "Fast CU partition decision for H.266/VVC based on the improved dag-svm classifier model," *Multimedia Syst.* **27**(1), 1–14 (2021).
10. Q. Zhang et al., "Fast CU partition and intra mode decision method for H.266/VVC," *IEEE Access* **8**, 117539–117550 (2020).
11. G. Wu et al., "SVM based fast CU partitioning algorithm for VVC intra coding," in *IEEE Int. Symp. Circuits and Syst. (ISCAS)*, IEEE, pp. 1–5 (2021).
12. Z. Jin et al., "Fast QTBT partition algorithm for intra frame coding through convolutional neural network," *IEEE Access* **6**, 54660–54673 (2018).
13. M. Amna et al., "Fast intra-coding unit partition decision in H.266/FVC based on deep learning," *J. Real-Time Image Process.* **17**(6), 1971–1981 (2020).
14. G. Tang et al., "Adaptive CU split decision with pooling-variable CNN for VVC intra encoding," in *IEEE Vis. Commun. Image Process.*, pp. 1–4 (2019).
15. K. He et al., "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.*, pp. 770–778 (2016).
16. T. Li et al., "DeepQTMT: a deep learning approach for fast QTMT-based CU partition of intra-mode VVC," *IEEE Trans. Image Process.* **30**, 5377–5390 (2021).
17. G. Tech et al., "Fast partitioning for VVC intra-picture encoding with a CNN minimizing the rate-distortion-time cost," in *Data Compression Conf. (DCC)*, IEEE, pp. 3–12 (2021).
18. P.-C. Fu et al., "Two-phase scheme for trimming QTMT CU partition using multi-branch convolutional neural networks," in *IEEE 3rd Int. Conf. Artif. Intell. Circuits and Syst. (AICAS)*, IEEE, pp. 1–6 (2021).
19. B. Abdallah et al., "Low-complexity QTMT partition based on deep neural network for versatile video coding," *Signal, Image Video Process.* **15**, 1153–1160 (2021).
20. M. Xu et al., "Reducing complexity of HEVC: a deep learning approach," *IEEE Trans. Image Process.* **27**(10), 5044–5059 (2018).
21. B. Abdallah et al., "Fast QTMT decision tree for versatile video coding based on deep neural network," *Multimedia Tools Appl.* 1–17 (2022).
22. D.-T. Dang-Nguyen et al., "Raise: a raw images dataset for digital image forensics," in *Proc. 6th ACM Multimedia Syst. Conf.*, pp. 219–224 (2015).
23. X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. Fourteenth Int. Conf. Artif. Intell. and Stat., JMLR Workshop and Conf. Proc.*, pp. 315–323 (2011).
24. Joint Video Experts Team, "VVC Test Model (VTM) vesion 3.0," https://vcgit.hhi.424 fraunhofer.de/jvet/VVCSoftware_VTM/tree/VTM-3.0 (accessed Dec. 2018).
25. F. Bossen et al., "JVET common test conditions and software reference configurations for SDR video," Joint Video Experts Team (JVET) of ITU-T SG 16 (2018).
26. G. Bjøntegaard, "Calculation of average PSNR differences between RD-curves (VCEG-M33)," in *VCEG Meeting (ITU-T SG16 Q. 6)*, pp. 2–4 (2001).
27. T. Fu et al., "Fast CU partitioning algorithm for H.266/VVC intra-frame coding," in *IEEE Int. Conf. Multimedia and Expo (ICME)*, pp. 55–60 (2019).
28. T. Amestoy et al., "Tunable VVC frame partitioning based on lightweight machine learning," *IEEE Trans. Image Process.* **29**, 1313–1328 (2019).

29. X. Liu et al., "An adaptive CU size decision algorithm for hevc intra prediction based on complexity classification using machine learning," *IEEE Trans. Circuits Systems Video Technol.* **29**(1), 144–155 (2017).

Bouthaina Abdallah received her electrical engineering degree from the National Engineering School of Sfax (ENIS), Tunisia, in 2018. Since 2019, she has joined the Electronics and Information Technology Laboratory (LETI), Sfax. She received her PhD in electronic and microelectronic engineering in 2021. Her research interests include video coding and compression, potential video coding standards and codecs, and deep learning techniques.

Fatma Belghith received her degree in electrical engineering from the National School of Engineering (ENIS), Sfax, Tunisia, in 2012. She received her PhD in electronic engineering in 2016. She is currently an assistant professor at the Faculty of Sciences and Techniques of Sidi Bouzid, Tunisia. Her current research interests include video coding with an emphasis on HEVC standard and beyond, hardware implementation using FPGA, and embedded systems technology.

Mohamed Ali Ben Ayed received his BS degree in computer engineering from Oregon State University; his MS degree in electrical engineering from Georgia Institute of Technology in 1988; and his DEA, PhD, and HDR degrees in electronics engineering from Sfax National School of Engineering in 1998, 2004, and 2008, respectively. He is currently a Maître de Conferences in the Department of Communication at Sfax High Institute of Electronics and Communication. He was a cofounder of Ubvideo Tunisia in the techno-pole El-GHAZLA Tunis, an international leader company in the domain of video coding technology. He has been a member of a research team since 1994 at (LETI, Sfax) in the domain of electronics and information technology and a reviewer in many international and national journals and conferences. He is currently a technical advisor of EBREASK video, a research and development company specializing in the next high-efficient video coding generation H265. His current research interests include DSP and VHDL implementation of digital algorithms for multimedia services and development of digital video compression algorithms.

Nouri Masmoudi received his electrical engineering degree from the Faculty of Sciences and Techniques - Sfax, Tunisia, in 1982 and his DEA degree from the National Institute of Applied Sciences-Lyon and University Claude Bernard, Lyon, France, in 1984. From 1986 to 1990, he prepared his thesis at the Laboratory of Power Electronics (LEP) at the National School Engineering of Sfax (ENIS). He received his PhD from the National School Engineering of Tunis (ENIT), Tunisia, in 1990. From 1990 to 2000, he was an assistant professor in the Electrical Engineering Department-ENIS. Since 2000, he has been an associate professor and head of the Circuits and Systems group in the Laboratory of Electronics and Information Technology. Since 2003, he has been responsible for the Electronic Master Program at ENIS. His research activities have been devoted to several topics including design, telecommunication, embedded systems and information technology, video coding (motion estimation, Mode Decision, H.264 Standard), and image processing (wavelet image compression, subband image coding, image interpolation, and denoising).