J. Vis. Commun. Image R. 87 (2022) 103542

Contents lists available at ScienceDirect

Journal of Visual Communication and Image Representation

journal homepage: www.elsevier.com/locate/jvci

Full Length Article

Fast coding unit partitioning algorithms for versatile video coding intra coding☆

Jiunn-Tsair Fang [a,*], Bang-Hao Liu [b], Pao-Chi Chang [b]

[a] Ming Chuan University, Department of Electronic Engineering, No. 5, Deming Rd., Taoyuan City 33348, Taiwan
[b] National Central University, Department of Communication Engineering, No. 300, Jhongda Rd., Taoyuan City 32001, Taiwan

ARTICLE INFO

Keywords:
Versatile video coding
Quadtree with nested multi-type tree
Coding unit
Intra coding
Convolutional neural network

ABSTRACT

Versatile video coding (VVC) is the newest video compression standard. It adopts quadtree with nested multi-type tree (QT-MTT) to encode square or rectangular coding units (CUs). The QT-MTT coding structure is more flexible for encoding video texture, but it is also accompanied by many time-consuming algorithms. So, this work proposes fast algorithms to determine horizontal or vertical split for binary or ternary partition of a 32×32 CU in the VVC intra coding to replace the rate-distortion optimization (RDO) process, which is time-consuming. The proposed fast algorithms are actually a two-step algorithm, including feature analysis method and deep learning method. The feature analysis method is based on variances of pixels, and the deep learning method applies the convolution neural networks (CNNs) for classification. Experimental results show that the proposed method can reduce encoding time by 28.94% on average but increase Bjontegaard delta bit rate (BDBR) by about 0.83%.

1. Introduction

High efficiency video coding (HEVC) [1] has become gradually incapable of supporting high resolution in videos. The joint video exploration team (JVET) discussed and formulated the latest video compression standard called versatile video coding (VVC) from 2015, with the final version completed in July 2020 [2]. VVC can encode video content to the same level of visual quality while using about 50% fewer bits than HEVC. In addition to supporting 4 K or higher video compression, VVC also supports versatile applications, such as high dynamic range, screen content coding, aerial photography, and 360° videos [3–5], etc.

VVC adopts quadtree with nested multi-type tree (QT-MTT) coding structure [6]. A coding tree unit (CTU) is first partitioned by a quaternary tree, and then the quaternary tree leaf nodes can be further partitioned by a multi-type tree structure. Multi-type tree structure comprises four splitting types, including binary tree horizontal structure (BTH), binary tree vertical structure (BTV), ternary tree horizontal structure (TTH), and ternary tree vertical structure (TTV). BTH and BTV are grouped as BT, and TTH and TTV are grouped as TT. The union of BT and TT is called MTT. VVC removes the separation of the CU, prediction unit (PU), and transform unit (TU) coding process concepts used in HEVC. A CU can be either a square or rectangular shape, which allows the selection of CUs to be closer to the textures of figure content. The QT-MTT coding structure can thus achieve better predictive coding performance than only using the QT coding structure of HEVC [7]. Fig. 1 shows an example of a CTU partitioned into multiple CUs using MTT coding structure [8]. The black lines represent QT in which CUs are square shapes. The blue lines represent BT, and the red lines represent TT with rectangular-shaped CUs. QT-MTT provides a content-adaptive coding tree structure comprised of a CTU.

The encoding process of VVC intra-frames is a revision of HEVC, with the primary differences in procedures between HEVC and VVC described as follows [9]. HEVC adopts rough mode decision (RMD) and most probable modes (MPM), but VVC divides RMD into RMD-1 and RMD-2 because VVC has 67 intra modes. RMD-1 evaluates 33 directional (angular) intra modes plus DC and planar, just as HEVC does. These 35 intra modes are calculated by the sum of absolute transformed differences (SATDs), a simpler calculation method, to select MPM candidates. RMD-2 performs a refinement step to evaluate the SATDs of angular modes adjacent to the angular modes selected by RMD-1. In addition, VVC adopts other new coding tools to improve coding efficiency, such as multiple reference line (MRL), matrix-based intra prediction (MIP) and intra subpartition (ISP) [10–12]. Table 1 lists the coding performance of

☆ This paper has been recommended for acceptance by Zicheng Liu.

* Corresponding author.
E-mail address: fang@mail.mcu.edu.tw (J.-T. Fang).

https://doi.org/10.1016/j.jvcir.2022.103542
Received 30 August 2021; Received in revised form 12 March 2022; Accepted 14 May 2022
Available online 3 June 2022

all intra configurations between VTM 7.0, the reference software of VVC, and HM 16.20, the reference software of HEVC [13], and shows that the average encoding time of VVC intra coding is about 27 times that of HEVC. Thus, developing some faster algorithms to balance the coding quality and the time-consuming coding process has become an important issue.

Fast algorithms for VVC intra coding based on the feature analysis of a CU are summarized as follows. Fan. et al. [14], and Chen et al. [15] proposed fast algorithms for intra-frame prediction by applying the variance or gradient on a 32 × 32 CU. The variance of pixels was used to determine the smoothness of a CU, while gradient measurement was applied by Sobel filters. Horizontal gradient and vertical gradient determined the horizontal split or vertical split of a CU. The result of feature analysis was applied for QT early termination or QT early split to skip the rate distortion optimization (RDO) process. In [16], the authors selected five main intra modes, including angles 2, 18, 34, 50, and 66. By calculating their SATD values, the BT partition or TT partition could be predicted. The split (shape) between a CU and its sub-CUs were analyzed [17]. If the parent CU was a horizontal (or vertical) split, then there was a higher chance that this CU would be horizontal (or vertical). The conditional probability of splitting was calculated by Bayesian theorem. In [18], the authors analyzed pixel differences between the current CU and neighboring CUs. The gradients were measured in 4 directions, including the horizontal, vertical, 45°, and 135°. They proposed that if there were not many gradient differences between these four directions, the texture of this CU could be smooth, and it was not to be partitioned. By contrast, if the gradient in one direction was much larger than the other 3 directions, this CU tended to be partitioned.

From the work related to VVC intra coding, feature analysis such as variances or gradients of a CU block can be used for partition decisions. If the variance is small, indicating that the texture pattern is smooth, this CU is likely not to be partitioned. In contrast, if the variance is large, this CU is likely to be partitioned.

Deep learning is a new area of machine learning, and it is also a rapidly growing field in artificial intelligence. Artificial intelligence tries to mimic the mechanism of the human brain by training from the data and extracting significant information. This work uses convolutional neural networks (CNNs) to extract features during intra prediction. The basic structure of CNNs consists of convolutional layers, pooling layers, and fully connective layers. By the operations of convolution and downsampling, some significant characteristics of the input data can be extracted [19–20].

Table 1
Comparative performance of VTM 7.0 and HM 16.20 [13].

	All Intra Over HM16.20				
	Y	U	V	EncT	DecT
Class A1	-28,27%	-33,55%	-33,95%	1635%	173%
Class A2	-27,97%	-20,66%	-12,93%	2664%	182%
Class B	-21,27%	-20,04%	-27,35%	2916%	184%
Class C	-22,00%	-19,82%	-23,88%	4102%	184%
Class E	-25,40%	-21,93%	-26,55%	2364%	166%
Overall	-24,40%	-22,66%	-25,14%	2717%	179%
Class D	-17,82%	-13,97%	-15,50%	4532%	187%
Class F	-38,93%	-39,38%	-41,87%	4825%	180%

Fast algorithms for VVC intra coding based on deep learning are summarized as follows. In [21], the authors applied ResNets [22] of CNNs. The input was a 65 × 65 pixel block, which was a 64 × 64 CU plus one additional line on the left and top of the CU. After convolution and pooling, the output was a vector that gave probabilities of 4 × 4 boundaries of the block. This probability vector was further exploited by the encoder to determine whether or not this CU was required to be split. An adaptive CU split decision was proposed for intra frame with the pooling-variable CNNs [23]. In particular, they proposed shape-adaptive CNNs by using a pooling method to allow the input CU with square shapes to retain the original information of this CU. With the shape-based CNNs training scheme, various training sample sizes of CUs could be processed on the same CNNs. In [24], the authors proposed a deep learning approach to predict the CU partition for reducing the HEVC complexity at both intra-and inter-modes, which was based on CNNs and long- and short-term memory networks. They also established a large-scale database [25] for training, and an early-terminated hierarchical CNN that can learn to predict from the CU partition map. Consequently, the encoding complexity of intra-mode HEVC could be drastically reduced. CNNs were used to predict the quadtree with binary tree (QTBT) partition depth range of 32 × 32 blocks directly according to the inherent texture richness of the image [26]. For training optimization, they applied a misclassification penalty term combined with L2

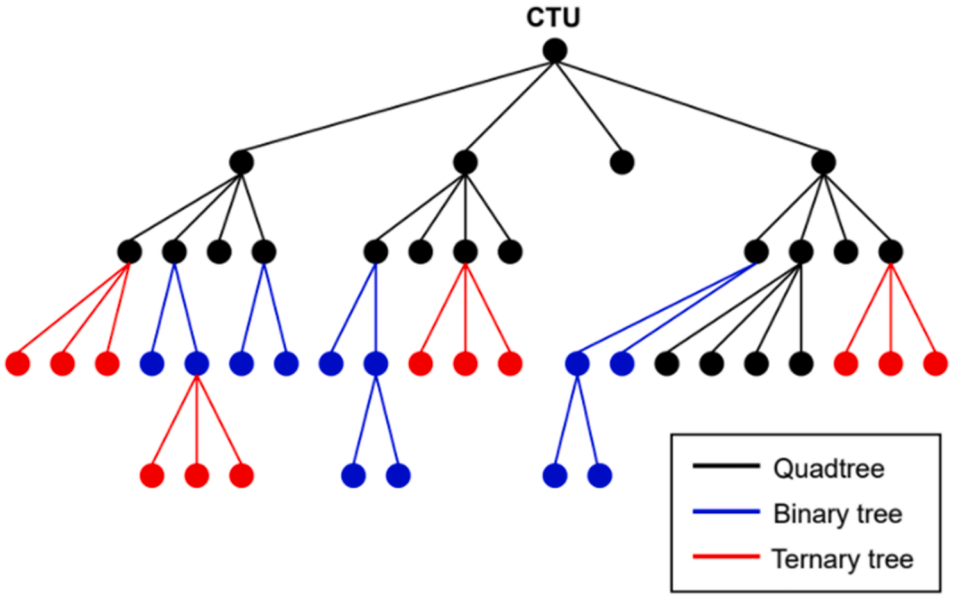

Fig. 1. An example of MTT coding structure [8].

HingeLoss function for regulation.

This work aims to reduce the encoding time of VVC intra coding and preserve high quality of video coding. Fast algorithms are proposed to determine the vertical split or horizontal split of 32 × 32 CUs from BT or TT partition. The proposed methods can be separated into two steps, the first step of which is the feature analysis method. By using feature map conversion and calculating variances, most 32 × 32 CUs can be classified with little coding gain lost. Those undermined CUs are classified by CNNs. Feature analysis method can be treated as a preprocessing step to the CNNs step. The preprocessing has two advantages. First, it can reduce the load of CNNs for training. Second, it can reduce the coding performance lost from CNNs.

This paper is organized as follows. Section II describes the proposed fast algorithms. Experimental results are described in Section III. Finally, Section IV presents the conclusion.

2. Fast algorithms for intra coding

Proposed fast algorithms are a two-step method to reduce the encoding time of VVC intra coding. Fig. 2 plots the difference between the original VVC and the proposed method. The procedure of VTM processing MTT is plotted on the right-hand side, marked by a yellow color zone. It processes the BTH, BTV, TTH, and TTV sequentially, and then determines the best one. By contrast, the procedure of the proposed method is plotted on the left-hand side, marked by a blue color zone. Fast algorithms determine the horizontal and vertical split for BT, and TT, sequentially. If a splitting condition satisfies the proposed criteria, then this split is selected as the candidate without further comparison.

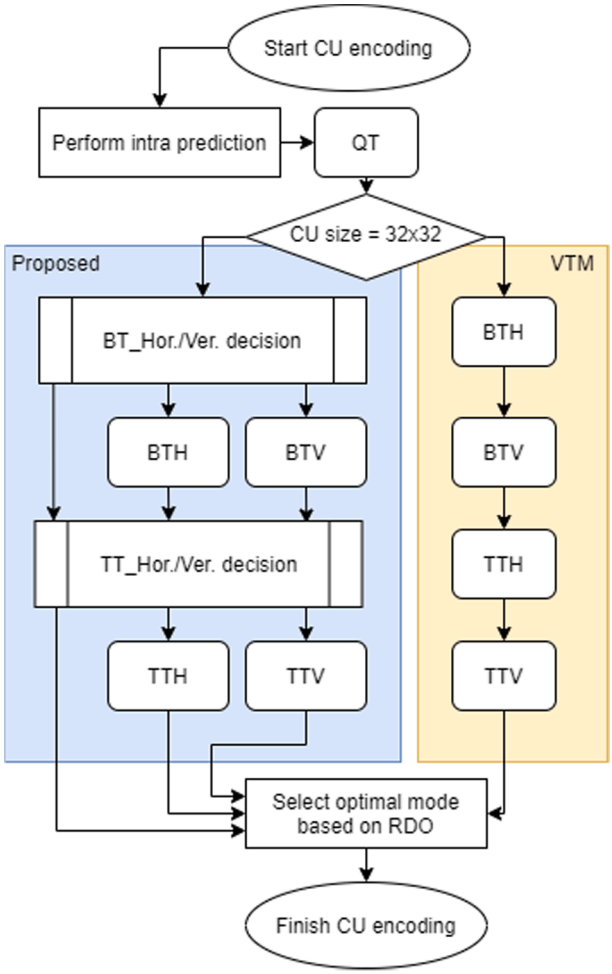

Fig. 2. Comparison of procedures between VTM and proposed method.

In this work, we applies fast algorithms to determine the splits of 32 × 32 CU blocks. This is because if there is an error in determination of splits for 64 × 64 CU blocks, a huger loss may occur in the encoding performance, and the splits for 16 × 16 CU blocks do not occupy much encoding time. Since the transmission bit rate increases little, the decoding time is almost the same as the original VTM. This work focuses on the reduction of encoding time, and the early termination of CU split decision is the main contribution of the proposed method.

The proposed fast algorithms can be separated into two steps. The first step is the feature analysis method, and the second step is to apply CNNs for classification. The following two sub-sections describe the proposed method in detail.

A. Feature Analysis of CUs

From the analysis of related work, variances of pixels in a CU can be applied to measure texture complexity [14–15]. In this work, we use variances of pixels to determine horizontal or vertical split of BT or TT partitions, respectively. To reduce heavy calculations and to avoid the influence from small region changes, this work applies feature map conversion. Each 32 × 32 CU is divided into 16 8 × 8 pixel blocks. The variance of each 8 × 8 pixels is calculated as a new unit, called a feature map. The variance of each feature map can be calculated by (1),

$$FM_{var}(i,j) = \frac{1}{64}\sum_{k=0}^{7}\sum_{l=0}^{7}(p_{8i+k,8j+l} - \mu_{i,j})^2 \quad (1)$$

where (i,j) is the coordinate of the feature map, $p_{i,j}$ is the pixel value at (i, j), and $\mu_{i,j} = \frac{1}{64}\sum_{k=0}^{7}\sum_{l=0}^{7}p_{8i+k,8j+l}$ is the mean value at (i,j). (1) simplifies the calculation of variance; in particular, the feature map can be fitted to any CU size for variance calculation because it is a common divisor of any size of CU.

To determine vertical or horizontal split, BTH, BTV, TTH, or TTV use different blocks of variances for measurement. Fig. 3 plots variance calculations for BTH, BTV, TTH, or TTV, respectively. For BTH, it requires the Var_{01}^{BTH} and Var_{02}^{BTH}. For BTV, it the requires Var_{01}^{BTV} and Var_{02}^{BTV}. For TTH, it requires Var_{01}^{TTH}, Var_{02}^{TTH}, and Var_{03}^{TTH}. And for TTV, it requires Var_{01}^{TTV}, Var_{02}^{TTV}, and Var_{03}^{TTV}.

Variance calculations for BTH, BTV, TTH, or TTV splits are based on the pixel blocks which partition a 32 ×32 CU. Pixel blocks are plotted by the green lines for these four splits, respectively, as shown in Fig. 4. Formulas for BTH variance calculations based on a feature map are expressed by (2) and (3),

$$Var_{01}^{BTH} = \frac{1}{2\bullet 4}\sum_{i=0}^{1}\sum_{j=0}^{3}\left[FM_{\text{var}}(i,j) - \frac{1}{2\bullet 4}\sum_{k=0}^{1}\sum_{l=0}^{3}FM_{\text{var}}(k,l)\right]^2 \quad (2)$$

$$Var_{02}^{BTH} = \frac{1}{2\bullet 4}\sum_{i=2}^{3}\sum_{j=0}^{3}\left[FM_{\text{var}}(i,j) - \frac{1}{2\bullet 4}\sum_{k=2}^{3}\sum_{l=0}^{3}FM_{\text{var}}(k,l)\right]^2 \quad (3)$$

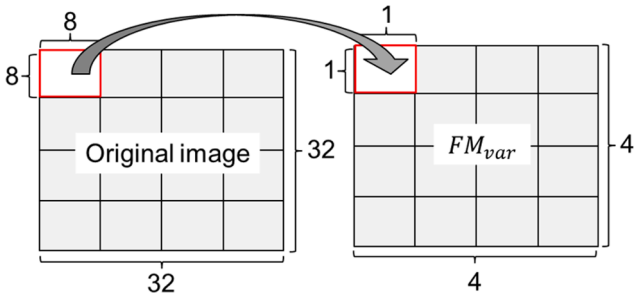

Fig. 3. Variance calculations based on feature map.

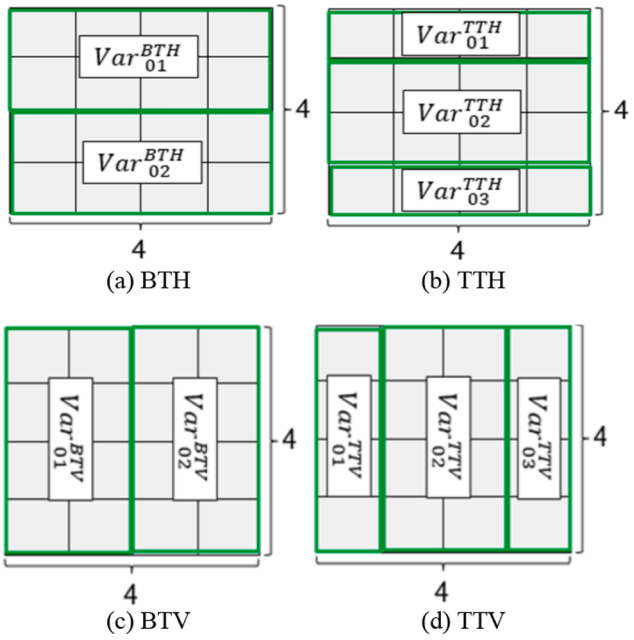

Fig. 4. Examples of variance calculations for classification.

where Var_{01}^{BTH} is for the upper sub-CU and Var_{02}^{BTH} is for the lower sub-CU. Formulas for TTH variance calculations based on a feature map are expressed by (4), (5), and (6),

$$Var_{01}^{TTH} = \frac{1}{4}\sum_{j=0}^{3}\left[FM_{var}(0,j) - \frac{1}{4}\sum_{l=0}^{3}FM_{var}(0,l)\right]^2, \tag{4}$$

$$Var_{02}^{TTH} = \frac{1}{2\bullet 4}\sum_{i=1}^{2}\sum_{j=0}^{3}\left[FM_{var}(i,j) - \frac{1}{2\bullet 4}\sum_{k=1}^{2}\sum_{l=0}^{3}FM_{var}(k,l)\right]^2, \tag{5}$$

$$Var_{03}^{TTH} = \frac{1}{4}\sum_{j=0}^{3}\left[FM_{var}(3,j) - \frac{1}{4}\sum_{l=0}^{3}FM_{var}(3,l)\right]^2. \tag{6}$$

Variance calculations based on feature maps for Var^{BTV} and Var^{TTV} can be derived based on Fig. 3 (c) and (d). Formula of Var_{01}^{BTV}, Var_{02}^{BTV}, Var_{01}^{TTV}, Var_{02}^{TTV}, and Var_{03}^{TTV} are similar to formula of (2), (3), (4), (5), and (6), respectively. Fig. 4 plots examples of variance calculations based on a feature map where different colors represent different variance values. Take BTH in Fig. 4(a) as an example. The variance difference between Var_{01}^{BTH} and Var_{02}^{BTH} is large, and this CU has a high chance to be selected as BTH partition. In other words, if the upper (or lower) half block has smooth content, but the other half block contains complex content, then the variance difference between Var_{01}^{BTH} and Var_{02}^{BTH} is large, and there is a high chance that this CU selects BTH partition. Similar deduction of feature analysis can be applied to other partitions.

For building models to determine vertical or horizontal split of BT partition, this work used sequences 768 × 512 and 2880 × 1920 from CPIH database [27]. Frames were encoded by VVC intra coding under four kinds of quantization parameters (QPs) setting, i.e. 22, 27, 32, and

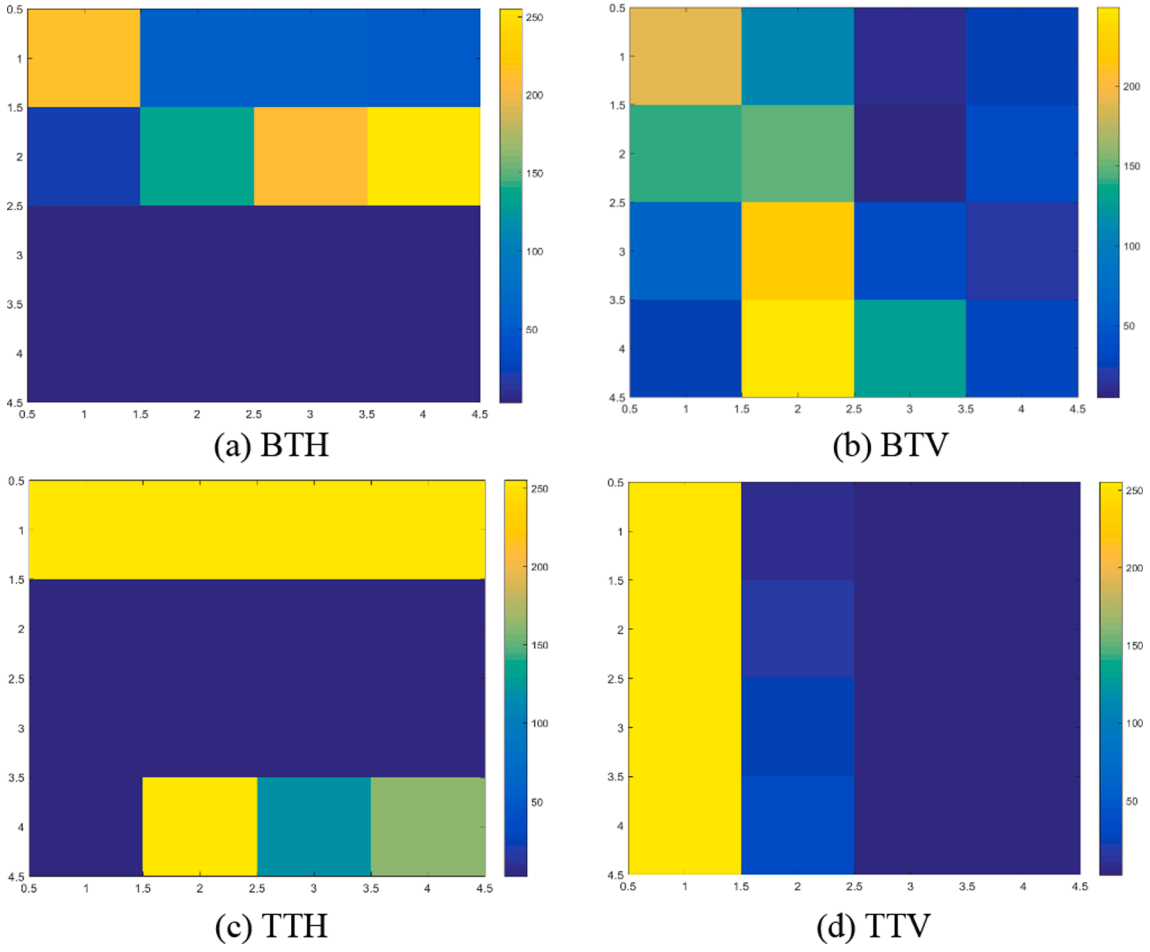

Fig. 5. Variance distributions of BT.

37, separately. CUs with 32 × 32 pixels determined by RDO to be BTH or BTV were selected. Approximately 1 $\times 10^5$ image blocks were selected, and these images were almost equally distributed between vertical or horizontal split.

To build a relationship between Var_{01}^{BTH} and Var_{02}^{BTH} and BTH (or BTV), data of Var_{01}^{BTH} and Var_{02}^{BTH} are treated as a point in a two-dimensional Cartesian coordinate. Fig. 5(a) plots Var_{01}^{BTH} and Var_{02}^{BTH}distributions of BTH and BTV under QP 22, respectively, where the x-axis is Var_{01}^{BTH} calculated by (2), and the y-axis is Var_{02}^{BTH} calculated by (3). Red triangle points represent BTH, and blue star points represent BTV. Fig. 5(a) shows that if CUs are finally determined by BTH, their Var_{01}^{BTH} and Var_{02}^{BTH} distributions are near the x-axis or y-axis. This result is consistent with Fig. 4(a) of BTH. By contrast, if CUs are finally determined by BTV, their Var_{01}^{BTH} and Var_{02}^{BTH} distributions are near 45^0 between x-axis and y-axis. This result is consistent with Fig. 4(b) of BTV.

Fig. 5(b) shows Var_{01}^{BTV} and Var_{02}^{BTV} distributions of BTH and BTV under QP 22, respectively, where the x-axis is Var_{01}^{BTV}, and the y-axis is Var_{02}^{BTV}. Red triangle points (BTH) are near 45^0 between x-axis and y-axis, and blue star points (BTV) are near the x-axis or y-axis. Var_{01}^{BTV}, while Var_{02}^{BTV} distributions are in opposite positions, compared to Var_{01}^{BTH} and Var_{02}^{BTH} distributions.

In Fig. 5(a), dashed red lines are plotted to separate the red triangle points and blue star points. Mathematically, a parabolic equation is used to separate vertical split and horizontal split. The parabolic equation is derived by (7),

$$f_{BT}(x, y) = a(x-b)^2 - (y-b) \tag{7}$$

where a is the weight, and b is the bias from the origin. In Fig. 5(a), most red triangle points are located near the x-axis or y-axis, outside the dashed red line. By contrast, most blue star points are located inside the dashed red line. In other words, if data of Var_{01}^{BTH} and Var_{02}^{BTH} in (7) are less than zero, then this CU is classified as BTH. Otherwise, the split is classified as BTV. From the experiment data, the bias b is set at 1 $\times 10^5$. The weight a can determine the curvature of the parabola. To reduce the error decision, a is set at 1 $\times 10^{-5}$

Fig. 5(b) has opposite distributions to Fig. 5(a). In Fig. 5(b), most blue star points are located near the x-axis or y-axis, outside the dashed red line. By contrast, most red triangle points are located inside the dashed red line. For simplification, we use the same equation and parameters in (7) for classification. If data of Var_{01}^{BTV} and Var_{02}^{BTV} in (7) are less than zero, then this CU is classified as BTV. Otherwise, the split is classified as BTH.

Similar methods are used to build models for TT classification. Sequences from CPIH database [27] were adopted. About 1 $\times 10^5$ CUs finally determined by RDO to be vertical or horizontal split from TT partition were selected. Fig. 6(a) plots Var_{01}^{TTH}, Var_{02}^{TTH}, and Var_{03}^{TTH} distributions for TTH and TTV, respectively. A three-dimensional parabolic equation to separate TTH and TTV can be expressed by (8),

$$f_{TT}(x, y, z) = \alpha\left((x-\beta)^2 + (y-\beta)^2\right) - (z-\beta) \tag{8}$$

where α is the weight, and β is the bias from the origin.

In Fig. 6(a), most red triangle points are located near the x-axis, y-axis, or z-axis. These points are outside the dashed red line. By contrast, most blue star points are located inside the dashed red lines. This means that if data of Var_{01}^{TTH}, Var_{02}^{TTH}, and Var_{03}^{TTH} in (8) are less than zero, then the CU is classified as TTH. Otherwise, it classified as TTV. Distributions in Fig. 6(b) have the opposite distributions to Fig. 6(a). For simplification, we use the same equation and parameters in (8). In Fig. 6(b), most blue star points are located near the x-axis, y-axis, or z-axis. These points are outside the dashed red line. By contrast, most red triangle points are located inside the dashed red line. If value of Var_{01}^{TTV}, Var_{02}^{TTV}, and Var_{03}^{TTV} in (8) are less than zero, then this CU is classified as TTV; otherwise it is classified as TTH. From the experimental data, the bias β is set at 1 $\times 10^5$, and the weight α is set at 1 $\times 10^{-5}$.

The proposed feature analysis method is used to determine horizontal or vertical split for BT and TT partition, respectively. Data of Var_{01}^{BTH} and Var_{02}^{BTH}, as well as Var_{01}^{BTV} and Var_{02}^{BTV} in (7) are for BTH and BTV split determination. Using (7) for BT split determination, three conditions may result. The first condition is that both equations are less than zero. This condition represents that variances in both x-axis and y-axis are small; this CU is located near the original point as shown in Figs. 5 and 6. It means that this CU has a low likelihood of choosing BT partition and can skip horizontal and vertical split decision by RDO. The second condition is that only one direction (either vertical or horizontal) has high variance value. This condition represents that this CU is located near the x-axis or y-axis as shown in Figs. 5 and 6. It means that this CU has a high chance to be determined as requiring vertical (or horizontal) split. The third condition is that variances in both directions are high. This CU is located inside the dashed red line as shown in Figs. 6 and 7. Under this condition, the proposed feature analysis method fails to determine this CU partition, and this CU split decision has to use CNNs for classification.

Data of Var_{01}^{TTH}, Var_{02}^{TTH}, and Var_{03}^{TTH}, as well as of Var_{01}^{TTV}, Var_{02}^{TTV}, and Var_{03}^{TTV}, in (8) are for TTH and TTV split determination. Three conditions may occur in TT classification by using (8) for determination. Similar

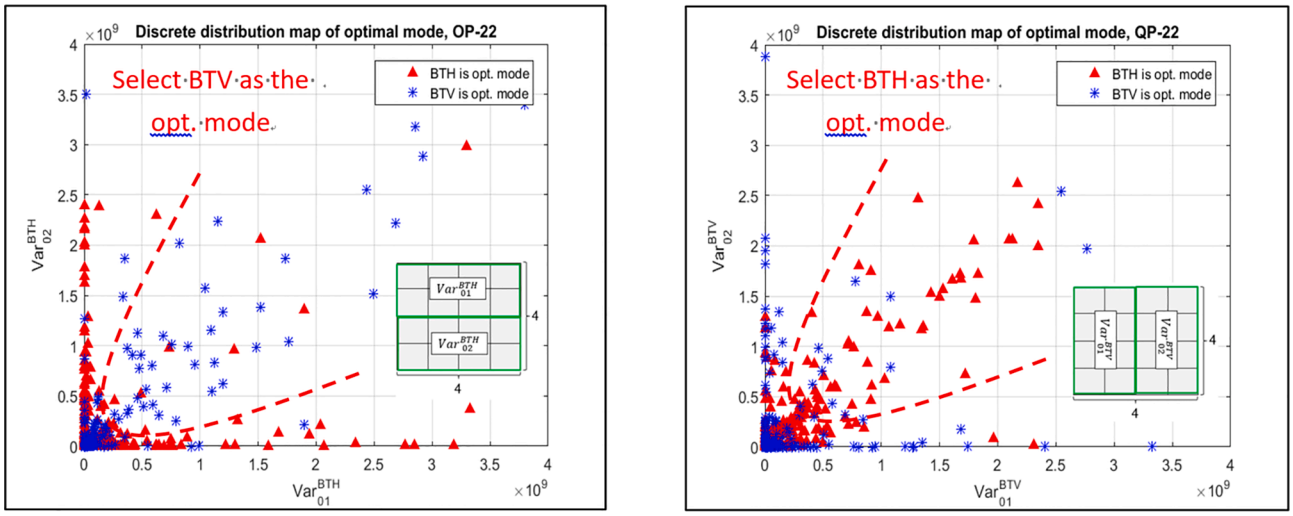

Fig. 6. Variance distributions of TT.

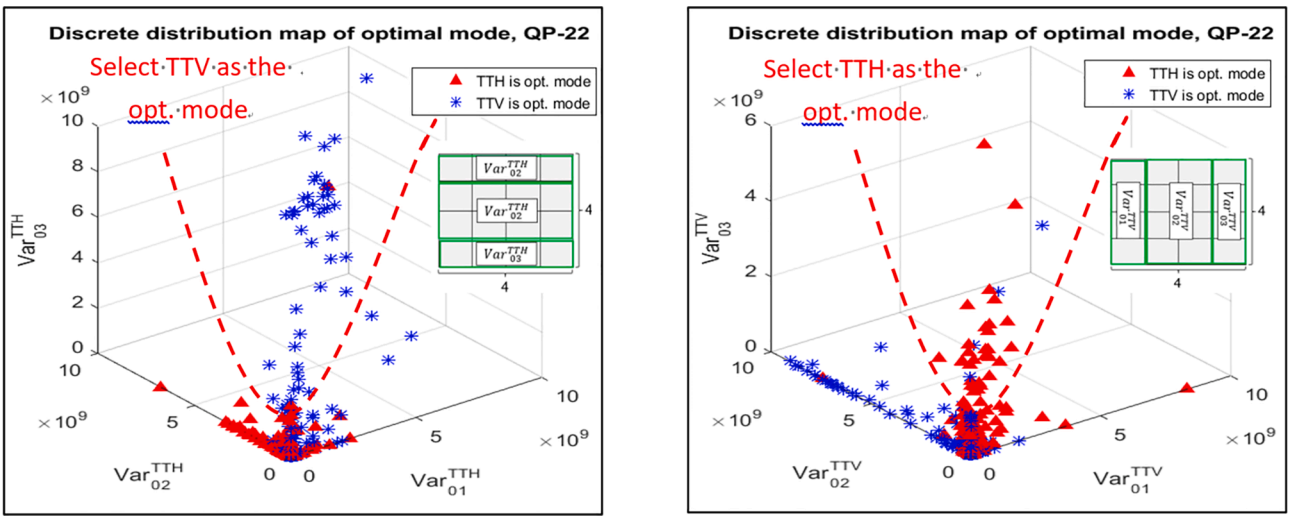

Fig. 7. Flow chart of the feature analysis method.

analysis procedure to BT partition can be applied for TT partition.

Fig. 7 plots the flow chart for determination of vertical and horizontal splits for MTT (either BT or TT) coding structure based on the proposed feature analysis method. In particular, from our later experiment results, most 32 × 32 CUs are under conditions (1) and (2). The small amount of CUs under condition (3) used CNNs for classification, and this can reduce the input loading of CNNs.

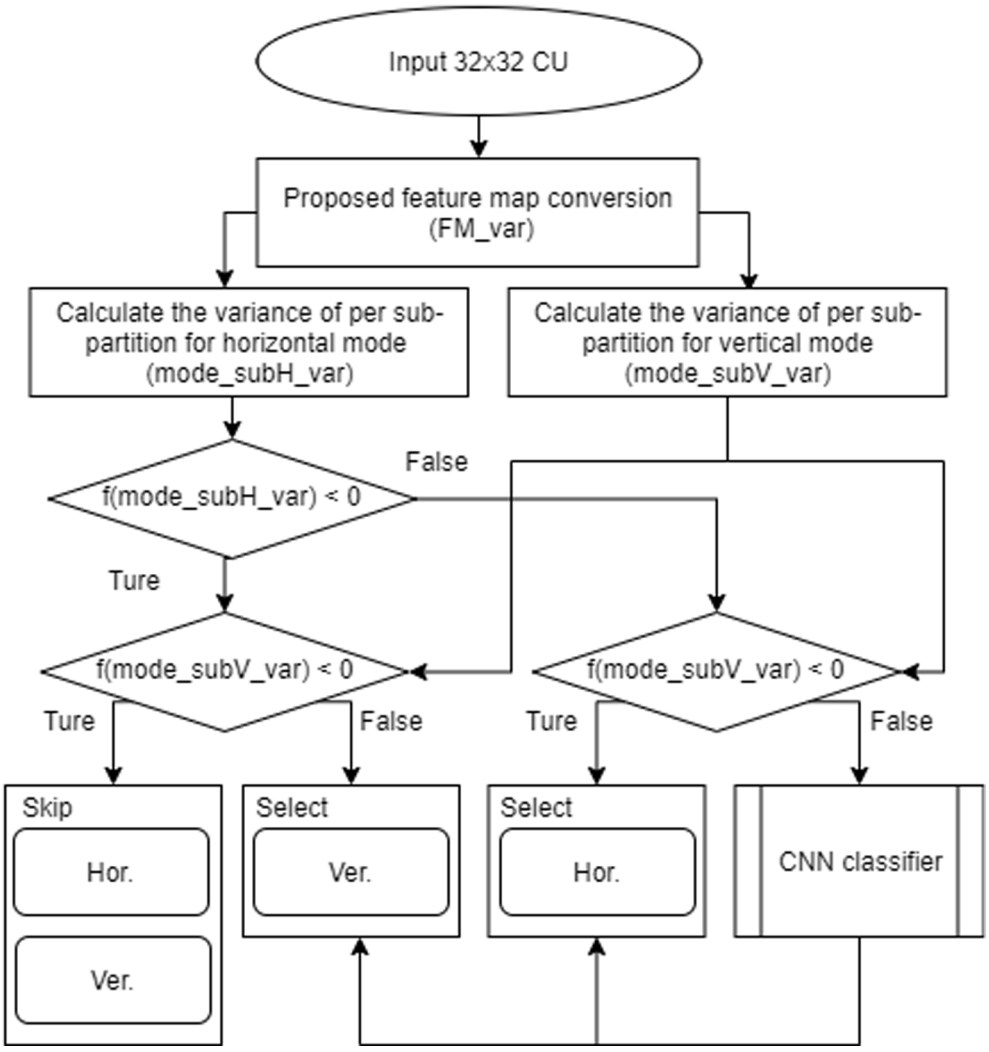

Fig. 8. Proposed CNNs structure.

B. 4 Convolution neural networks for classification

CNNs are used to determine those undetermined MTT (either BT or TT) partitions as discussed in the previous subsection. For each QP, image content for prediction coding may be changed. The prediction coding may be changed under different QP setting; so, four different CNNs models are required under different QPs (22, 27, 32, and 37). Training data are from CPIH database [27]. To reduce the correlation of labels in CNNs classification, 32 × 32 CU image blocks at increased intervals are selected for training. Thus, more test sequences are selected for labeling, including sequences with sizes of 768 × 512, 1536 × 1024, 2880 × 1920, and 4928 × 3264 from CPIH database. These sequences are first encoded by VVC intra coding. Then, from the encoded result, those CUs with 32 × 32 pixels, and determined by RDO to be MTT partitions, are extracted. These extracted 32 × 32 CUs are then transformed into 4 × 4 feature maps as input data to CNNs. In other words, image data are classified into two categories, i.e. BTH, and BTV, (or TTH, and TTV). This work adopts supervised learning, and two categories of images data are used for CNNs labeling. After some experience with training the CNNs model, the following procedure was found to improve the accuracy of CNNs performance. That is, BT image data were first used to train the neural networks model, and then TT image data were used to update the trained neural networks model. The same CNNs model are used for BT and TT split decisions. CNNs classify the vertical or horizontal split from MTT partition of a 32 × 32 CU. After classification, the split result is sent back to VTM, and VTM continues the remaining procedures of intra coding.

Proposed CNNs structure is plotted in Fig. 8, including one convolution layer, four dense block layers, and one classification layer. The lost function is cross entropy with L_2 regulation, expressed by (9),

$$L_{y'}(y) = \frac{1}{n}\left[-\sum_{i=1}^{n} y_i' \log(y_i) \right] + \alpha \sum_{w} w^2 \tag{9}$$

where y' is the prediction probability, expressed by $y'_i(x) = softmax(x)_i = \frac{\exp(x_i)}{j\sum \exp(x_j)}$. L_2 is regularization, where w is the kernel, and α is the parameter of weight decay.

Input data to CNNs are a matrix with dimension 4 × 4, composed of 16 4 × 4 feature maps. The feature map is a 8 × 8 pixels conversion, which can reduce heavy calculations by CNNs [29].Sixteen 4 × 4 feature maps equals 32 × 32 pixels of a CU. The first convolution layer of CNNs is applied as a 3 ×3 kernel with stride 1. After the first layer of convolution, 16 4 × 4 feature maps can be transformed into 32 4× 4 output. The remaining 4 convolution layers and dense blocks refer to [28]. Using dense blocks can avoid the vanishing gradient problem. Batch normalization is achieved by re-centering and re-scaling to make the neural networks faster and more stable through normalization of the layers' inputs [30].

Fig. 8 shows that the dense net connects each layer to every other layer in a feed-forward fashion. For each layer, the feature maps of all preceding layers are used as inputs, and its own feature map is used as input into all subsequent layers [28]. After 4 dense layers, average pooling is used to remove noise points. Finally, 64 dimensions of full connections are employed for classification and the final classification result is sent to the output layer.

The main hyper-parameters of proposed CNNs are listed in Table 2.

Table 2
Hyper-parameters of proposed CNNs.

Batch size	64
Learning rate	0.0001
Epochs	60
Dropout	0.75
Weight decay	0.005

Note that the dropout is used to avoid overfitting in the training process. To increase the accuracy of validating, the dropout is omitted in the validating process.

3. Experimental results

Experiments are designed to demonstrate the performance of the proposed algorithms. Experiment environment is listed as follows. For training, the CPU is i7-9700@3.6 GHz, the RAM is 64GM, the GPU is RTX 2080, and the OS is Ubuntu-X64 18.04. For validating, the CPU is i7-6700 @3.4 GHz, the RAM is 24GM, and the OS is Windows 10-X64. The software of CNNs is Python 3.6 and the neural network took is Tensorflow 1.12.0. Reference software is VTM version 7.0 [31], and all intra configuration is used. Test sequences refer to common to test conditions (CTC) of VVC [32].

Table 3 lists the accuracy of CNNs under different QP settings. Training data are approximately 1 $\times 10^5$ image blocks, and validation data are approximately 1 $\times 10^4$ image blocks. The training procedure is separated into two steps. The first step is to train a CNNs model for BT partition, and the second step is using the already trained CNNs model to build another CNNs model for TT partition, thus the two-step procedure to build CNNs models is more achievable. More 32 × 32 CU image blocks are used for labeling in BT partition than labeling in TT partition because the former requires more data to start a new CNNs model. About 60% of image blocks are for BT partition training, and about 40% for TT partition training. For BT/TT partition, the number of image blocks used for labeling the horizontal split or vertical split are almost equal. Table 3 demonstrates the ratio between the training data and validating data is about 10:1 because the two-step training process is applied. Average accuracy from four CNNs models is 72.2%. In other words, 28.8% of classifications by CNNs may be wrong. The dropout procedure is not applied in the validating process, and this may result in the validating performance having higher accuracy than the training performance.

Table 4 lists performances of the feature analysis method. About 70% (from 59% to 78%) of 32 × 32 CUs use the feature analysis method. In other words, only about 30% of CUs use CNNs for classification, which greatly reduces loading, and also reduces the impact on video quality drop by misclassification of CNNs. Table 4 indicates that as QP setting decreases, the percentage of using feature analysis method increases. This is because more details of image content can be preserved for lower values of QPs, and the variance differences between sub-CU blocks are more significant. Images using feature analysis method at QP 37 are 1.27 times of images at QP 22.

Table 4 also lists bit rate difference and PSNR difference compared to original VVC performance. Average bit rate increases about 0.5% and average PSNR drops about 0.02% by using the feature analysis method alone. The percentages of BT and TT in Table 4 show the percentages of 32 × 32 CUs that applied the feature analysis method for classification, respectively. This performance is good because about 70% of BT/TT applied the feature analysis method, and about 30% of BT/TT applied the CNNs for horizontal/vertical split determination. Furthermore, as QP increases, the number of CUs using the feature analysis method increases. So, the bit rate increases and the PSNR performance drops more.

Performance of proposed methods and performance comparisons with state-of-the-art works [14] are listed in Table 5. The Bjøntegaard delta bit rate (BDBR) is calculated based on [33]. Reduction of coding

Table 3
Performance of CNNs.

QP	Number of data		Accuracy (%)	
	Training	Validating	Training	Validating
37	103,164	10,000	70.1	71.4
32	100,782	10,000	69.8	72.7
27	93,998	10,000	69.6	72.8
22	88,018	10,000	68.6	71.7

Table 4
Performance by feature analysis method.

Class	Sequence	QP	Proportion of BT (%)	Proportion of TT (%)	Δ Biterate (%)	Δ $PSNR_{yuv}$ (%)
B	BQTerrace	22	15.99	16.36	0.06	<0.01
		27	32.48	33.98	0.36	<0.01
		32	37.02	39.99	0.54	−0.02
		37	39.39	44.26	0.96	−0.04
C	PartyScene	22	6.03	6.48	0.06	<0.01
		27	9.06	10.12	0.09	<0.01
		32	14.92	16.84	0.21	0.01
		37	26.65	31.23	0.57	0.01
	RaceHorsesC	22	40.74	42.02	0.30	0.01
		27	43.42	44.68	0.23	−0.01
		32	45.23	47.26	0.39	−0.01
		37	49.38	52.74	0.69	−0.01
D	BlowingBubbles	22	7.18	7.33	−0.04	−0.01
		27	12.82	12.97	0.10	0.01
		32	25.27	25.71	0.30	0.01
		37	45.86	47.84	0.99	0.03
	RaceHorses	22	18.68	19.56	0.09	<0.01
		27	21.68	22.49	0.15	0.01
		32	33.41	34.80	0.42	0.01
		37	46.52	49.52	0.48	−0.03
E	Johnny	22	27.33	30.26	0.63	−0.01
		27	24.55	27.98	0.73	−0.02
		32	24.70	28.91	1.32	−0.04
		37	24.84	29.57	1.26	−0.13
	KristenAndSara	22	39.55	41.17	0.62	−0.01
		27	36.36	38.29	0.77	−0.03
		32	33.87	36.10	0.73	−0.07
		37	29.39	31.91	0.77	−0.14
All class average		22	22.21	23.31	0.25	−0.01
		27	25.77	27.22	0.35	−0.01
		32	30.63	32.80	0.56	−0.02
		37	37.43	41.01	0.82	−0.04

Table 5
System performance and comparison.

Class	**Sequence**	[7] **(QTMT/ VTM-7.0)**			**Proposed algorithm (QTMT/ VTM-7.0)**					
					CNN off			**Overall**		
		BDBR (%)	ΔT (%)	TS/BDBR	BDBR (%)	ΔT (%)	TS/BDBR	BDBR (%)	ΔT (%)	TS/BDBR
B	BQTerrace	1.08	45.30	41.94	0.08	8.60	**111.76**	0.58	30.24	52.58
	Cactus	1.84	52.44	28.50	0.08	6.68	**81.78**	0.81	30.11	37.24
	BasketballDrive	3.28	59.35	18.09	0.06	7.61	**128.17**	1.79	33.78	18.89
C	BasketballDrill	1.82	48.48	26.64	0.21	10.32	**65.42**	0.92	29.93	32.49
	BQMall	1.87	52.47	28.06	0.07	15.73	**212.57**	1.09	32.63	29.94
	PartyScene	0.26	38.62	148.54	0.06	13.68	**234.33**	0.22	25.55	117.55
	RaceHorsesC	0.88	49.05	55.74	0.04	11.54	**262.27**	0.45	31.64	70.31
D	BasketballPass	1.95	47.70	24.46	0.02	10.32	**443.68**	1.13	29.19	25.90
	BQSquare	0.19	31.95	168.16	< 0.01	7.62	**> 999.99**	0.08	19.95	264.82
	BlowingBubbles	0.19	40.35	85.85	0.02	6.10	**305.00**	0.23	23.93	119.96
	RaceHorses	0.54	41.69	77.20	0.02	13.31	**605.00**	0.28	30.95	111.35
E	FourPeople	2.70	57.57	21.32	0.11	12.28	**116.43**	1.27	28.83	22.23
	Johnny	3.22	56.88	17.66	0.05	8.59	**162.64**	1.53	31.88	22.22
	KristenAndSara	2.78	55.11	19.82	0.06	7.74	**138.87**	1.29	26.50	20.41
All class average		1.61	48.35	29.95	0.07	10.01	**147.85**	0.83	28.94	34.71

time by the proposed method is calculated by (10),

$$\Delta T = \frac{1}{4}\sum_{QP}\frac{T_R(QP) - T_C(QP)}{T_R(QP)} \times 100\% \tag{10}$$

where T_R(QP) and T_C (QP) are coding time of VVC reference software and proposed method, respectively. Four QPs are 22, 27, 32, and 37. Table 5 shows that the proposed method can save about 28.94% of coding time and BDBR increases about 0.83%. The ratio of time saving per BDBR (TS/BDBR) is about 34.7. For feature analysis method alone, about 10% of coding time can be saved and BDBR increases about 0.07. In other words, the feature analysis method results in little video quality loss. Analysis of BDBR performance of each test sequence, in general, shows that sequences with complex content, such as Class_B BQTerrace, Class_C PartyScene, and Class_D BlowingBubbles and RaceHorses, can maintain better video quality. The reason for this can be explained as follows. Table 5 shows that about 70% of 32 × 32 CUs use feature analysis method for split decision, and this method can maintain high video quality. Most 32 × 32 CUs with complex content are from these sequences and use feature analysis method, so, their BDBR performance drops less.

Fan. et al. used the variance of variance to determine the best choice among 5 coding structures (QT, BTH, BTV, TTV, and TTH) [14]. By contrast, this work proposed a two-step algorithm to determine the BTH/BTV or TTH/TTV, which combined statistical feature analysis methods and the CNN classification methods. Both methods are applied in all intra configuration of the same version of VTM. Although the

method in Fan. et al. [14] can save more coding time, the video quality drops more compared with the proposed method. Table 5 shows that for most of the test sequences, the proposed method has better TS/BDBR performance than that by the method of [14].

4. Conclusion

This work proposes fast algorithms to determine MTT vertical or horizontal split decisions on 32 × 32 CUs from VVC intra coding. A two-step method is proposed. The first step is the feature analysis method. By using feature map conversion and calculating variances, most 32 × 32 of CUs can be classified with little coding gain lost. The remaining undetermined CUs are classified by CNNs, the second step in the method. From experiment results, about 70% of 32 × 32 CUs use the feature analysis method, which can maintain high video quality. The proposed algorithms take advantage of existing methods but combines them in a novel way. Improving the CNNs performance is a topic for future work, including the accuracy of classification, the compatibility with VTM, and the number of models.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] G.J. Sullivan, J.R. Ohm, W.J. Han, T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard, IEEE Trans. Circuits Syst. Video Technol. 22 (12) (2013) 1649–1668.

[2] G. Sullivan, Versatile Video Coding (VVC) Arrives, 2020 IEEE International Conference on Visual Communications and Image Processing (VCIP), pp. 1-1, Macau, China, 2020.

[3] A. Segall, E. François, D. Rusanovskyy, JVET common test conditions and evaluation procedures for HDR/WCG video, JVET-K1011, 11th JVET meeting: Ljubljana, SI, 10–18 July 2018.

[4] P. Hanhart, J. Boyce, K. Choi, JVET common test conditions and evaluation procedures for 360° video, JVET-K1012, 11th JVET meeting: Ljubljana, SI, 10–18 July 2018.

[5] M. Karczewicz, E. Alshina, JVET AHG report: Tool evaluation (AHG1), JVET-H0001, 8th. Meeting, China, Oct. 2017F.

[6] J. Chen, E. Alshina, Algorithm description for versatile video coding and test model (VTM1), JVET-J1002-v2, Apr. 2018.

[7] Z. Wang, S. Wang, J. Zhang, S. Ma, Probability decision based block partitioning for future video coding, IEEE Trans. Image Process 27 (3) (Mar. 2018) 1475–1486.

[8] J. Chen, Y. Ye, S.-H. Kim, Algorithm description for versatile video coding and test model (VTM11), JVET-T2002-v1, 20 Meeting, Oct. 2020.

[9] M. Saldanha, G. Sanchez, C. Marcon, L. Agostini, Complexity analysis of VVC intra coding, in Proc. IEEE Int. Conf. Image Processing (ICIP), United Arab Emirates, pp. 3119-23, Oct. 2020.

[10] S. De-Luźan-Herńandez, V. George, J. Ma, T. Nguyen, H. Schwarz, D. Marpe, T. Wiegan, An intra subpartition coding mode for VVC, in Proc. IEEE Int. Conf. Image Processing (ICIP), Taipei, 2019.

[11] L. Zhao, X. Zhao, S. Liu, X. Li, J. Lainema, G. Rath, F. Urban, F. Racapé, Wide Angular Intra Prediction for Versatile Video Coding, in Proc. IEEE Int. Conf. Image Processing (ICIP), Taipei, 2019.

[12] B. Bross, et. al., CE3: Multiple reference line intra prediction, JVET 12 Meeting, JVET-L0283, Macao, Oct. 2018.

[13] F. Bossen, X. Li, K. Suehring, AHG report: Test model software development (AHG3), JVET-Q0003-v1, 17th Meeting: Brussels, BE, 7–17 Jan. 2020.

[14] Y. Fan, J. Chen, H. Sun, J. Katto, M. Jing, A Fast QTMT Partition Decision Strategy for VVC Intra Prediction, IEEE Access 8 (2020) 107900–107911.

[15] J. Chen, H. Sun, J. Katto, M. Jing, Y. Fan, Fast QTMT Partition Decision algorithm in VVC Intra coding based on variance and gradient, in Proc. IEEE Int. Conf. Image Processing (ICIP), Taipei, Sep. 2019.

[16] M. Lei, F. Luo, X. Zhang, S. Wang, S. Ma, Look-ahead prediction based coding unit size pruning for VVC intra coding, in Proc. IEEE Int. Conf. Image Processing (ICIP), Taipei, pp. 4120-24, Sep. 2019.

[17] T. Fu, H. Zhang, F. Mu, H. Chen, Fast CU partitioning algorithm for H.266/VVC intra-frame coding, in Proc. IEEE Int. Conf. Multimedia and Expo (ICME), Shanghai, pp. 55-60, Jul. 2019.

[18] J. Chi, T. Zhang, C. Gu, X. Zhang, S. Ma, Gradient-based early termination of CU partition in VVC intra coding, in Data Compression Conference (DCC), virtual conference, pp. 103-112, Mar. 2020.

[19] A. Krizhevsky, I. Sutskever, G. Hinton, Imagenet classification with deep convolutional neural networks, January Adv. Neural Inform. Process. Syst., 25(2), Jan. 2012 (NIPS 2012) .

[20] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L. D. Jackel, Backpropagation applied to handwritten zip code recognition, Neural Comput. 1 (4) (1989) 541–551.

[21] A. Tissier, W. Hamidouche, J. Vanne, F. Dalpin, D. Menard, CNN oriented complexity reduction of VVC intra encoder, in Proc. IEEE Int. Conf. Image Processing (ICIP), United Arab Emirates, pp. 3139-43, Oct. 2020.

[22] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, Jun. 2016.

[23] G. Tang, M. Jing, X. Zeng, Y. Fan, Adaptive CU split decision with pooling-variable CNN for VVC intra encoding, IEEE Visual Communications and Image Processing (VCIP), Australia, Dec. 2019.

[24] M. Xu, T. Li, Z. Wang, X. Deng, R. Yang, Reducing Complexity of HEVC: A Deep Learning Approach, IEEE Trans. Image Process. 27 (10) (Oct. 2018) 5044–5059.

[25] D.-T. Dang-Nguyen, C. Pasquini, V. Conotter, G. Boato, RAISE: A raw images dataset for digital image forensics, in: Proc. 6th ACM Multimedia Syst. Conf, 2015, pp. 219–224.

[26] Z. Jin, P. An, C. Yang, L. Shen, Fast QTBT partition algorithm for intra frame coding through convolutional neural network, IEEE Access 6 (2018) 54660–54673.

[27] M. X. a. X. D. T. Li, A deep convolutional neural network approach for complexity reduction on intra-mode HEVC, IEEE International Conference on Multimedia and Expo (ICME), Jul., 2017.

[28] G. Huang, Z. Liu, L.V.D. Maaten, K.Q. Weinberger, Densely Connected Convolutional Networks, International Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, Jul. 2017.

[29] S.-H. Park, J.-W. Kang, Fast Multi-Type Tree Partitioning for Versatile Video Coding Using a Lightweight Neural Network, IEEE Trans. Multimedia 23 (2021) 4388–4399.

[30] I. Sergey, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, International conference on machine learning. PMLR, 2015.

[31] *VTM Reference Software*. Available: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM..

[32] F. Bossen, J. Boyce, X. Li, a. V. Seregin, K. Sühring, VTM Common Test Conditions and Software Reference Configurations for SDR Video, Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29-Doc. JVET-T2010-v1, 2020.

[33] G. Bjontegaard, Calculation of Average PSNR Difference Between RD-curves, ITU-T Q.6/SG16 VCEG 13th Meeting, Doc. VCEG-M33, 2001.