IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 14, 2021, accepted August 23, 2021, date of publication August 26, 2021, date of current version September 3, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3108238

Fast CU Decision-Making Algorithm Based on DenseNet Network for VVC

QIUWEN ZHANG, (Member, IEEE), RUIXIAO GUO, BIN JIANG, AND RIJIAN SU
College of Computer and Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou 450002, China

Corresponding author: Rijian Su (rijiansu@126.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61771432, Grant 61302118, and Grant 61773018; and in part by the Basic Research Projects of Education Department of Henan under Grant 21zx003 and Grant 20A880004.

ABSTRACT The joint video expert team (JVET) is currently developing a new video coding standard called H.266/Versatile Video Coding (VVC). Compared with High Efficiency Video Coding (HEVC), VVC has added a variety of coding tools. These tools have greatly improved video compression efficiency and maintained a high level video quality. However, due to the increase in computational complexity, the encoding time is much longer than HEVC. We propose a prediction tool based on DenseNet (a convolutional neural network) to decrease the VVC coding complexity. We predict the probability that the edge of 4×4 blocks in each 64×64 block is the division boundary by Convolutional Neural Networks (CNN). Then, we skip the unnecessary rate distortion optimization (RDO) and speed up the coding by probability vectors in advance. The proposed method can reduce the coding complexity of 46.10% in VTM10.0 intra coding, while Bjøntegaard delta bit rate (BDBR) only increases by 1.86%. In the sequence with a resolution greater than 1080P, the acceleration efficiency can be at 64.81%, the BDBR loss only increased by 1.92%.

INDEX TERMS Versatile video coding, convolutional neural network, coding unit partition.

I. INTRODUCTION

Due to the increase in IP video traffic in recent years [1] and the emergence of new video formats such as 4K, 8K, High Frame Rate (HFR), Wide Color Gamut (WCG) and VR video, the demand for video transmission bandwidth and storage has exploded. At present, how to more effectively encode new generation videos, such as ultra-high resolution and high dynamic range, has become one of the research hotspots in the world academia.

Some video coding standards have emerged, such as H.264/Advanced Video Coding (AVC) and HEVC. However, the compression ratios that these standards can achieve cannot keep up with the rapid growth in demand for video data. The International Telecommunication Union (ITU) and the ISO/IEC Moving Picture Experts Group (MPEG) formed the JVET to develop new video coding standards. In April 2018, the JVET officially named VVC [2].

Compared with HEVC, the compression ratio of VVC has been greatly improved, but its encoding time is also many times that of HEVC. VVC uses a hybrid coding technology framework. Its image division has evolved from a single, fixed division to a diverse and flexible division structure, which can more efficiently adapt to the encoding and decoding of high-resolution images. Among them, the QTMT division scheme [3], [4] is used in VVC to obtain better compression efficiency. The scheme has five division modes: Quad-Tree (QT), Binary-Tree-Vertical (BTV), Binary-Tree-Horizontal (BTH), Ternary-Tree-Vertical (TTV) and Ternary-Tree-Horizontal (TTH), which are more than HEVC in two ways, TTV and TTH, as shown in Fig.1. All five modes can be used, but QT splitting cannot be used for sub-blocks in other split modes. This segmentation scheme makes the segmented sub-blocks more suitable for the texture distribution of the image, which greatly improves the accuracy of the internal prediction and reduces the prediction residuals. Compared with HEVC [5], this change increases the coding efficiency by 8.5%. However, due to the two additional partition types, the computational complexity of RDO is much greater than that of HEVC, resulting in its encoding time several times that of HEVC.

An important direction of the current VVC video coding research is how to reduce the coding complexity and increase the coding speed without reducing the VVC coding efficiency or with little loss.

The associate editor coordinating the review of this manuscript and approving it for publication was Zhaoqing Pan.

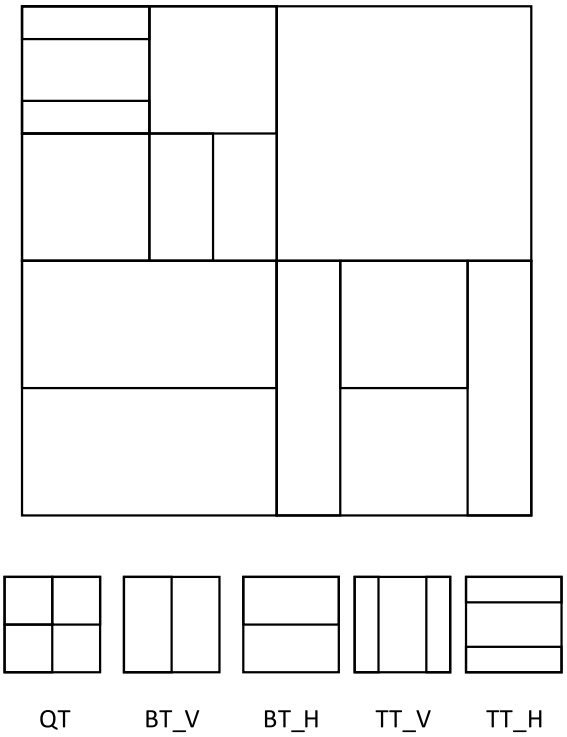

FIGURE 1. Schematic diagram of QTMT split structure.

Recently, the development of artificial neural networks has provided a new direction for the development of fast video coding.

In this paper, we design a DenseNet-based VTM10.0 internal encoder complexity reduction technique. We provide CNN with a 64 × 64 pixel luminance Coding Unit (CU) to predict a vector to represent the probability of an edge on the 4 × 4 boundary of the block. The encoder further uses this probability vector to skip the low-probability segmentation.

II. RELATED WORKS

The VVC video coding standard has five different CU partitioning methods. The attempts of different CU partitioning methods during encoding occupy most of the encoding time. Therefore, reducing the number of CU partitioning can significantly reduce the encoding time.

In 2019, Tissier *et al.* studied the CU partition complexity of video coding in [6]. They proposed that the computational complexity of block partitioning in VVC can be reduced to 3% of the original at most by predicting the correct CU splitting method. So a lot of coding time can be saved.

At present, most block segmentation acceleration algorithms reduce the computational complexity and save coding time by terminating unnecessary RDO in advance [5]. Algorithms are mainly divided into two categories, traditional algorithms and applied machine learning methods. Traditional algorithms analyze the complexity of the texture by extracting features, such as the variance and mean square error of the image, and set a threshold to determine whether to terminate the RDO calculation early.

Reference [7] proposed a fast intra algorithm based on variance and Sobel operator, where the variance and gradient information of the CU is calculated to determine whether the current CU should be split. Reference [8] used the Canny operator to extract the edge information of the image, and analyzed the edge information to determine the most likely dividing direction of the current CU. A threshold is set when the horizontal and vertical edge information. When the ratio of the features is higher than the threshold, the horizontal division is tried. When the reciprocal of the ratio is higher than the threshold, try the vertical division. If the two conditions are not met, try QT division. In [9], an algorithm proposed to calculate the rate distortion (RD) cost of the horizontal and vertical binary tree partitions. In the binary tree splitting, the cost is smaller, and the cost is usually higher in the MTT splitting. Reference [10] also uses the two features of image variance and gradient to accelerate segmentation. The method [11] uses Bayesian probabilities as features to skip unnecessary splitting modes. Most traditional algorithms only use one or two features, which can only filter out a small amount of redundancy. Therefore, the acceleration effect of block segmentation is limited.

Due to the rapid development of machine learning and deep learning in the past two years, the combination of block segmentation acceleration algorithms and machine learning has gradually increased. The trained Support Vector Machine (SVM) in [12]–[15] is used to filter possible segmentation strategies. In [12], 6 different SVM classifiers are trained for blocks from 32 × 32 to 16 × 8 and 8 × 16 to adapt for the situation of CU division of different sizes. In [13], two SVM classifiers are trained to divide the results into three categories. These two types of data are segmented and non segmented, and the error is small. The third type of data is at the boundary of the two types, so it needs to be calculated in the next step to determine whether to divide. In [14], 11 features of the SVM training image are used to determine whether the current CU needs to be segmented. Reference [15] trained multiple support vector machines to predict the probabilities of different partitioning methods respectively, and skip the partitioning methods with lower probability. Reference [16] used a decision tree to predict the segmentation mode. Reference [17] used Bayesian classifiers to speed up segmentation. Random forest classifiers are also used for fast CU partitioning [18]–[20], which use its characteristics to reduce the risk of division errors.

In addition to neural networks, other machine learning methods need to manually design the way to extract features. Neural networks can learn the required features through gradient descent without manual intervention. Under the appropriate training set, their learning speed and accuracy are generally high in the way of manual design. Recently, deep learning methods using neural networks have developed rapidly in the field of video coding [21]. In deep learning, the most suitable for processing image information is CNN

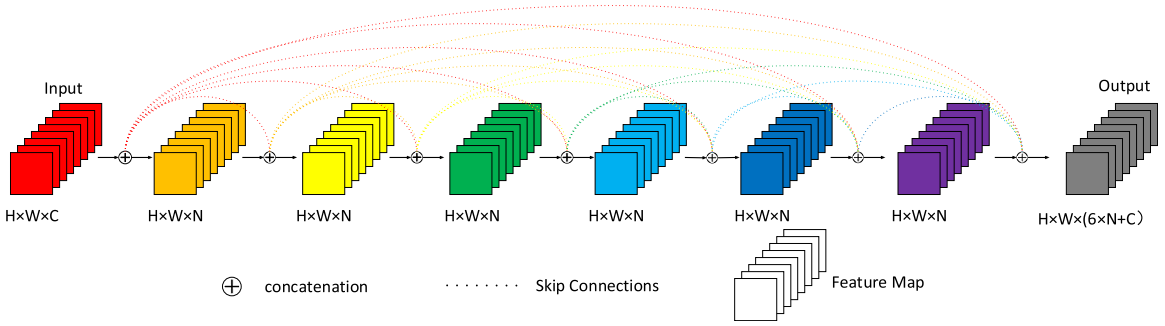

FIGURE 2. DenseNet architecture demonstration.

that can learn image spatial information. Due to the irregular shape and size of CUs, some methods based on CNN are used on these CUs. [22], [23] use an adaptive pooling layer to solve this problem. The adaptive pooling layer can compress feature maps of any size into a fixed size. References [24]–[26] Directly input fixed-size blocks into the network to predict the range of division depth, and terminate the RD calculation early through the depth range. Reference [27] converted the block structure into a hierarchical representation and directly predict the division of the entire Coding Tree Unit (CTU). Reference [28] trained three CNN classifiers to handle CUs of different depths and sizes. Reference [29] divide the CU into sub-blocks of the same size, and use CNN to predict the probability of each sub-block boundary as the division boundary to terminate the partial division mode early. In [30], the explicit VVC features (EVF) and the derived VVC features (DVF) that can be obtained during intra prediction are input into a lightweight neural network to determine which split modes to skip.

Because the VVC division method is more complicated, many methods that originally worked well on HEVC cannot be applied to VVC. We have designed a new fast partitioning method for coding blocks based on CNNs. This article divides the method into two steps. First, we train a CNN network to predict the probability that the edge of the 4×4 brightness block is the segmentation boundary. Then, it is divided according to the predicted probability.

III. PROPOSED METHOD

A. STRUCTURE OF CNN

Most existing algorithms combine the CNN network to only calculate the probability of a single CU for the next segmentation. The common problem of these algorithms is that the network model needs to be called multiple times to make predictions during the same CTU division process, which will bring a large time overhead. Moreover, the shapes of the CUs that need to be predicted vary greatly, which will bring difficulties to the training of the network.

Because the 128×128 size CTU only allows QT division, the proposed algorithm divides a CTU into four 64×64 blocks, which are used as a 4 batch input network. We use this batch to predict the probability that the boundaries of all 4×4 blocks within each 64×64 CU are divided boundaries. Then, we determine the division situation based on these probabilities, and make full use of vectorization to save the time that the network needs to run. Each CTU only needs to call the model once. The proposed network refers to DenseNet [31], and the main part of the network is made up of sub-blocks of DenseNet structure. The size of the feature map output by each convolutional layer is the same.

The structure of the DenseNet block is shown in Fig.2. DenseNet brings the idea of skip connection in ResNet [32] into the mechanism. A large number of jump connections make the propagation of features and gradients more effective, and alleviate the problem of gradient disappearance that often occurs when the neural network is too deep. The input of each convolutional layer in the DenseNet block comes from the output of all convolutional layers before that layer. These outputs are subjected to concatenation operation to form a feature map with more channels, and then use 1×1 convolution to adjust the number of channels to 4 times the number of output channels, and then pass through a 3×3 convolution layer, and use the rule function to activate later as the output of this layer. Considering one 1×1 convolution and one 3×3 convolution as one layer, and we use 6 layers for each DenseNet Block. Concatenation the feature map output by each layer as the output of the block. Due to the good performance of the attention mechanism in various computer vision tasks, we try to add an ECA attention module after each DenseNet Block. ECA is a lightweight channel attention mechanism, and its structure is shown in Fig.3. ECA attention module can play a good effect in most computer vision tasks. However, in the experiment of this task, it does not seem to significantly improve the network performance. So in the end we didn't use it in the model.

It can be seen from Fig.4 that the blue block is a two-dimensional convolution and Relu activation function with a 3×3 convolution kernel. The orange block represents DenseNet Block, which contains 6 convolutional layers using 1×1 and 3×3 size convolution kernels. The layers represented by red and green are collectively called the transition layer, which is used to compress the amount of data. Red represents the average pooling layer. Green represents the convolution kernel with 1×1 convolutional layer and Relu activation function. Gray represents the global pooling layer, which is used to transform the feature map into a vector.

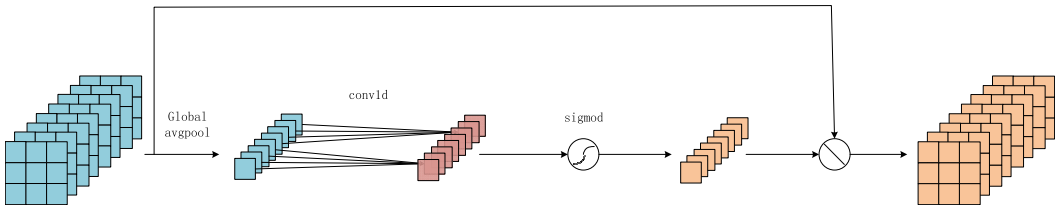

FIGURE 3. Efficient channel attention, ⊗ indicates that the corresponding channel is multiplied.

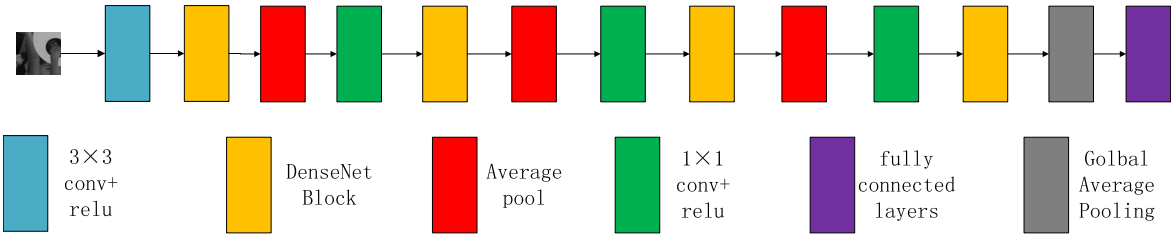

FIGURE 4. Structure of the proposed model.

Purple represents the fully connected layer, which is used to output the final result. The hyperparameters of the network are shown in Table 1.

TABLE 1. Number of parameters in each layer of CNN.

Layers	output size(w×H×N)	operation	Channel of conv in dense block(N)
convolution	64×64×16	3×3 conv	
Densenet block(1)	64×64×208	$\begin{cases}1\times1\text{ conv}\\3\times3\text{ conv}\end{cases}\times6$	32
transition layer(1)	21×21×208	3×3 Avgpool stride3	
	21×21×104	1×1 conv	
Densenet block(2)	21×21×392	$\begin{cases}1\times1\text{ conv}\\3\times3\text{ conv}\end{cases}\times6$	48
transition layer(2)	7×7×392	3×3 Avgpool stride3	
	7×7×196	1×1 conv	
Densenet block(3)	7×7×484	$\begin{cases}1\times1\text{ conv}\\3\times3\text{ conv}\end{cases}\times6$	48
transition layer(3)	3×3×484	2×2 Avgpool Stride2	
	3×3×242	1×1 conv	
Densenet block(4)	3×3×530	$\begin{cases}1\times1\text{ conv}\\3\times3\text{ conv}\end{cases}\times6$	48
FC layer	1×1×530	3×3 global avgpool	
	480	linear	

Four 64 × 64 luma blocks are the input of CNN, and one CTU is QT divided into fours CUs of 64 × 64 size. In order to use vectorization, we input these four brightness blocks into the network as a batch to speed up the prediction. The output is a probability vector corresponding to 4 × 4 block boundaries of four 64 × 64 CU partitions. The input sizes of CNN correspond to the maximum luminance conversion block (64 × 64) in the "All Intra" configuration. Fig.5 shows the matching of the CU partition and the probability vector. For example, the first value of the vector corresponds to the right boundary of the 4 × 4 block in the upper left corner, and the second value corresponds to the right boundary of the next 4 × 4 block. The 241st value represents the lower boundary of the first 4 × 4 block in the upper left corner.

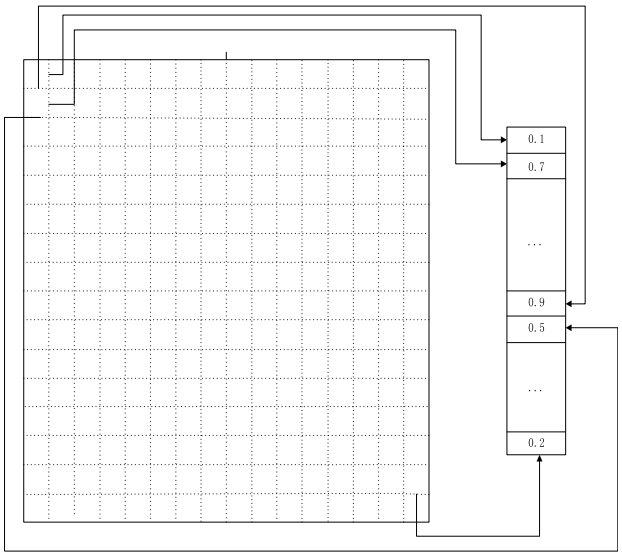

FIGURE 5. Mapping of probability vectors to 4 × 4 block boundaries. The first 240 elements of the probability vector correspond to the probability of all vertical boundaries, and the last 240 elements correspond to the probability of the horizontal boundary.

B. DIVISIONAL JUDGMENT

According to the coordinates and shape of the upper left corner of the current block, we can obtain the vector values corresponding to the five dividing lines in the block. The

probability of BT division is determined by the average value of the corresponding vector. The probability of QT division is the average of BT divisions in two directions. The probability of TT division is the larger of the corresponding two dividing line probabilities.

The probability of BTV is calculated as:

$$\begin{cases} n = \dfrac{x}{4} + \dfrac{w}{8} - 1 \\ j = \dfrac{y}{4} + n \times 16 \\ P = \dfrac{4}{h} \displaystyle\sum_{i=1}^{h} p_{j+i} \end{cases} \tag{1}$$

The probability of BTH is calculated as:

$$\begin{cases} n = \dfrac{y}{4} + \dfrac{h}{8} - 1 \\ j = \dfrac{x}{4} + n \times 16 + 240 \\ P = \dfrac{4}{w} \displaystyle\sum_{i=1}^{w} p_{j+i} \end{cases} \tag{2}$$

The probability of TTV is calculated as:

$$\begin{cases} n_1 = \dfrac{x}{4} + \dfrac{w}{16} - 1 \\ n_2 = \dfrac{x}{4} + \dfrac{3}{16} \times w - 1 \\ j_1 = \dfrac{y}{4} + n_1 \times 16 \\ j_2 = \dfrac{y}{4} + n_2 \times 16 \\ P_1 = \displaystyle\sum_{i=1}^{h/4} p_{j_1+i} \\ P_2 = \displaystyle\sum_{i=1}^{h/4} p_{j_2+i} \\ P = \dfrac{4}{h} \mathrm{Max}(P_1, P_2) \end{cases} \tag{3}$$

The probability of TTH is calculated as:

$$\begin{cases} n_1 = \dfrac{y}{4} + \dfrac{h}{16} - 1 \\ n_2 = \dfrac{y}{4} + \dfrac{3}{16} \times h - 1 \\ j_1 = \dfrac{x}{4} + n_1 \times 16 + 240 \\ j_2 = \dfrac{x}{4} + n_2 \times 16 + 240 \\ P_1 = \displaystyle\sum_{i=1}^{w/4} p_{j_1+i} \\ P_2 = \displaystyle\sum_{i=1}^{w/4} p_{j_2+i} \\ P = \dfrac{4}{w} \mathrm{Max}(P_1, P_2) \end{cases} \tag{4}$$

where x and y are the coordinates of the upper left corner of the current CU in the 64×64 block, and h and w are the height and width of the current block. P_{j+i} is the value of the element with index $j + i$ in the processed probability vector.

The threshold we use is dynamic. When the size of the CU is larger, more vectors are involved in the calculation, and a small amount of prediction error causes less impact, so a higher threshold is used. When the size of the CU is small, there are fewer vectors involved in the calculation, so there is a greater risk, so a lower threshold is used. The size of a CU is usually related to its depth. A deeper CU usually has a smaller size. For the convenience of calculation, the threshold is set as below:

$$T = 0.7 - 0.1 \times depth \tag{5}$$

where *depth* represents the depth of the current CU, which can be obtained from the currDepth property of the Partitioner class in the VTM. It is numerically equal to the sum of QtDepth and MtDepth.

The process of embedding in VTM software is shown in Fig.6.

After the encoder divides each frame of image into CTU, it performs CNN prediction on the CTU currently to be encoded. After entering the CU encoding stage, try various encoding methods including 5 partitioning modes for the current CU. When trying the segmentation mode, the probabilities of various segmentation methods are calculated according to the area attributes of the current CU. If the probability of the currently tried segmentation mode is greater than the threshold, an attempt is made, otherwise the attempt of the segmentation mode is skipped. In order to reduce more coding time, we also skipped the case where the probability difference between the horizontal split and the vertical split in the BT and TT splits is large.

C. TRAINING

Because the value to be predicted is between [0, 1], we use the cross-entropy loss function, which is defined as:

$$Loss = -\frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{n} p(x_{i,j}) \log(q(x_{i,j})) \tag{6}$$

where m is the number of samples in a batch, n is the number of elements in each sample, $p(x_{i,j})$ is the true value of the $j - th$ vector of the $i - th$ sample, $q(x_{i,j})$ is the corresponding predicted value. Adam optimizer is used [34] to perform gradient descent on CNN. The training process uses the pytorch1.7.0 framework in the python3.7 environment, and the learning rate adjustment strategy selects the cosine annealing strategy (CosineAnnealingWarmRestarts). We trained 20 generations on GTX1650 GPU. Batchsize is 16.

For the input data set of 64 × 64 luminance blocks and their corresponding label vectors, 100 images are extracted at equal intervals from the 800 HR samples of the Div2k [35] dataset used to train the super-resolution network, and these

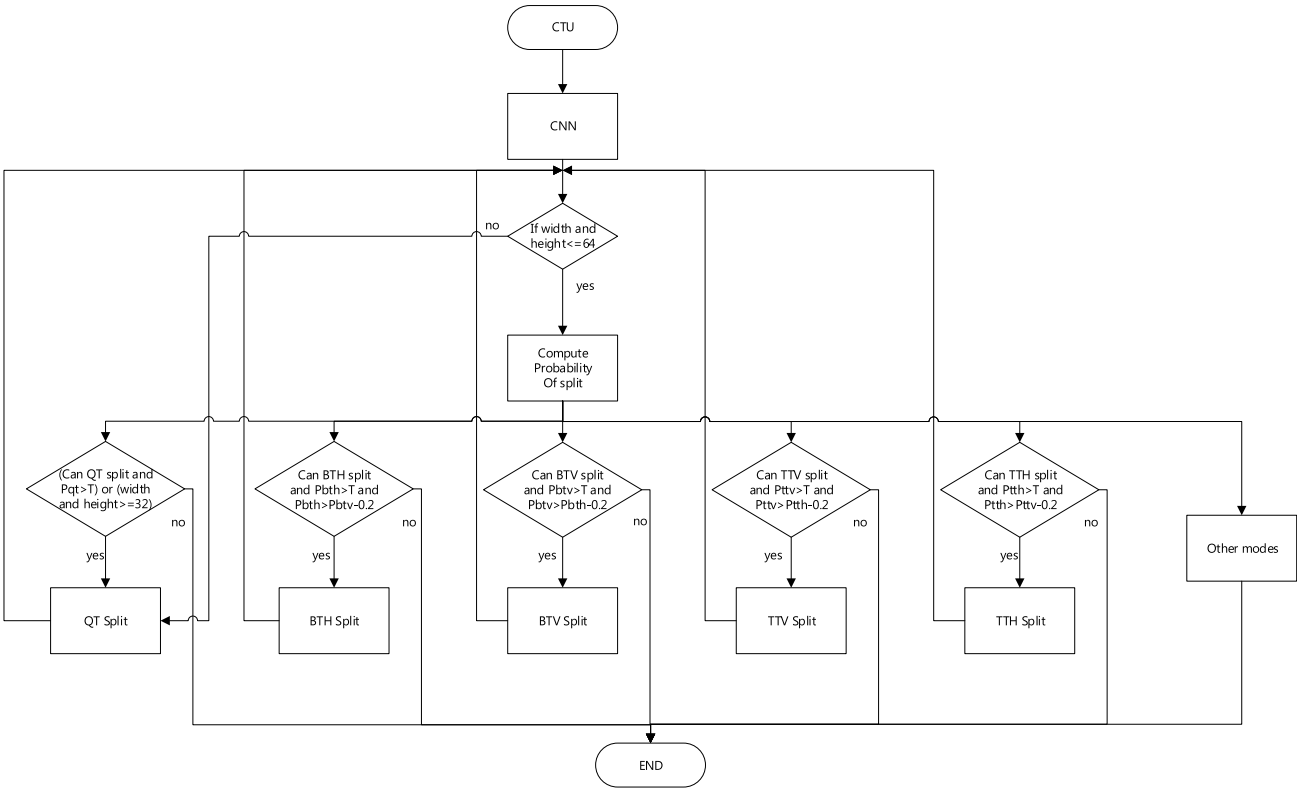

FIGURE 6. Process of skipping CU split, CNN means using a trained CNN model for prediction .Pqt, Pbth, etc. represent the probability of the corresponding split mode. T is the preset threshold. Other modes indicate prediction modes other than CU split, such as intra prediction, inter prediction, and so on. After executing the Split operation, it will enter the sub-CU and repeat all operations after CNN (not including CNN inference).

images are divided into 64 × 64 blocks as data set. The luminance signal of the 64 × 64 luminance block is obtained from the VTM when the picture in the data set is encoded. These data sets consist of static images, because the proposed solutions are mainly used for "AI" configuration. This has better diversity than the video sequence dataset. In order to test the generalization ability of the network and the effect of real coding, we did not use the common test conditions (CTC) [36] sequence for training. The input data set is coded by the VTM 10.0 software under the "All Intra" configuration to establish the corresponding label. QTMT partition information is collected for each 64 × 64 CU and convert it to the output format of CNN. The label consists of a one-dimensional vector of 480 elements consisting of 1 (for splitting boundary) and 0 (not for splitting boundary). The partition information comes from the code stream analysis tool (DecoderAnalyserApp) in VTM.

IV. EXPERIMENTAL RESUTS

This section introduces the experimental setup in detail and compares our results with several advanced technologies.

A. EXPERIMENTAL SETUP

All experiments are carried out under the "All Intra" configuration, and the VTM 10.0. Each encoding and CNN prediction is performed individually on an Intel i5-8500 processor running at 3.00GHz on the windows operating system. The test set consists of 25 sequences with different resolutions. It contains a wide range of resolutions, textures, bit depths and motion. The test sequence is divided into seven categories: A1 (3840 × 2160),A2 (3840 × 2160), B (1920 × 1080), C (832 × 480), D (416 × 240), E (1280 × 720) and F (832 × 480 to 1920 × 1080). They are encoded according to four quantization parameter (QP) values: 22, 27, 32, and 37.

The coding quality is measured by BDBR and complexity reduction, and the coding time saving rate (ΔT) is determined as:

$$\Delta T = \frac{1}{4} \sum_{QP=22,27,32,37} \frac{T_{OC} - T_{SC}}{T_{OC}} \times 100\% \qquad (7)$$

Among them, T_{OC} is the reference coding time of the VTM10.0 anchor point, and T_{SC} is the coding time of our algorithm. We counted the inference time during the test phase of training the network. When the network runs on the CPU and the input size is 4 × 64 × 64, the inference time of the network is about 0.18 seconds. Due to the different performance of different platforms, this time does not count the time spent by the neural network.

B. RESULTS AND ANALYSIS

Because VTM 4.0, VTM 5.0 and VTM 10.0 use the same CU splitting scheme, our comparison with [11] and [19] is reasonable. We conducted experiments on the role of the ECA module in the proposed algorithm, as shown in Table 2. ΔT in Table 2 does not include CNN inference time.

TABLE 2. Experimental data with or without ECA module.

Class	Sequence	Our process(add ECA)		Our process(No ECA)	
		BDBR(%)	ΔT(%)	BDBR(%)	ΔT(%)
A1	Campfire	2.21	71.34	1.94	70.72
	FoodMarket4	2.10	77.05	1.88	76.26
	Tango2	2.41	76.53	2.23	76.34
A2	CatRobot	2.85	75.11	2.71	75.08
	DaylightRoad2	1.93	77.78	2.00	77.83
	ParkRunning3	0.85	69.85	0.79	69.45
B	BasketballDrive	2.07	62.73	2.03	63.74
	MarketPlace	1.49	81.10	1.39	80.86
	RitualDance	2.30	76.06	2.12	75.69
	BQTerrace	1.52	51.61	1.55	52.37
	Cactus	1.93	58.98	1.88	58.39
C	BasketballDrill	2.28	34.40	2.32	35.49
	BQMall	1.58	36.82	1.64	37.61
	PartyScene	0.76	27.34	0.76	27.73
	RaceHorses	1.02	39.39	1.03	39.43
D	BasketballPass	0.98	23.50	1.03	25.54
	BlowingBubbles	0.419	16.60	0.56	18.61
	BQSquare	0.41	17.94	0.58	17.85
	RaceHorses	0.66	23.21	0.80	23.13
E	FourPeople	2.69	54.98	2.67	55.10
	Johnny	3.32	55.54	3.25	55.13
	KristenAndSara	2.42	50.79	2.43	50.87
F	BasketballDrillText	1.93	33.03	2.06	32.92
	SlideShow	2.31	33.10	2.74	56.22
	SlideEditing	1.86	37.13	2.50	42.42
	Mean	1.77	51.34	1.79	51.79

According to the data in Table 2, it can be seen that whether the ECA module is used has no obvious impact on the final performance. On BDBR, the scheme using ECA module only reduces the average loss by 0.02%. In most high-resolution sequence tests, the BDBR loss of the scheme using the ECA module is higher than that of the scheme not using this module. In the coding time comparison that does not consider the network model inference time, the scheme using the ECA module does not show obvious advantages. Even if the ECA module is a lightweight attention module, the global average pooling operation will still take a lot of time. After experiments, the ECA module will increase the inference time of CNN by about 10% when performing calculations on the CPU. In summary, the ECA module is not suitable for this solution.

Our algorithm is only applied to a 128 × 128 CTU. Our algorithm saves less time on low-resolution sequences, because the proportion of the image area occupied by the CTU with a size of less than 128 × 128 at the boundary of the low-resolution image is larger than that of the high-resolution image and our training set only images with 2K resolution are included. But on the high-resolution A1 and A2 sequences, the proposed algorithm saves 74.28% of the coding on average, and only brings a 1.925% BDBR loss. For the low-resolution C-D, the proposed method can reduce the complexity by 27.39% and increase the BDBR by 1.01%. The training database consists of 2K images, so the proposed CNN performs better on high-resolution sequences. At the same time, since the training set does not contain pictures with text content, there is a higher BDBR loss in the sequence where the main content is text in the E class.

Table 3 shows how our method compares with other methods. The data of other algorithms are calculated from the data given in their papers. Because the sequences adopted in different papers are different, we only make comparisons between classes. Compared with [11], our method can save more coding time in most cases. In the higher resolution sequences of A1 and A2, even if the reasoning time of the network is added, the time saved by our scheme is still much higher than [11], and the BDBR loss gap is within an acceptable range. When ignoring the network inference time, our method outperforms [19] on high-resolution sequences. When the reasoning time is added, our method is better than [19] on the A2 class, and the time saving is very close to [19], while the BDBR loss is significantly better than [19] on the A1 class. Our algorithm saves less time when the resolution is lower because we only predict blocks that meet the size of 128 × 128. In most of the sequences, we obtained a time saving rate much higher than [30] under an acceptable BDBR loss.

The method is divided into two parts: prediction and decision-making. The running time of the decision-making part is extremely short and can be ignored. The time for the prediction part using the neural network is shown in Fig.7. The running time of the prediction part occupies an average of 6.5% of the original VTM encoding time. And this ratio can be further shortened by means of model quantification or graphics card acceleration.

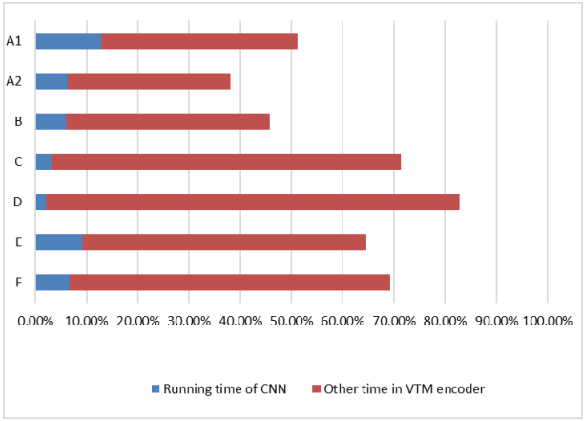

FIGURE 7. Running time of the proposed MSE-CNN model and the VTM encoder.

TABLE 3. Comparison with other algorithms.

Class	S. Park et al. [11],VTM4.0(TABLE 5.)		T. Amestoy et al.[19],VTM5.0(TABLE VIII,MT_2)		S Park et al.[30],VTM4.0(TABLE IV, $\alpha = 1/2$)		Our process,VTM10.0		
	BDBR(%)	ΔT(%)	BDBR (%)	ΔT(%)	BDBR (%)	ΔT(%)	BDBR (%)	ΔT(%)	ΔT(%)(Including CNN inference time)
A1	0.67	32.00	3.06	65.10	0.34-	32.33	2.02	74.44	61.59
A2	1.07	33.00	1.91	62.90	0.45	29.33	1.83	74.12	68.03
B	0.98	33.75	2.60	62.48	0.53	26.00	1.79	66.21	60.18
C	1.17	35.25	2.59	61.97	0.48	24.75	1.44	35.06	31.84
D	0.81	35.00	1.84	54.63	0.32	23.75	0.74	21.28	19.22
E	1.34	33.67	1.82	54.95	0.61	25	2.78	53.70	44.54

V. CONCLUSION

This paper proposes a CU split acceleration scheme based on DenseNet network. CNN is used to analyze the texture in every 4 64×64 coded blocks, and predict the probability that each 4×4 block in these blocks is a partition boundary. Starting from the probability of the boundary, the segmentation probability is derived and compared with the preset threshold. Compared with the original encoding time of VTM, the execution time of CNN is shorter. In the "All Intra" configuration under VTM 10.0, the proposed solution reduces the complexity by 46.10%, and BDBR slightly increases by 1.86%. When using high-resolution sequences, the acceleration effect is higher, up to 64.81%, but requires 1.92% of the BDBR overhead. These results prove the effectiveness of the proposed method and motivate us to conduct further research and analysis. Since the training set is small and the network is not optimized to the optimum, the proposed solution still has a lot of room for improvement. We will try to use a larger data set for training and further optimize the structure of the network. At the same time, we will also try to improve the performance of our method on low-resolution sequences.

REFERENCES

[1] *Cisco Visual Networking Index : Forecast and Trends, 2017–2022*, Cisco, San Jose, CA, USA, 2019.

[2] J. Chen, M. Karczewicz, Y.-W. Huang, K. Choi, J.-R. Ohm, and G. J. Sullivan, "The joint exploration model (JEM) for video compression with capability beyond HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 5, pp. 1208–1225, May 2020, doi: 10.1109/TCSVT.2019.2945830.

[3] Y.-W. Huang, C.-W. Hsu, C.-Y. Chen, T.-D. Chuang, S.-T. Hsiang, C.-C. Chen, M.-S. Chiang, C.-Y. Lai, C.-M. Tsai, Y.-C. Su, Z.-Y. Lin, Y.-L. Hsiao, O. Chubach, Y.-C. Lin, and S.-M. Lei, "A VVC proposal with quaternary tree plus binary-ternary tree coding block structure and advanced coding techniques," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 5, pp. 1311–1325, May 2020, doi: 10.1109/TCSVT.2019.2945048.

[4] B. Bross, K. Andersson, M. Blaser, V. Drugeon, S.-H. Kim, J. Lainema, J. Li, S. Liu, J.-R. Ohm, G. J. Sullivan, and R. Yu, "General video coding technology in responses to the joint call for proposals on video compression with capability beyond HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 5, pp. 1226–1240, May 2020, doi: 10.1109/TCSVT.2019.2949619.

[5] A. Wieckowski, J. Ma, H. Schwarz, D. Marpe, and T. Wiegand, "Fast partitioning decision strategies for the upcoming versatile video coding (VVC) standard," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Taipei, Taiwan, Sep. 2019, pp. 4130–4134.

[6] A. Tissier, A. Mercat, T. Amestoy, W. Hamidouche, J. Vanne, and D. Menard, "Complexity reduction opportunities in the future VVC intra encoder," in *Proc. IEEE 21st Int. Workshop Multimedia Signal Process. (MMSP)*, Kuala Lumpur, Malaysia, Sep. 2019, pp. 1–6.

[7] 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, pp. 107900–107911, 2020, doi: 10.1109/ACCESS.2020.3000565.

[8] N. Tang, J. Cao, F. Liang, J. Wang, H. Liu, X. Wang, and X. Du, "Fast CTU partition decision algorithm for VVC intra and inter coding," in *Proc. IEEE Asia Pacific Conf. Circuits Syst. (APCCAS)*, Bangkok, Thailand, Nov. 2019, pp. 361–364.

[9] M. Lei, F. Luo, X. Zhang, S. Wang, and S. Ma, "Look-ahead prediction based coding unit size pruning for VVC intra coding," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Taipei, Taiwan, Sep. 2019, pp. 4120–4124.

[10] J. Chen, H. Sun, J. Katto, X. Zeng, and Y. Fan, "Fast QTMT partition decision algorithm in VVC intra coding based on variance and gradient," in *Proc. IEEE Vis. Commun. Image Process. (VCIP)*, Sydney, NSW, Australia, Dec. 2019, pp. 1–4.

[11] S.-H. Park and J.-W. Kang, "Context-based ternary tree decision method in versatile video coding for fast intra coding," *IEEE Access*, vol. 7, pp. 172597–172605, 2019, doi: 10.1109/ACCESS.2019.2956196.

[12] F. Chen, Y. Ren, Z. Peng, G. Jiang, and X. Cui, "A fast CU size decision algorithm for VVC intra prediction based on support vector machine," *Multimedia Tools Appl.*, vol. 79, nos. 37–38, pp. 27923–27939, Oct. 2020.

[13] X. Liu, Y. Li, D. Liu, P. Wang, and L. T. Yang, "An adaptive CU size decision algorithm for HEVC intra prediction based on complexity classification using machine learning," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 1, pp. 144–155, Jan. 2019, doi: 10.1109/TCSVT.2017.2777903.

[14] L. Zhu, Y. Zhang, Z. Pan, R. Wang, S. Kwong, and Z. Peng, "Binary and multi-class learning based low complexity optimization for HEVC encoding," *IEEE Trans. Broadcast.*, vol. 63, no. 3, pp. 547–561, Sep. 2017, doi: 10.1109/TBC.2017.2711142.

[15] Y. Zhang, Z. Pan, N. Li, X. Wang, G. Jiang, and S. Kwong, "Effective data driven coding unit size decision approaches for HEVC INTRA coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 11, pp. 3208–3222, Nov. 2018, doi: 10.1109/TCSVT.2017.2747659.

[16] Z. Wang, S. Wang, J. Zhang, S. Wang, and S. Ma, "Effective quadtree plus binary tree block partition decision for future video coding," in *Proc. Data Compress. Conf. (DCC)*, A. Bilgin, M. W. Marcellin, J. SerraSagrista, and J. A. Storer, Eds., Apr. 2017, pp. 23–32.

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

[18] Q. Zhang, Y. Wang, L. Huang, and B. Jiang, "Fast CU partition and intra mode decision method for H.266/VVC," *IEEE Access*, vol. 8, pp. 117539–117550, 2020, doi: 10.1109/ACCESS.2020.3004580.

[19] T. Amestoy, A. Mercat, W. Hamidouche, D. Menard, and C. Bergeron, "Tunable VVC frame partitioning based on lightweight machine learning," *IEEE Trans. Image Process.*, vol. 29, pp. 1313–1328, 2020, doi: 10.1109/TIP.2019.2938670.
[20] T. Amestoy, A. Mercat, W. Hamidouche, C. Bergeron, and D. Menard, "Random forest oriented fast QTBT frame partitioning," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Brighton, U.K., May 2019, pp. 1837–1841.
[21] D. Liu, Z. Chen, S. Liu, and F. Wu, "Deep learning-based technology in responses to the joint call for proposals on video compression with capability beyond HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 5, pp. 1267–1280, May 2020, doi: 10.1109/TCSVT.2019.2945057.
[22] J. Zhao, Y. Wang, and Q. Zhang, "Adaptive CU split decision based on deep learning and multifeature fusion for H.266/VVC," *Sci. Program.*, vol. 2020, Aug. 2020, Art. no. 8883214.
[23] G. Tang, M. Jing, X. Zeng, and Y. Fan, "Adaptive CU split decision with pooling-variable CNN for VVC intra encoding," in *Proc. IEEE Vis. Commun. Image Process. (VCIP)*, Sydney, NSW, Australia, Dec. 2019, pp. 1–4.
[24] 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, pp. 54660–54673, 2018, doi: 10.1109/ACCESS.2018.2872492.
[25] Z. Jin, P. An, L. Shen, and C. Yang, "CNN oriented fast QTBT partition algorithm for JVET intra coding," in *Proc. IEEE Vis. Commun. Image Process. (VCIP)*, Dec. 2017, pp. 1–4.
[26] Z. Wang, S. Wang, X. Zhang, S. Wang, and S. Ma, "Fast QTBT partitioning decision for interframe coding with convolution neural network," in *Proc. 25th IEEE Int. Conf. Image Process. (ICIP)*, Athens, Greece, Oct. 2018, pp. 2550–2554.
[27] M. Xu, T. Li, Z. Wang, X. Deng, R. Yang, and Z. Guan, "Reducing complexity of HEVC: A deep learning approach," *IEEE Trans. Image Process.*, vol. 27, no. 10, pp. 5044–5059, Oct. 2018, doi: 10.1109/TIP.2018.2847035.
[28] Y. Zhang, G. Wang, R. Tian, M. Xu, and C. C. J. Kuo, "Texture-classification accelerated CNN scheme for fast intra CU partition in HEVC," in *Proc. Data Compress. Conf. (DCC)*, Snowbird, UT, USA, Mar. 2019, pp. 241–249.
[29] A. Tissier, W. Hamidouche, J. Vanne, F. Galpin, and D. Menard, "CNN oriented complexity reduction of VVC intra encoder," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Abu Dhabi, United Arab Emirates, Oct. 2020, pp. 3139–3143.
[30] S.-H. Park and J. Kang, "Fast multi-type tree partitioning for versatile video coding using a lightweight neural network," *IEEE Trans. Multimedia*, early access, Dec. 2, 2020, doi: 10.1109/TMM.2020.3042062.
[31] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, vol. 1, no. 2, p. 3.
[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[33] Q. Wang, B. Wu, P. Zhu, P. Li, W. Zuo, and Q. Hu, "ECA-Net: Efficient channel attention for deep convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11534–11542.
[34] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent.*, San Diego, CA, USA, 2015, pp. 1–15.
[35] E. Agustsson and R. Timofte, "NTIRE 2017 challenge on single image super-resolution: Dataset and study," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Honolulu, HI, USA, Jul. 2017, pp. 1122–1131.
[36] F. Bossen, J. Boyce, K. Suehring, X. Li, and V. Seregin, *JVET Common Test Conditions and Software Reference Configurations for SDR Video*, document JVET-M1010, Jan. 2019.

QIUWEN ZHANG (Member, IEEE) received the Ph.D. degree in communication and information systems from Shanghai University, Shanghai, China, in 2012. Since 2012, he has been with the Faculty of the College of Computer and Communication Engineering, Zhengzhou University of Light Industry, where he is currently an Associate Professor. He has published over 30 technical papers in the field of pattern recognition and image processing. His major research interests include 3D signal processing, machine learning, pattern recognition, video codec optimization, and multimedia communication.

RUIXIAO GUO received the B.S. degree in the Internet of Things engineering from Zhengzhou University of Aeronautics, Zhengzhou, China, in 2018. He is currently pursuing the master's degree in electronic information with the School of Computer and Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou. His current research interests include image processing, deep learning, and extensions of the high efficiency video coding.

BIN JIANG received the M.S. degree in computer application technology from Henan University, Kaifeng, China, in 2009, and the Ph.D. degree in electronic circuit and system from Beijing University of Technology, Beijing, China, in 2014. He is currently a Lecturer with Zhengzhou University of Light Industry, China. His current research interests include 2D and 3D video processing, image processing, and multimedia systems.

RIJIAN SU received the Ph.D. degree in control science and engineering from Huazhong University of Science and Technology, Wuhan, China, in 2010. He is currently a Professor with Zhengzhou University of Light Industry, China. His current research interests include image processing, Bayesian estimation, and intelligent computation.