J. Vis. Commun. Image R. 88 (2022) 103621

Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

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

Full Length Article

A GCN-based fast CU partition method of intra-mode VVC☆

Saiping Zhang *, Shixuan Feng, Jingwu Chen, Chunjie Zhou, Fuzheng Yang

The State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China

ARTICLE INFO

MSC:
41A05
41A10
65D05
65D17

Keywords:
Versatile Video Coding
Intra partition mode prediction
Complexity reduction
Global convolutional network

ABSTRACT

In this paper, a global convolutional network (GCN)-based fast coding unit (CU) partition method of intra-mode VVC is proposed. By using the GCN module with large kernel size convolutions, the proposed method can capture global information in CUs, leading to an accurate partition mode prediction in the quad-tree plus multi-type tree (QTMT) structure. Ranked according to predicted probabilities, the partition modes with lower probabilities are discarded, which reduces the computational complexity of VVC. Additionally, tradeoffs between performance and complexity can be achieved with different strategies. Experimental results demonstrated that the proposed method can reduce encoding time by 51.06%~61.15% while increasing Bjøntegaard delta bit-rate (BD-BR) by 0.84%~1.52% when implemented in VTM 10.0, outperforming the state-of-the-art methods, and that the proposed method can be used to accelerate VVenC 1.0 at the preset *slower*, achieving higher performance and lower complexity compared with the original VVenC 1.0 at the presets *slow* and *medium*.

1. Introduction

Recent years have witnessed the tremendous development in video services. As interest in ultra-high definition (UHD) videos [1], panoramic videos [2] and virtual reality (VR) videos [3] has grown, there are now very strict criteria for video compression algorithms. In this situation, the High Efficiency Video Coding (HEVC) [4] standard is unable to effectively compress such a significant amount of video data. To this end, the Joint Video Experts Team (JVET) of the International Telecommunication Union-Telecommunication Standardization Sector (ITU-T) Video Coding Experts Group (VCEG) and the International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) Moving Picture Experts Group (MPEG) are devoted to developing the next-generation video coding standard, i.e., Versatile Video Coding (VVC) [5]. In comparison to its predecessor HEVC, VVC uses a large number of advanced video coding techniques, including the quad-tree plus multi-type tree (QTMT) structure of coding unit (CU) partition, finer-granularity angular prediction, multiple reference lines and adaptive motion vector resolution. When compared to HEVC, VVC can save an average of 44.4% bit-rate while retaining the same coding performance because of the usage of these potent coding techniques. However, the encoding time of VVC becomes 10.2 times longer than that of HEVC [6], which makes it difficult to employ VVC in the majority of real-world applications. Particularly, the QTMT structure of the CU partition takes up around 99% of the encoding time of VVC [7]. It is crucial to avoid redundant CU partition modes in the QTMT structure and expedite the CU partition decision-making process. In the past few years, a number of fast CU partition methods [8–27] have been proposed. Particularly, these works can be divided into two groups: the conventional data analysis-based methods [8–15] and the deep learning (DL)-based methods [16–27]. The conventional data analysis-based methods rely heavily on manually creating the correlations between variables that can be computed, including the variance and gradient, and CU partition modes, making it simple to settle on the local optimal solution. The extensive DL-based methods, on the other hand, have surpassed the conventional data analysis-based methods as a result of the tremendous development of DL in the past decade. By constructing a large-scale database, the DL-based methods train networks to automatically build up the complicated correlations between CU features and partition modes. Guided by a well-designed loss function, networks are more readily able to identify the global optimal solution.

To the best of our knowledge, however, the performance of existing DL-based fast CU partition methods is limited since these methods are seldom designed based on investigating global information in CUs. Considering selecting the best partition mode of a CU is equivalent to segmenting the CU into small blocks in the best way (six CU partition modes in the QTMT structure, i.e., non-splitting, quad-tree, horizontal binary-tree, vertical binary-tree, horizontal ternary-tree and vertical

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

* Corresponding author.
E-mail address: spzhang@stu.xidian.edu.cn (S. Zhang).

https://doi.org/10.1016/j.jvcir.2022.103621
Received 24 March 2022; Received in revised form 20 July 2022; Accepted 13 August 2022
Available online 24 August 2022

Table 1
Key abbreviations in this paper.

Abbreviation	Definition
GCN	Global convolutional network
HEVC	High Efficiency Video Coding
VVC	Versatile Video Coding
QTMT	Quad-tree plus multi-type tree
CU	Coding unit
DL	Deep learning
VTM	VVC Test Model
VVenC	Fraunhofer Versatile Video Encoder
CTU	Coding tree unit
RDcost	Rate distortion cost
RDO	Rate distortion optimization
RDT	Rate–distortion-time
CNN	Convolutional neural network
FC	Fully-connected
QP	Quantization parameter
AI	All-intra
BD-BR	Bjøntegaard delta bit-rate

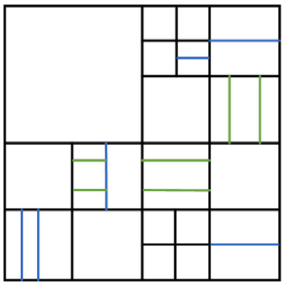

Fig. 1. An example of the QTMT structure in VVC.

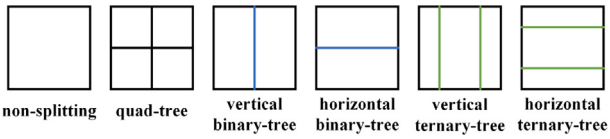

Fig. 2. Six available CU partition modes in VVC.

ternary-tree, correspond to six segmentation ways, respectively), capturing global information in CUs to achieve accurate segmentation can be helpful for predicting CU partition modes.

The global convolutional network (GCN) was proposed in [28] as a solution for the classification and localization problems associated with segmenting images. By using the GCN with large kernel size convolutions to capture global pixel correlations, the method proposed in [28] has achieved superior performance. Motivated by the success of the GCN in [28], we apply it to capture global information in CUs, paving the way for accurately segmenting CUs and determining the appropriate CU partition mode, thus expediting the CU partition process with tolerable coding performance loss.

In this paper, we propose a GCN-based fast CU partition method to reduce the computational complexity of the QTMT structure of the intra-mode VVC. The central idea of the proposed method is to apply the GCN, which was proposed to capture pixel correlations and achieved superior performance in image segmentation, to the CU partition mode decision. The probabilities of CU partition modes are output by the proposed network. After ranking the partition modes according to the predicted probabilities, we discard redundant modes with lower probabilities to reduce the encoding time of VVC. Additionally, the proposed method can achieve tradeoffs between the coding performance and the computational complexity with different strategies according to the preferences of users.

Our main contributions are summarized as:

1. A GCN-based fast CU partition network is proposed to reduce the computational complexity of the QTMT structure of the intra-mode VVC. To the best of our knowledge, the proposed method is the first attempt to apply the idea of image segmentation to the CU partition.
2. The proposed method can achieve tradeoffs between the coding performance and the computational complexity with different strategies according to the preferences of users.
3. The proposed method is implemented both on VVC Test Model (VTM) 10.0 and Fraunhofer Versatile Video Encoder (VVenC) 1.0 at the preset *slower*. Experimental results have demonstrated that the proposed method outperforms the state-of-the-art fast CU partition methods and the original VVenC 1.0 at the presets *slow* and *medium*.

For the sake of the convenience of readers, we have included key abbreviations in Table 1.

The remainder of the paper is organized as follows. A brief overview of CU partition in VVC and related works are introduced in Section 2. The proposed fast CU partition method is detailed in Section 3. Experimental results are shown in Section 4 to verify expected performance when the proposed fast CU partition method is used. Finally, Section 5 concludes the paper.

2. Related work

2.1. Overview of CU partition in VVC

In contrast to HEVC which uses the quad-tree-based CU partition structure, VVC adopts a more flexible technique, i.e., the QTMT structure, which has a great contribution to the coding performance. An example of the QTMT structure is shown in Fig. 1. Specifically, the beginning of the CU partition in the QTMT structure is the largest coding unit or coding tree unit (CTU) in 128 × 128. It can have a single CU or be split into four square sub-CUs in 64 × 64 (non-splitting is not supported for CTUs in the intra-VVC). Then a CU of size 64 × 64 can be non-splitting or further split into four square sub-CUs in 32 × 32. A CU has a maximum of six partition modes (i.e., non-splitting, quad-tree, horizontal binary-tree, vertical binary-tree, horizontal ternary-tree and vertical ternary-tree, as shown in Fig. 2) for the subsequent partition stages, satisfying the minimum width or height of CUs is 4. The rate distortion cost (RDcost) is calculated for each partition mode to determine the optimal one with the lowest RDcost. However, since the size of CUs varies from 128 × 128 to 4 × 4 in VVC, there should be 5781 CUs to be checked in the QTMT structure (compared to only 81 CUs in HEVC) [16], which significantly increases the computational complexity of VVC. To expedite the encoding process of VVC, a large number of fast CU partition methods have been proposed.

2.2. Existing fast CU partition methods

2.2.1. Traditional data analysis-based methods

Fan et al. [8] proposed a fast CU partition method for VVC intra coding based on the variance and Sobel operator. If the variance of a CU fell below a certain threshold, the CU would be regarded as smooth, and further splitting would be early terminated. Then the Sobel operator extracted gradient features to determine whether to skip the multi-type tree partitions. Finally, only one partition mode was selected by calculating the variance of variance. Chen et al. [9] proposed a fast intra mode decision method for VVC. To shorten the candidate mode list, the modes selected in the rough modes decision (RMD) process were examined. The early termination of rate distortion optimization (RDO) process was then performed by computing the correlation between the RMD cost and RDcost. Wieckowski et al. [10] described multiple strategies for accelerating the CU partition process in the QTMT structure in VVC. By combining different acceleration

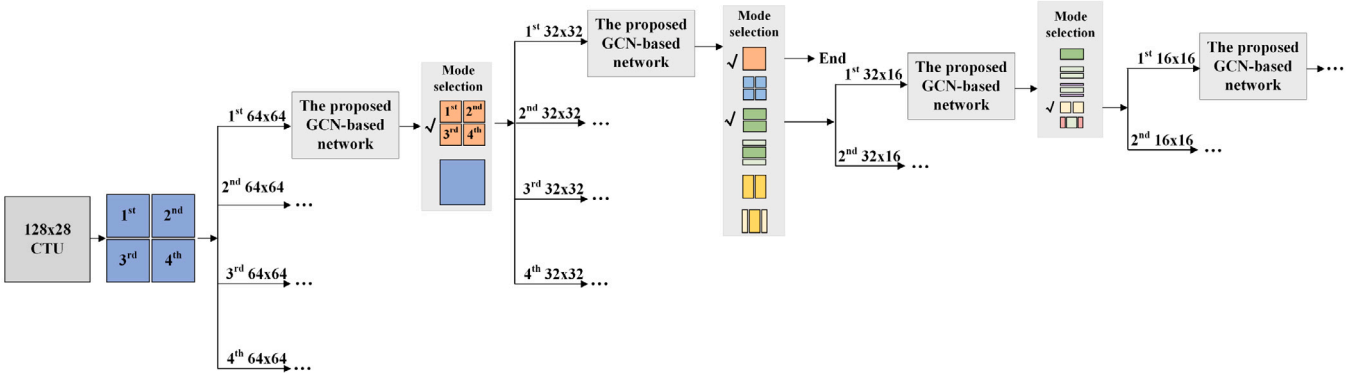

Fig. 3. Overall process of CU partition of intra-mode VVC with the proposed method.

strategies, their proposed method could achieve the tradeoff between coding performance and computational complexity. Tang et al. [11] described a fast QTMT partition method for both intra and inter coding in VVC. For the intra-mode VVC, the Canny detector was used to capture the information of CU edges to skip horizontal or vertical partition modes. And for the inter-mode VVC, the three-frame difference method was used to screen stationary CUs where further split could be early terminated. Peng et al. [12] developed a fast CU partition method of the intra-mode VVC. CUs were classified into three classes, i.e., simple CUs, common CUs and complex CUs, based on texture features and the preset thresholds. Then, to lessen the computational cost of VVC, partition and non-partition modes were omitted for simple and complicated CUs, respectively. Zhang et al. [13] used the gray level co-occurrence matrix (GLCM) to capture the texture information of CUs, which helps to early terminate the horizontal/vertical binary/ternary-tree partition. Li et al. [14] estimated the rate and distortion using the information provided by the reference block before encoding. The RDcosts of CU partition modes were predicted, making it possible to choose the most appropriate mode. Yang et al. [15] proposed a novel fast QTMT partition decision framework and a fast intra mode decision framework for the intra-mode VVC based on the statistical learning and gradient descent after a thorough exploration of the new block size and coding mode distribution properties.

2.2.2. DL-based methods

Li et al. [16] established a large-scale database with videos containing different contents and proposed a multi-stage exit convolutional neural network (MSE-CNN) with an early termination strategy. Guided by an adaptive loss function, the MSE-CNN model could predict the CU partition modes for accelerating the intra-VVC encoding. Tissier et al. [17] designed a convolutional neural network (CNN) with max-pooling and fully-connected (FC) layers to predict probabilities of partition modes of CUs in 64 × 64 to decrease the intra coding complexity of VVC. Tech et al. [18] created a training dataset by experimenting with all possible combinations of the two partition parameters which limited the width and height of CUs and recording the resulting rate–distortion-time (RDT) cost. Then, by training a CNN minimizing the RDT cost, their proposed method could simplify the partition process of CUs in 32 × 32 of the intra-mode VVC. Further developing the work in [18], Tech et al. [19] explored if and how adding more parameters could improve the RDT performance. Then, for fast intra-mode VVC encoding, they proposed a method with extra parameters to control the CU size from which the multi-type tree could start. Tang et al. [20] proposed a fast CU partition method based on the pooling-variable CNN. Their proposed method could adjust to different CU sizes by varying the size of the pooling layer. It indicated that CUs of various sizes could be handled by a single trained model, reducing the storage and transmission cost of model parameters. HoangVan et al. [22] utilized the early-terminated hierarchical CNN proposed in [21] to square CUs and the quad-tree structure in the intra-mode VVC. The intra encoding time of VVC was reduced by early terminating the CU partition process. Fu et al. [23] proposed a two-phase method to simplify the QTMT structure of CU partition of the intra-mode VVC. A multi-branch CNN used extracted latent features to predict the quad-tree depth and whether or not to use ternary-tree for CUs in the first phase. In the second phase, a number of partition modes were then pruned to achieve complexity reduction. Zhang et al. [24] proposed a CU partition speeding-up method for VVC intra coding based on the DenseNet proposed in [29]. The DenseNet was used to predict the probabilities that the edges of blocks in 4 × 4 were the partition boundaries of CUs in 64 × 64. The QTMT structure was then made simpler by excluding unnecessary partition modes. Tech et al. [25] trained a CNN to predict a pair of parameters including the minimum width and height of the sub-blocks of CUs in 32 × 32. In this way, a pruned QTMT was acquired to reduce the intra encoding time of VVC. Inspired by the U-net [30]. Park et al. [26] employed a lightweight neural network (LNN) to remove the redundant CU partition modes from the QTMT structure of the intra-mode VVC. Specifically, two types of features extracted during the intra prediction of VVC were fed into the LNN to determine whether horizontal/vertical ternary-tree should be skipped. Tsang et al. [27] proposed a fast CU partition mode prediction network to provide labels on the basis of which CUs and all associated sub-CUs were classified. Depending on the type of CU, different partition modes were skipped to reduce the computational complexity.

3. The proposed fast CU partition method

A GCN-based fast CU partition method that fully utilizes global information in CUs is proposed to achieve accurate CU segmentation and assist in choosing the best CU partition mode. This will help remove redundant CU partition modes from the QTMT structure and shorten the encoding time of the intra-mode VVC. The proposed method is only applied to the CU luminance channel. This section first introduces the overall process of CU partition in the intra-mode VVC using the proposed method, followed by a description of the architecture of the proposed GCN-based network, which accepts a CU as an input and predicts the probabilities of its possible partition modes.

3.1. Overall CU partition process with the proposed method

The overall process of CU partition in the intra-mode VVC using the proposed method is shown in Fig. 3. Since a CTU in 128 × 128 must be split into four sub-CUs in 64 × 64 in the intra-mode VVC, the proposed GCN-based network is first applied to CUs in 64 × 64.

Specifically, as shown in Fig. 3, taking the first CU in 64 × 64 as an example, it is fed into the proposed network to predict its best partition mode. For CUs in 64 × 64 in the intra-mode VVC, there

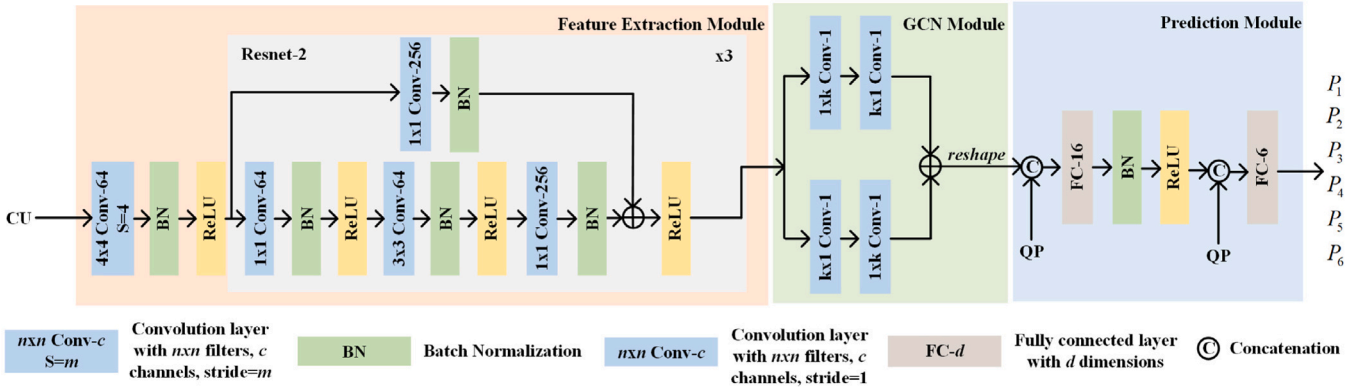

Fig. 4. Architecture of the proposed GCN-based network.

are only two possible partition modes (i.e., non-splitting and quad-tree), thus we only select one with higher probability and immediately discard the other. If the quad-tree mode is selected, four sub-CUs in 32×32 will be produced and further fed into the proposed GCN-based network. Otherwise, the partition process of the CU in 64×64 will be early terminated because the non-splitting mode is selected. For CUs in 32×32, there are six possible partition modes in the intra-mode VVC. We select the top two modes with higher probabilities predicted by the proposed GCN-based network and discard the other redundant ones. If the non-splitting and horizontal binary-tree modes are selected, one option will be to immediately terminate the partition process while the other will be to continue feeding the produced rectangular CUs in 32×16 into the proposed network to predict their partition modes. In contrast to CUs in 64×64 and 32×32, we establish a threshold to determine how many partition modes should be selected for each size of rectangular CUs (detailed in Section 4). If the highest probability exceeds the preset threshold, only the top one mode with the highest probability will be selected, otherwise top two modes will be selected. For instance, if the predicted probability of the vertical binary-tree mode is the highest and also exceeds the preset threshold, it will be selected to continue the partition process, and the other modes will be immediately discarded. Then the produced sub-CUs in 16×16 are fed into the proposed GCN-based network... The CU partition continues until the minimum width or height is 4 for CUs.

After discarding many redundant CU partition modes in the QTMT structure, we only calculate the RDcosts of the rest of partition modes. The mode with the smallest RDcost is finally selected as the best one. In this way, the computational complexity of the intra-mode VVC is significantly reduced.

Additionally, we can achieve tradeoffs between the coding performance and computational complexity for the intra-mode VVC by designing different strategies. Specifically, we have two options: either save more encoding time at the cost of poorer coding performance, or save less encoding time while attempting to keep the coding performance. The proposed method can be used in a variety of real-world applications, depending on the preferences of the users.

3.2. The proposed GCN-based network

The architecture of the proposed GCN-based network is shown in Fig. 4. It can be divided into three modules, i.e., the feature extraction module, the GCN module and the prediction module.

Specifically, as shown in Fig. 4, a (square or rectangular) CU is fed into the feature extraction module to extract its features. We intentionally design the kernel size and the stride of the first convolution to be 4×4 and 4, respectively, translating from a block in 4×4 in the CU to a pixel in the resulting feature map. This allows us to think of a block in 4×4 in the CU as a unit. It is consistent with the observation that the size of the smallest block that a CTU can be split into in the intra-mode VVC is 4×4. Then, to extract even more detailed feature maps, we employ Resnet-2, i.e., the second module of the pretrained ResNet-50 proposed in [31]. Resnet-2 is composed of three ResBlocks, the architecture of which is depicted in great detail in Fig. 4. With the skip connection, Resnet-2 is deep yet simple to train.

For the purpose of performing the classification and localization tasks concurrently for image segmentation, the GCN was proposed in [28]. The authors of [28] believe that the large kernel is crucial for image segmentation, where long-range pixel correlations and global information are the key to getting good performance. This is in contrast to one of the popular trends of network architecture design, which prefers to stack small filters in a deep way. To this end, the kernel size of the convolution layer (i.e., $k \times k$) in the GCN was designed to be basically the same as the resolution of the input feature maps. Furthermore, to reduce the computational complexity, the $k \times k$ convolution in the GCN was replaced by $1 \times k + k \times 1$ and $k \times 1 + 1 \times k$ convolutions without performance loss. Inspired by the success of the GCN in image segmentation, we use the GCN module to capture global pixel correlations in the feature maps output by the feature extraction module in the proposed GCN-based network. By acquiring the global receptive field in this manner, the global information of the CU can be fully utilized, paving the way for precise CU segmentation with the goal of determining the best CU partition mode. Note that the kernel size of the GCN module (i.e., k) varies with the size of the input CU because k is equal to the size of the feature maps (output by the feature extraction module) minus one. For instance, if the size of the input CU is 64×64, the size of feature maps will be 16×16, and k will be 15. And if the CU is rectangular, such as in 16×32, the stride of the first convolution in the feature extraction module will be (4,8) to ensure the size of the output feature maps is 4×4, and k will be 3. After adding the outputs of two branches in the GCN module, we reshape the results from $w \times h$ to $1 \times wh$ and then feed it into the prediction module.

The predicted probabilities of the available CU partition modes are output by the prediction module. To make the QTMT structure simpler and the computational complexity lower, we exclude some CU partition modes with lower probabilities after ranking the probabilities from large to small. The prediction module is mainly composed of two FC layers. Considering that the best partition modes of CUs vary with the values of quantization parameters (QPs), i.e., larger QP value corresponds to larger probability that a CU prefers the non-splitting mode and vice versa, we also feed the QP into the FC layers as an additional input. Through concatenation, the prediction module is modulated by QPs, thus being QP adaptive. Under this circumstance, we can train a single model to predict partition modes of CUs compressed at different QPs, reducing the storing and transmitting costs. Note that the number of the output nodes of the final FC layer matches the number of partition modes that the CU can support. For instance, if the size

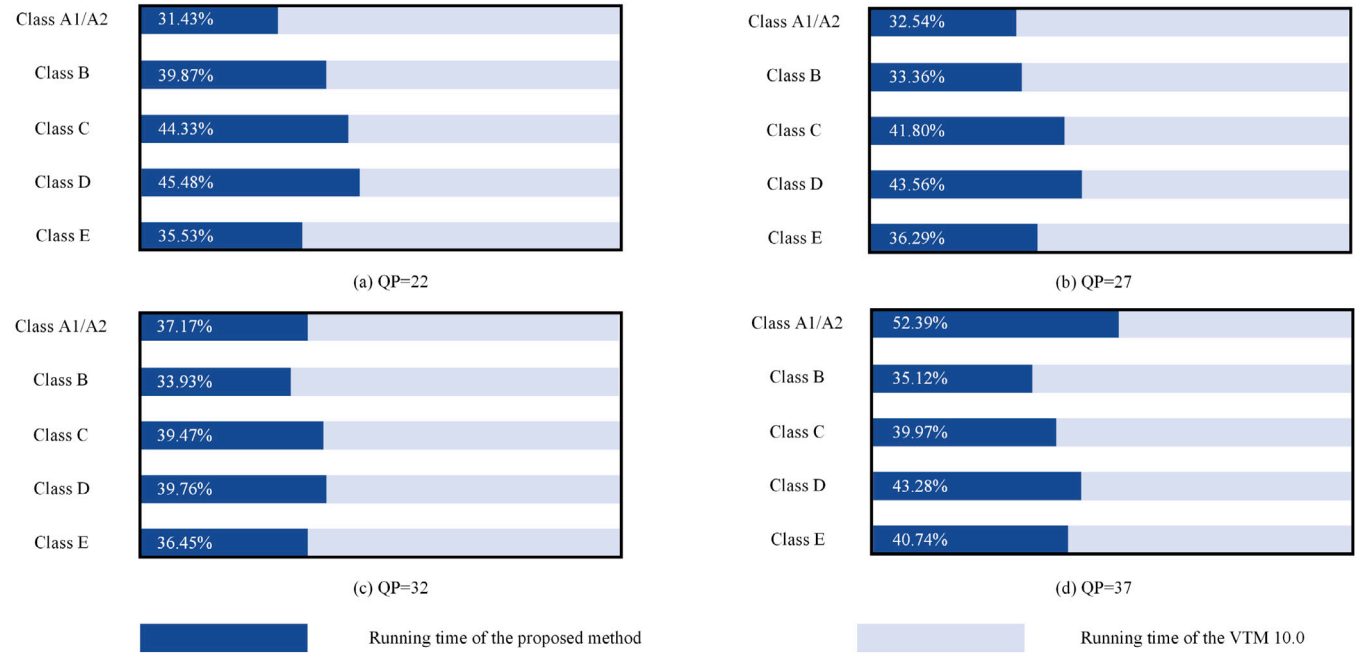

Fig. 5. Running time of the proposed method compared with that of the VTM 10.0.

of a CU is 64 × 64, the number of the output nodes of the final FC layer will be two because there are only two possible partition modes (i.e., non-splitting and quad-tree) for CUs in 64 × 64 in the intra-mode VVC. If the size of a CU is 32 × 32, the number of the output nodes will be six.

3.3. Loss function

The cross-entropy loss is frequently used in classification networks [32,33] because it can reduce the gap between the prediction and the ground-truth. Although we can think of the problem of determining the best partition mode for the CU as the challenge of CU classification, we cannot just use the cross-entropy loss in the proposed method because there are highly imbalanced proportions for various CU partition modes. For instance, in general, the probability of the vertical ternary-tree mode is lower than that of the vertical binary-tree mode. However, the cross-entropy loss completely ignores this kind of imbalance.

In the field of dense object detection, the focal loss [34] was proposed to address the problem of extreme foreground-background class imbalance by down-weighting the loss assigned to well-classified examples. The focused loss is essentially the cross-entropy loss that has had a modifying element added to it. Followed this idea, in this paper, we also use the focal loss to address the issue of imbalance of CU partition modes by giving greater weight to the CU partition mode with lower selection probability and vice versa. Furthermore, considering determining the best partition modes for CUs should be a multi-classification problem since there are a maximum of six CU partition modes, we extended the form of the focal loss proposed in Eq. (4) in [34] from the binary classification case to the multi-classification case. The loss function of the proposed network is designed as

$$L = -\frac{1}{N}\sum_{n=1}^{N}\sum_{m=0}^{M}(1-\hat{p}_{n,m})^{\gamma} p_{n,m}\log(\hat{p}_{n,m}), \tag{1}$$

where N denotes batch size, and M denotes the number of partition modes. $p_{n,m}$ and $\hat{p}_{n,m}$ denote the ground-truth of the probability and the output of the proposed network (i.e., the predicted probability), respectively. γ is the focusing parameter which is used to adjust the weight, set to 2 empirically.

4. Experimental results

4.1. Datasets

We used the DIV2K dataset [35] which consists of 800 high-resolution images for training and 22 video sequences from *class A* to *class E* in the JVET test set for test. All images in the training dataset were encoded by the VTM 10.0 at four QPs, i.e., 22, 27, 32 and 37 in the all-intra (AI) mode. Since the proposed method is only designed for the luminance component of CU (extending it to the chrominance component will be the future work), a training sample is composed of luminance pixel values, the best luminance partition mode, QP and RDcost of the CU. During the encoding process, RDcost of each CU partition mode is calculated, and the mode with the minimum RDcost is selected as the best one. However, when the difference of RDcosts between two partition modes is negligible, selecting either of the two modes as the best one has little impact on the coding performance. It will be quite challenging for us to succeed if we have to train a network to distinguish between these two partition modes. To this end, the training samples with $\Delta RD \leq 0.005$ were removed from the training dataset. ΔRD is defined as:

$$\Delta RD = \frac{RD_{op} - RD_{sub}}{RD_{op} + RD_{sub}}, \tag{2}$$

where RD_{op} and RD_{sub} are RDcosts of the optimal and the sub-optimal partition mode, respectively. Note that we used labels (i.e., 0, 1, 2, 3, 4 and 5) to represent non-splitting, quad-tree, horizontal binary-tree, vertical binary-tree, horizontal ternary-tree and vertical ternary-tree partition mode, respectively, in datasets.

4.2. Implementation details

Model training was conducted on a computer with GeForce RTX 2070 SUPER GPU. And evaluation experiments were conducted on a computer with an Intel i7-9700KF CPU @3.60 GHz, 16 GB RAM. We trained a model independently for each size of CU. In total, 10 independent models were trained for CUs in 64 × 64, 32 × 32, 16 × 16, 8 × 8, 32 × 16, 16 × 32, 32 × 8, 8 × 32, 16 × 8 and 8 × 16, respectively. Each model was trained 300k iterations with the batch size of 128 and

Table 2
Coding performance and time saving comparison to the state-of-the-art methods.

Class	Sequence	Method	BD-BR(%)	ΔT (%)				Class	Sequence	Method	BD-BR(%)	ΔT (%)			
				QP=22	QP=27	QP=32	QP=37					QP=22	QP=27	QP=32	QP=37
A1	*Campfire*	[15]	2.88	51.80	57.84	35.33	46.40	*C*	*BasketballDrill*	[15]	4.60	58.17	62.68	63.21	53.45
		[16]	2.91	57.62	60.56	61.21	60.07			[16]	2.99	53.47	53.11	53.23	50.67
		[21]	1.66	35.69	28.64	40.36	35.59			[21]	1.74	41.90	45.66	44.42	41.05
		Proposed	1.68	64.97	67.44	65.58	52.79			Proposed	2.16	50.76	56.70	61.42	57.98
	FoodMarket4	[15]	3.45	60.36	67.47	41.65	51.72		*BQMall*	[15]	4.98	61.97	67.25	64.64	59.23
		[16]	1.95	61.34	55.78	51.27	40.70			[16]	2.05	63.61	61.43	58.95	59.09
		[21]	2.25	35.96	27.32	33.50	34.31			[21]	1.14	43.95	43.91	43.26	40.10
		Proposed	1.43	69.88	64.36	60.09	42.51			Proposed	1.62	58.84	60.53	62.32	62.17
	Tango2	[15]	6.85	51.42	72.07	22.70	29.24		*PartyScene*	[15]	2.24	63.59	66.90	67.57	52.39
		[16]	2.33	64.46	60.72	47.69	25.52			[16]	1.16	57.69	57.88	55.56	62.50
		[21]	2.60	34.14	34.69	38.63	36.28			[21]	0.78	44.89	45.95	47.78	48.00
		Proposed	1.84	81.01	75.69	62.89	27.12			Proposed	0.76	53.82	55.36	56.61	58.24
A2	*CatRobot1*	[15]	5.27	53.00	62.05	40.30	40.55		*RaceHorses*	[15]	2.75	57.22	60.90	54.99	51.40
		[16]	3.28	61.29	57.10	54.06	51.51			[16]	1.61	57.69	57.88	55.56	54.63
		[21]	1.48	32.25	20.74	36.13	33.63			[21]	1.00	44.81	41.90	45.77	44.66
		Proposed	1.93	64.74	65.53	60.09	42.51			Proposed	1.03	59.26	60.22	61.76	61.72
	DaylightRoad2	[15]	7.97	59.76	71.35	56.03	64.67	*D*	*BasketballPass*	[15]	2.91	51.80	55.80	50.54	44.78
		[16]	1.95	65.72	61.27	56.79	53.68			[16]	2.35	58.36	57.76	55.07	49.63
		[21]	1.21	40.94	30.84	23.12	27.92			[21]	0.90	41.25	41.03	41.44	39.32
		Proposed	2.83	71.12	69.80	67.71	59.10			Proposed	1.37	57.52	58.67	58.95	55.31
	ParkRunning3	[15]	3.17	50.18	57.20	36.12	52.14		*BlowingBubbles*	[15]	1.80	59.73	59.55	60.04	44.02
		[16]	1.76	49.84	49.99	53.01	62.27			[16]	1.57	52.88	55.30	52.45	52.97
		[21]	0.92	23.22	24.65	33.86	34.51			[21]	0.61	42.17	42.64	43.62	47.47
		Proposed	0.86	59.67	61.96	60.64	61.63			Proposed	0.71	51.14	52.06	53.61	55.80
B	*MarketPlace*	[15]	3.29	51.66	68.29	43.56	63.12		*BQSquare*	[15]	1.99	55.97	60.21	62.96	42.92
		[16]	1.28	55.99	56.13	58.72	62.03			[16]	1.33	53.25	53.89	57.45	56.05
		[21]	1.17	23.14	35.44	34.33	40.12			[21]	1.40	53.52	56.68	59.81	63.04
		Proposed	1.13	67.43	71.65	75.12	72.57			Proposed	0.86	54.00	57.60	59.04	57.44
	RitualDance	[15]	4.05	60.39	70.14	49.14	58.13		*RaceHorses*	[15]	2.19	53.50	54.17	50.90	43.34
		[16]	1.80	59.06	57.74	55.67	54.54			[16]	1.88	53.50	52.70	53.01	54.01
		[21]	1.55	16.47	24.19	41.36	38.64			[21]	0.86	41.61	42.19	43.71	43.11
		Proposed	1.58	64.39	65.88	66.04	59.32			Proposed	0.88	55.40	57.43	59.35	58.33
	BasketballDrive	[15]	5.92	56.33	70.28	55.24	59.31	*E*	*FourPeople*	[15]	5.94	66.74	71.65	68.51	62.48
		[16]	2.61	64.48	65.93	60.62	56.95			[16]	2.2	64.67	59.62	56.84	57.81
		[21]	1.60	37.79	39.95	41.65	36.07			[21]	1.34	44.88	44.41	40.76	36.78
		Proposed	1.66	66.00	68.90	59.63	65.54			Proposed	1.87	61.84	62.21	63.62	62.28
	BQTerrace	[15]	4.38	48.89	70.15	67.68	63.55		*Johnny*	[15]	6.60	61.68	62.15	62.48	57.19
		[16]	1.79	43.37	62.10	60.64	61.66			[16]	3.57	64.04	61.21	56.75	49.53
		[21]	1.50	22.34	48.56	49.64	50.20			[21]	2.64	49.99	43.49	46.12	43.77
		Proposed	1.73	39.77	62.56	63.61	63.14			Proposed	2.01	66.69	65.78	64.60	58.46
	Cactus	[15]	4.56	57.75	67.39	61.31	63.65		*KristenAndSara*	[15]	5.41	62.53	62.12	61.74	51.83
		[16]	1.86	61.80	60.56	59.14	60.75			[16]	2.74	66.66	61.78	58.32	53.26
		[21]	1.30	41.36	41.55	43.73	40.13			[21]	2.23	47.86	43.79	46.29	46.29
		Proposed	1.53	63.04	64.23	65.94	63.85			Proposed	2.03	64.87	63.13	62.42	57.05
Average		[15]	4.24	57.02	64.44	53.48	52.52	*Average*		[16]	2.14	58.73	58.13	55.99	54.02
		[21]	1.45	38.19	38.56	41.79	40.95			Proposed	1.52	61.19	63.08	63.03	57.31

the learning rate set to 10^{-4}. Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ was adopted during the training.

The proposed method was implemented on the VTM 10.0 and VVenC 1.0 at the preset *slower*. Two metrics, i.e., ΔT and Bjøntegaard delta bit-rate (BD-BR), were used to evaluate the performance of the proposed method. ΔT indicated encoding time saving and computational costs reduction. It is calculated as:

$$\Delta T = \frac{T_{orig} - T_{prop}}{T_{orig}} \times 100\%, \tag{3}$$

where T_{orig} is the encoding time of the original encoder, and T_{prop} is the encoding time after using the proposed method. BD-BR indicated the encoding performance.

4.3. Performance evaluation

4.3.1. Implemented on VTM 10.0

When implemented on the VTM 10.0, we use the proposed method to fast predict the best partition mode of CUs in 64 × 64, 32 × 32, 16 × 16, 32 × 16 and 16 × 32. Only the top mode with the highest predicted probability is selected for CUs in 64 × 64, and all the other modes are immediately discarded. The top two modes are selected for CUs in 32 × 32 and 16 × 16 before the encoder selecting the best one through the RDO process. If the estimated probability of the top one mode exceeds the threshold, which is 0.5, only the top one mode will be selected for CUs in 32 × 16 and 16 × 32. Otherwise, the top two modes will be first selected, and the encoder will then select the best one through the RDO process. As mentioned before, the proposed method is designed for the luminance component. But we shorten the chroma candidate partition mode list by two to further accelerate the encoding process with negligible performance loss.

We compare the running time of the proposed method with that of the VTM 10.0 at four different QPs, as shown in Fig. 5. For video sequences of all classes encoded at all QPs, the proposed method is significantly faster and can reduce encoding time by more than a half.

In addition, we compare the proposed method to other state-of-the-art methods proposed in [15,16,21] in terms of both computational complexity (measured by ΔT) and coding performance (measured by

Table 3
Coding performance and average time saving (under QP 22, 27, 32 and 27) comparison to the state-of-the-art methods.

Class	Sequence	Method	BD-BR (%)	Average ΔT (%)
A1	*Campfire*	[24]	2.21	71.34
		[26]	0.29	29.00
		Proposed	1.68	62.70
	FoodMarket4	[24]	2.10	77.05
		[26]	0.37	33.00
		Proposed	1.43	59.21
	Tango2	[24]	2.41	76.53
		[26]	0.35	35.00
		Proposed	1.84	61.68
A2	*CatRobot1*	[24]	2.85	75.11
		[26]	0.49	29.00
		Proposed	1.93	58.22
	DaylightRoad2	[24]	1.93	77.78
		[26]	0.66	29.00
		Proposed	2.83	66.93
	ParkRunning3	[24]	0.85	69.85
		[26]	0.21	30.00
		Proposed	0.86	60.98
B	*MarketPlace*	[24]	1.49	81.10
		[26]	0.32	33.00
		Proposed	1.13	71.69
	RitualDance	[24]	2.30	76.06
		[26]	0.47	29.00
		Proposed	1.58	63.91
	BasketballDrive	[24]	2.07	62.73
		[26]	0.61	28.00
		Proposed	1.66	65.02
	BQTerrace	[24]	1.52	51.61
		[26]	0.49	23.00
		Proposed	1.73	57.27
	Cactus	[24]	1.93	58.98
		[26]	0.48	27.00
		Proposed	1.53	64.27
C	*BasketballDrill*	[24]	2.28	34.40
		[26]	0.68	25.00
		Proposed	2.16	56.72
	BQMall	[24]	1.58	36.82
		[26]	0.60	23.00
		Proposed	1.62	60.97
	PartyScene	[24]	0.76	27.34
		[26]	0.34	25.00
		Proposed	0.76	56.01
	RaceHorses	[24]	1.02	39.39
		[26]	0.31	26.00
		Proposed	1.03	60.74
D	*BasketballPass*	[24]	0.98	23.50
		[26]	0.43	25.00
		Proposed	1.37	57.61
	BlowingBubbles	[24]	0.42	16.60
		[26]	0.23	22.00
		Proposed	0.71	53.15
	BQSquare	[24]	0.41	17.94
		[26]	0.32	24.00
		Proposed	0.86	57.02
	RaceHorses	[24]	0.66	23.21
		[26]	0.29	24.00
		Proposed	0.88	57.63
E	*FourPeople*	[24]	2.69	54.98
		[26]	0.69	24.00
		Proposed	1.87	62.49
	Johnny	[24]	3.32	55.54
		[26]	0.62	26.00
		Proposed	2.01	63.88
	KristenAndSara	[24]	2.42	50.79
		[26]	0.53	25.00
		Proposed	2.03	61.87
	Average	[24]	1.74	52.67
		[26]	0.44	27.00
		Proposed	1.52	61.15

(*continued on next page*)

BD-BR). Note that three configurations, i.e., "*medium*", "*fast*" and "*faster*", were set in [16], and we only make comparison to the configuration "*fast*" which achieved the complexity-performance tradeoff point closest to the proposed method. The DL based approach [21] was originally designed for HEVC complexity reduction, but we transferred it to the VVC standard for comparison. The coding performance and time saving comparison is shown in Table 2. Overall, the proposed method outperforms the other state-of-the-art methods in complexity-performance tradeoff. It can significantly reduce the computational complexity with tolerable coding performance degradation. For instance, for the sequence *Campfire*, although the method proposed in [21] achieves the minimum increase of BD-BR (i.e., 1.66%, indicating the best coding performance), the time saving only varies from 28.64% to 40.36% under different QPs. However, the proposed method can reduce encoding time from 52.79% to 67.44% with a 1.68% increase in BD-BR. From the view of complexity-performance tradeoff, our proposed method is better than that proposed in [21]. Additionally, the methods proposed in [15,16] are inferior to our proposed method in terms of both computational complexity and coding performance when tested on the sequence *Campfire*. For the sequence *BQSquare*, the method proposed in [16] achieves the maximum time saving at the cost of a 1.99% BD-BR increase. However, our proposed method achieves a similar time saving while only increasing the BD-BR by 0.86%. For the average performance of 22 tested video sequences, the proposed method nearly achieves the least BD-BR increase and the largest time saving when compared to the other methods, indicating the advanced performance of the proposed method.

Furthermore, we also compare our proposed method to the methods proposed in [24,26]. Considering only the results of average coding time saving under four QPs (i.e. QP 22, 27 32 and 37) were made public in [24,26], we also calculated the average coding time saving of our proposed method in the same way for fair comparison. The performance comparison is shown in Table 3.

From the average performance of 22 tested video sequences, we can draw the conclusion that our proposed method has outperformed the method proposed in [24] since our proposed method saves more coding time but increases less BD-BR. As for the method proposed in [26], it achieves less BD-BR increasing and also less coding time saving in comparison to the proposed method. It is difficult to determine either the proposed method or the method proposed in [26] is better. However, we argue that only a little amount of coding time is saved with the method proposed in [26], which makes it challenging to use it in real-world applications with strict encoding speed requirements.

4.3.2. Implemented on VVenC 1.0

To further validate the performance of the proposed method, the proposed method is implemented on VVenC 1.0 at the preset *slower* to accelerate its encoding process. We compare the computational complexity and coding performance of the accelerated VVenC at the preset *slower* with the original VVenC at the presets *slow* and *medium* to illustrate the superiority of the proposed method.

Specifically, when compared to the VVenC 1.0 at the preset *slow*, we use the proposed method to predict partition modes of CUs in 64×64, 32×32, 16×16, 32×16 and 16×32. The partition modes selection strategy is the same as that detailed in the Section 4.3.1. When

Table 4
Coding performance and time saving comparison with VVenC at the presets *slow* and *medium*.

Class	Sequence	Preset	BD-BR(%)	ΔT (%)				Class	Sequence	Preset	BD-BR(%)	ΔT (%)			
				QP=22	QP=27	QP=32	QP=37					QP=22	QP=27	QP=32	QP=37
A1	*Campfire*	*slow*	0.49	16.76	25.78	24.81	−1.95	*C*	*BasketballDrill*	*slow*	−0.56	−6.05	7.26	14.29	0.03
		medium	−0.29	20.70	24.85	14.92	−21.46			*medium*	−1.60	−5.67	−0.78	7.24	−5.60
	FoodMarket4	*slow*	0.20	35.56	19.61	0.71	−58.68		*BQMall*	*slow*	−0.34	13.03	12.89	13.09	8.34
		medium	1.03	23.87	18.24	14.12	−10.22			*medium*	−0.92	−4.57	−4.57	2.77	1.31
	Tango2	*slow*	0.87	50.65	35.45	−8.40	−118.39		*PartyScene*	*slow*	−0.60	1.19	1.18	−0.36	−0.85
		medium	0.33	44.59	28.35	7.35	−59.09			*medium*	−0.68	−10.49	−7.51	−3.11	1.16
A2	*CatRobot1*	*slow*	−0.09	28.21	26.38	20.07	−9.32		*RaceHorses*	*slow*	−0.15	13.78	13.19	10.98	7.30
		medium	−0.50	15.03	11.24	6.69	−12.46			*medium*	−0.27	5.19	4.76	7.57	5.10
	DaylightRoad2	*slow*	1.22	31.28	30.75	22.72	−7.52	*D*	*BasketballPass*	*slow*	−0.36	5.75	2.83	−1.25	−15.40
		medium	0.49	32.07	28.11	23.32	6.74			*medium*	−0.61	−18.04	−18.68	−22.02	−43.98
	ParkRunning3	*slow*	−0.17	28.22	29.93	30.69	21.66		*BlowingBubbles*	*slow*	−0.29	−3.87	−6.92	−6.84	−6.96
		medium	0.35	−3.83	2.00	9.16	0.67			*medium*	−0.52	−25.54	−25.41	−23.86	−28.13
B	*MarketPlace*	*slow*	0.06	32.66	37.08	41.12	27.69		*BQSquare*	*slow*	−0.70	−1.42	0.80	−3.59	−12.08
		medium	0.40	21.49	30.78	38.86	32.20			*medium*	−0.56	−10.01	−4.29	−4.61	−13.78
	RitualDance	*slow*	0.24	24.32	22.38	19.24	−3.68		*RaceHorses*	*slow*	−0.09	4.77	5.23	3.47	−5.46
		medium	0.26	12.88	16.19	21.41	13.95			*medium*	−0.28	−6.58	−9.66	−10.47	−26.14
	BasketballDrive	*slow*	−0.25	26.14	28.89	24.91	1.37	*E*	*FourPeople*	*slow*	0.20	20.68	17.12	15.52	7.45
		medium	−0.97	19.09	22.20	19.97	1.76			*medium*	−0.53	2.96	8.29	14.42	11.99
	BQTerrace	*slow*	−0.40	−17.37	15.73	18.56	13.72		*Johnny*	*slow*	0.52	28.85	21.75	15.26	−5.67
		medium	−0.51	−1.30	18.60	24.16	22.89			*medium*	0.16	7.30	7.35	5.58	−7.17
	Cactus	*slow*	0.09	23.84	23.87	25.53	15.82		*KristenAndSara*	*slow*	0.02	27.32	17.63	12.47	−4.53
		medium	0.12	16.37	17.96	21.94	13.94			*medium*	−0.49	2.76	2.42	3.63	−9.68
	Average	*slow*	0.00	17.47	17.67	13.32	−6.69		*Average*	*medium*	−0.25	6.28	7.75	8.14	−5.99

Table 5
Tradeoff between coding performance and time saving of the proposed method.

Class	Sequence	Preset	BD-BR(%)	ΔT (%)				Class	Sequence	Preset	BD-BR(%)	ΔT (%)			
				QP=22	QP=27	QP=32	QP=37					QP=22	QP=27	QP=32	QP=37
A1	*Campfire*	*medium*	0.98	53.80	57.59	53.25	39.16	*C*	*BasketballDrill*	*medium*	1.33	42.79	48.32	51.81	46.80
		fast	1.68	64.97	67.44	65.58	52.79			*fast*	2.16	50.76	56.70	61.42	57.98
	FoodMarket4	*medium*	0.78	57.15	49.88	45.98	32.54		*BQMall*	*medium*	0.79	50.59	53.22	54.88	53.56
		fast	1.43	69.88	64.36	60.09	42.51			*fast*	1.62	58.84	60.53	62.32	62.17
	Tango2	*medium*	0.94	68.52	60.02	47.74	20.09		*PartyScene*	*medium*	0.43	46.28	49.09	51.16	51.72
		fast	1.84	81.01	75.69	62.89	27.12			*fast*	0.76	53.82	55.36	56.61	58.24
A2	*CatRobot1*	*medium*	1.09	52.80	51.88	50.59	41.23		*RaceHorses*	*medium*	0.57	46.46	48.78	51.34	49.86
		fast	1.93	64.74	65.53	60.09	42.51			*fast*	1.03	59.26	60.22	61.76	61.72
	DaylightRoad2	*medium*	1.85	62.41	59.10	55.85	46.87	*D*	*BasketballPass*	*medium*	0.71	48.44	49.72	49.87	46.17
		fast	2.83	71.12	69.80	67.71	59.10			*fast*	1.37	57.52	58.67	58.95	55.31
	ParkRunning3	*medium*	0.47	42.89	44.69	47.73	43.04		*BlowingBubbles*	*medium*	0.49	43.32	44.72	47.17	48.94
		fast	0.86	59.67	61.96	60.64	61.63			*fast*	0.71	51.14	52.06	53.61	55.80
B	*MarketPlace*	*medium*	0.58	54.68	58.37	60.21	56.35		*BQSquare*	*medium*	0.49	46.68	51.68	53.41	52.22
		fast	1.13	67.43	71.65	75.12	72.57			*fast*	0.86	54.00	57.60	59.04	57.44
	RitualDance	*medium*	0.85	53.70	54.72	54.91	48.90		*RaceHorses*	*medium*	0.57	46.46	48.78	51.34	49.86
		fast	1.58	64.39	65.88	66.04	59.32			*fast*	0.88	55.40	57.43	59.35	58.33
	BasketballDrive	*medium*	0.89	56.40	58.15	55.61	47.83	*E*	*FourPeople*	*medium*	1.02	52.40	54.15	55.28	53.20
		fast	1.66	66.00	68.90	59.63	65.54			*fast*	1.87	61.84	62.21	63.62	62.28
	BQTerrace	*medium*	1.02	29.32	55.78	55.82	54.70		*Johnny*	*medium*	1.13	58.12	56.97	54.83	47.70
		fast	1.73	39.77	62.56	63.61	63.14			*fast*	2.01	66.69	65.78	64.60	58.46
	Cactus	*medium*	0.79	53.24	54.52	56.15	52.69		*KristenAndSara*	*medium*	0.79	56.46	54.87	54.21	49.26
		fast	1.53	63.04	64.23	65.94	63.85			*fast*	2.03	64.87	63.13	62.42	57.05
	Average	*medium*	0.84	51.24	53.10	52.80	47.10		*Average*	*fast*	1.52	61.19	63.08	63.03	57.31

compared to the VVenC 1.0 at the preset *medium*, we use the proposed method to predict partition modes of CUs in 64 × 64, 32 × 32, 16 × 16, 8 × 8, 32 × 16, 16 × 32, 32 × 8, 8 × 32, 16 × 8 and 8 × 16.

Only the top mode with the highest predicted probability is selected for CUs in 64 × 64 while the top two modes are selected for CUs in 32 × 32, 16 × 16 and 8 × 8 before the encoder selecting the best one through the RDO process. If the estimated probability of the top one mode exceeds the threshold, which is 0.5 (or 0.65), only the top one mode will be selected for CUs in 32 × 16 and 16 × 32 (or 32 × 8, 8 × 32, 16 × 8 and 8 × 16). Otherwise, the top two modes will be first selected, and the encoder will then select the best one through the RDO process. We further shorten the chroma candidate partition mode list by two to reduce the computational complexity. Note that when compared to the VVenC 1.0 at the preset *medium*, we do not consider the inference time of the network, referred to *Section 5.1* in [36], because the CNN can be evaluated in parallel to the encoding.

The comparison results are shown in Table 4. Compared to the VVenC at the preset *slow*, the accelerated VVenC at the preset *slower* is faster while maintaining the same average coding performance (i.e., the average BD-BR is increased by 0%). It indicates that the proposed

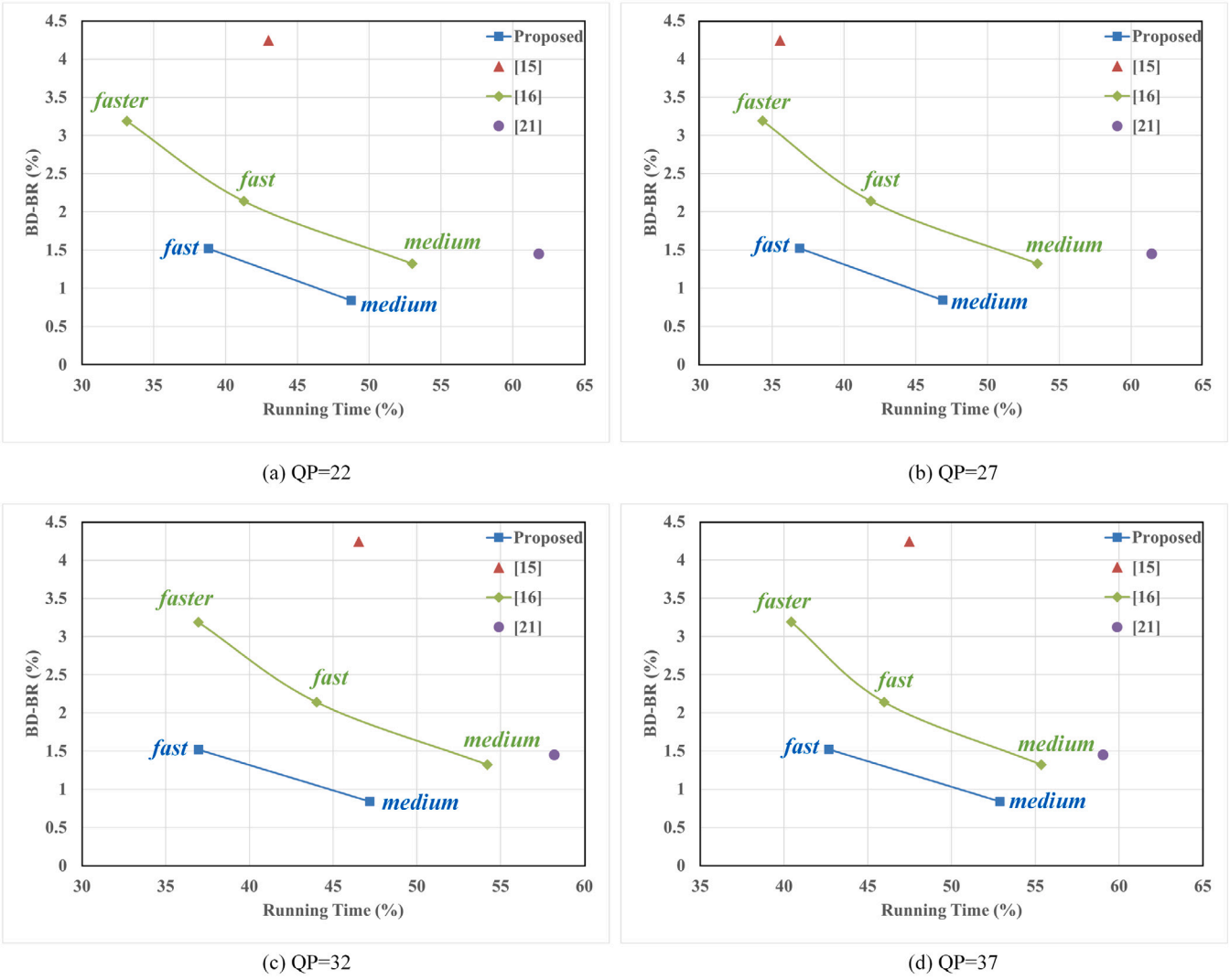

Fig. 6. Running time and performance comparison.

method provides a better tradeoff between the computational complexity and coding performance than that provided by the VVenC itself (from the preset *slower* to the preset *slow*). Compared to the VVenC at the preset *medium*, the proposed method even improves the average coding performance (i.e., the average BD-BR is decreased by 0.25%) with lower complexity.

4.3.3. Complexity-performance tradeoff of the proposed method

The proposed method itself can also achieve tradeoffs between the computational complexity and coding performance with different strategies. In this section, we implement the proposed method on the VTM 10.0 with two strategies (or called presets). The first one, referred to as "*fast*", has been introduced in the Section 4.3.1, and the second one is that we only predict the partition modes of CUs in 64 × 64, 32 × 32 and 16 × 16. We select top one mode for CUs in 64 × 64 and top two modes for CUs in 32 × 32 and 16 × 16, referred to as "*medium*". Performance comparison is shown in Table 5. The preset "*medium*" can achieve better coding performance but the preset "*fast*" can achieve faster encoding speed. Two presets should be appropriately selected by users according to their preferences or the requirements of real-world applications.

Additionally, as shown in Figs. 6 and 7, we compare the running time and coding performance of the proposed method with two presets (i.e., "*medium*" and "*fast*") to those of the other state-of-the-art methods

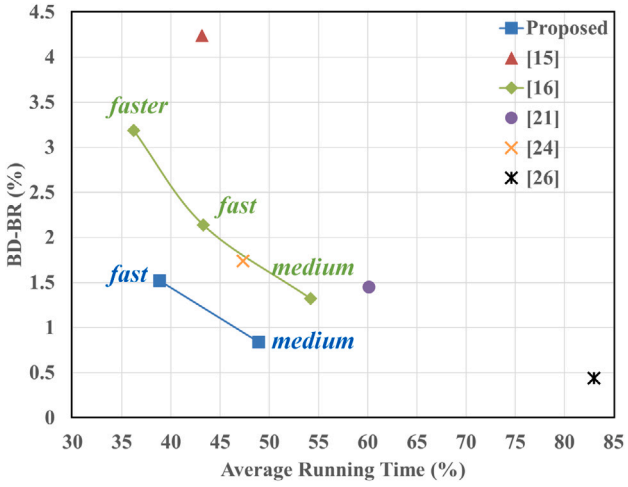

Fig. 7. Average running time and performance comparison under QP 22, 27, 32 and 37.

(the horizontal coordinates in Figs. 6 and 7 represent the running time under each QP and the average running time under QP 22, 27,

Table 6
Coding performance and time saving comparison in ablation studies.

Class	Sequence	Method	BD-BR(%)	ΔT (%)				Class	Sequence	Method	BD-BR(%)	ΔT (%)			
				QP=22	QP=27	QP=32	QP=37					QP=22	QP=27	QP=32	QP=37
A1	*Campfire*	w/o GCN	3.52	72.35	73.26	67.79	53.84	*C*	*BasketballDrill*	w/o GCN	9.05	62.97	67.30	68.71	63.27
		w/o Resnet	4.13	69.52	71.12	68.51	57.20			w/o Resnet	4.92	60.93	64.52	66.83	62.75
		Proposed	1.68	64.97	67.44	65.58	52.79			Proposed	2.16	50.76	56.70	61.42	57.98
	FoodMarket4	w/o GCN	2.12	74.91	67.23	61.09	43.18		*BQMall*	w/o GCN	5.20	67.17	68.41	69.56	68.17
		w/o Resnet	3.15	73.62	68.52	65.32	52.39			w/o Resnet	3.17	61.59	63.61	65.44	68.57
		Proposed	1.43	69.88	64.36	60.09	42.51			Proposed	1.62	58.84	60.53	62.32	62.17
	Tango2	w/o GCN	2.94	84.86	76.32	61.88	25.24		*PartyScene*	w/o GCN	4.00	62.20	64.64	66.55	68.12
		w/o Resnet	5.63	84.29	78.61	68.06	41.03			w/o Resnet	1.20	52.89	56.03	59.16	62.55
		Proposed	1.84	81.01	75.69	62.89	27.12			Proposed	0.76	53.82	55.36	56.61	58.24
A2	*CatRobot1*	w/o GCN	3.79	70.83	69.34	66.83	56.32		*RaceHorses*	w/o GCN	5.18	62.30	63.94	66.08	64.34
		w/o Resnet	5.32	68.49	70.13	68.66	60.43			w/o Resnet	1.49	55.25	58.78	60.25	59.05
		Proposed	1.93	64.74	65.53	60.09	42.51			Proposed	1.03	59.26	60.22	61.76	61.72
	DaylightRoad2	w/o GCN	6.68	78.10	75.46	71.72	61.62	*D*	*BasketballPass*	w/o GCN	5.92	65.28	66.33	65.76	61.17
		w/o Resnet	6.99	75.60	75.89	74.21	67.48			w/o Resnet	2.56	60.99	63.35	63.52	60.30
		Proposed	2.83	71.12	69.80	67.71	59.10			Proposed	1.37	57.52	58.67	58.95	55.31
	ParkRunning3	w/o GCN	1.39	64.77	65.09	66.32	61.44		*BlowingBubbles*	w/o GCN	3.08	58.19	60.07	62.30	63.65
		w/o Resnet	1.57	62.21	62.86	64.53	60.79			w/o Resnet	1.11	49.56	52.51	56.23	59.99
		Proposed	0.86	59.67	61.96	60.64	61.63			Proposed	0.71	51.14	52.06	53.61	55.80
B	*MarketPlace*	w/o GCN	1.92	74.30	76.59	77.60	73.35		*BQSquare*	w/o GCN	8.82	64.26	67.64	68.73	66.57
		w/o Resnet	2.39	71.70	73.94	74.93	72.10			w/o Resnet	1.49	55.25	58.78	60.25	59.05
		Proposed	1.13	67.43	71.65	75.12	72.57			Proposed	0.86	54.00	57.60	59.04	57.44
	RitualDance	w/o GCN	3.24	71.19	71.73	71.33	63.98		*RaceHorses*	w/o GCN	5.18	62.30	63.94	66.08	64.43
		w/o ResNet	3.62	69.28	70.22	70.78	65.41			w/o Resnet	1.81	55.57	59.00	62.19	61.55
		Proposed	1.58	64.39	65.88	66.04	59.32			Proposed	0.88	55.40	57.43	59.35	58.33
	BasketballDrive	w/o GCN	4.48	74.05	75.65	72.79	63.01	*E*	*FourPeople*	w/o GCN	4.83	69.61	69.97	70.38	68.19
		w/o Resnet	3.63	71.11	73.87	72.59	64.66			w/o Resnet	3.97	65.46	65.73	67.30	67.02
		Proposed	1.66	66.00	68.90	59.63	65.54			Proposed	1.87	61.84	62.21	63.62	62.28
	BQTerrace	w/o GCN	5.15	54.38	72.82	72.74	71.21		*Johnny*	w/o GCN	4.49	72.72	71.17	68.91	61.34
		w/o Resnet	3.12	46.22	67.25	67.60	67.20			w/o Resnet	4.22	70.02	69.30	68.12	62.45
		Proposed	1.73	39.77	62.56	63.61	63.14			Proposed	2.01	66.69	65.78	64.60	58.46
	Cactus	w/o GCN	4.06	71.04	71.97	72.58	68.82		*KristenAndSara*	w/o GCN	3.41	70.34	68.56	67.08	61.99
		w/o Resnet	2.91	66.00	68.09	69.40	66.65			w/o Resnet	1.89	68.23	66.19	65.01	61.48
		Proposed	1.53	63.04	64.23	65.94	63.85			Proposed	2.03	64.87	63.13	62.42	57.05
	Average	w/o GCN	4.42	68.77	69.59	68.44	61.67		*Average*	Proposed	1.52	61.19	63.08	63.03	57.31
		w/o Resnet	3.22	64.58	66.50	66.58	62.04								

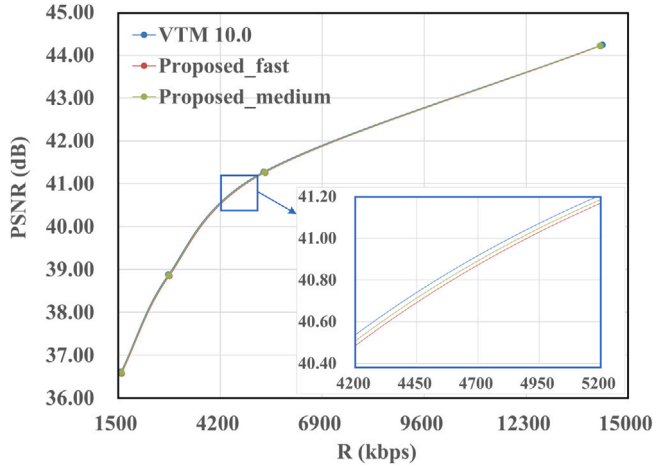

Fig. 8. Comparison of rate–distortion performance averaged on 22 tested video sequences.

32 and 37, respectively). As mentioned before, the method proposed in [16] has three presets, i.e., "*medium*", "*fast*" and "*faster*". For easier comparison, the performance points for each of these three presets are all shown in Figs. 6 and 7. As for the other methods, they were not designed with such presets. So, they only have a single performance point. Less running time corresponds to faster encoding speed, and less BD-BR increasing corresponds to better coding performance. To this end, the BD-BR-running time performance point of the best method should be located at the lower-left corner of the figure. Obviously, when compared to the other methods, the proposed method achieves advanced tradeoffs between coding performance and coding time.

Furthermore, we also show the rate–distortion curves of VTM 10.0 and the proposed method with two presets, i.e., "*medium*" and "*fast*" in Fig. 8. The rate (kbps) and PSNR (dB), which are averaged over all of the 22 tested video sequences, are used to assess the rate and distortion performance. As shown in Fig. 8, the rate–distortion curves of the proposed method with two different presets are almost overlapped with that of the VTM 10.0, but the proposed method can significantly reduce the coding time (by more than a half) of the VTM 10.0, which illustrates the superiority of the proposed method.

4.4. Ablation studies

Ablation studies are conducted to evaluate the effectiveness of components of the proposed network. One the one hand, the GCN module is removed from the proposed network, referred to as "w/o GCN". On the other hand, Resnet-2 in the feature extraction module is removed from the proposed network, referred to as "w/o Resnet". The coding performance and time saving of "w/o GCN" and "w/o Resnet" are independently tested to better illustrate their contributions. The experimental results are shown in Table 6 where the coding performance and time saving of the complete proposed network (i.e., "Proposed") is also shown for comparison.

Obviously, when the Resnet is removed, the coding performance is significantly penalized (i.e., average BD-rate increases from 1.52% to 3.22%). But the coding time is almost unchanged. When the GCN module is removed, the coding performance is even worse (i.e., average BD-rate increases to 4.42%) while the extra coding time saved can be almost ignored. It is proved that both the feature extraction module and the GCN module in the proposed network are effective and helpful for retaining the coding performance when accelerating the CU partition process of intra-mode VVC.

5. Conclusion

In this paper, we proposed a GCN-based fast CU partition mode decision method of intra-mode VVC for accelerating its encoding process. We implemented the proposed method both on VTM 10.0 and VVenC 1.0 to evaluate its performance. Based on the predicted probabilities of CU partition modes, top few modes were selected, and the best one was further determined by the encoder through the RDO process. Additionally, the proposed method could also achieve tradeoffs between the computational complexity and coding performance with different strategies. Experimental results have demonstrated the superiority of the proposed method when compared to the other state-of-the-art methods.

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.

Data availability

Data will be made available on request.

References

[1] M. Cheon, J.-S. Lee, Subjective and objective quality assessment of compressed 4K UHD videos for immersive experience, IEEE Trans. Circuits Syst. Video Technol. 28 (7) (2018) 1467–1480, http://dx.doi.org/10.1109/TCSVT.2017.2683504.

[2] K.-T. Ng, S.-C. Chan, H.-Y. Shum, Data compression and transmission aspects of panoramic videos, IEEE Trans. Circuits Syst. Video Technol. 15 (1) (2005) 82–95, http://dx.doi.org/10.1109/TCSVT.2004.839989.

[3] Z. Liu, C. Xu, M. Zhang, X. Guan, Fast intra prediction algorithm for virtual reality 360 degree video based on improved RMD, in: 2019 Data Compression Conference (DCC), 2019, p. 593, http://dx.doi.org/10.1109/DCC.2019.00105.

[4] 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) (2012) 1649–1668, http://dx.doi.org/10.1109/TCSVT.2012.2221191.

[5] B. Bross, Y.-K. Wang, Y. Ye, S. Liu, J. Chen, G.J. Sullivan, J.-R. Ohm, Overview of the versatile video coding (VVC) standard and its applications, IEEE Trans. Circuits Syst. Video Technol. 31 (10) (2021) 3736–3764, http://dx.doi.org/10.1109/TCSVT.2021.3101953.

[6] I. Siqueira, G. Correa, M. Grellert, Rate-distortion and complexity comparison of HEVC and VVC video encoders, in: 2020 IEEE 11th Latin American Symposium on Circuits and Systems (LASCAS), 2020, pp. 1–4, http://dx.doi.org/10.1109/LASCAS45839.2020.9069036.

[7] M. Saldanha, G. Sanchez, C. Marcon, L. Agostini, Performance analysis of VVC intra coding, J. Vis. Commun. Image Represent. 79 (2021) 103202, http://dx.doi.org/10.1016/j.jvcir.2021.103202.

[8] 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, http://dx.doi.org/10.1109/ACCESS.2020.3000565.

[9] Y. Chen, L. Yu, H. Wang, T. Li, S. Wang, A novel fast intra mode decision for versatile video coding, J. Vis. Commun. Image Represent. 71 (2020) 102849, http://dx.doi.org/10.1016/j.jvcir.2020.102849.

[10] A. Wieckowski, J. Ma, H. Schwarz, D. Marpe, T. Wiegand, Fast partitioning decision strategies for the upcoming versatile video coding (VVC) standard, in: 2019 IEEE International Conference on Image Processing (ICIP), 2019, pp. 4130–4134, http://dx.doi.org/10.1109/ICIP.2019.8803533.

[11] N. Tang, J. Cao, F. Liang, J. Wang, H. Liu, X. Wang, X. Du, Fast CTU partition decision algorithm for VVC intra and inter coding, in: 2019 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), 2019, pp. 361–364, http://dx.doi.org/10.1109/APCCAS47518.2019.8953076.

[12] S. Peng, Z. Peng, Y. Ren, F. Chen, Fast intra-frame coding algorithm for versatile video coding based on texture feature, in: 2019 IEEE International Conference on Real-Time Computing and Robotics (RCAR), 2019, pp. 65–68, http://dx.doi.org/10.1109/RCAR47638.2019.9044150.

[13] H. Zhang, L. Yu, T. Li, H. Wang, Fast GLCM-based intra block partition for VVC, in: 2021 Data Compression Conference (DCC), 2021, p. 382, http://dx.doi.org/10.1109/DCC50243.2021.00060.

[14] W. Li, C. Fan, P. Ren, Fast intra-picture partitioning for versatile video coding, in: 2020 IEEE 5th International Conference on Signal and Image Processing (ICSIP), 2020, pp. 108–111, http://dx.doi.org/10.1109/ICSIP49896.2020.9339262.

[15] H. Yang, L. Shen, X. Dong, Q. Ding, P. An, G. Jiang, Low-complexity CTU partition structure decision and fast intra mode decision for versatile video coding, IEEE Trans. Circuits Syst. Video Technol. 30 (6) (2020) 1668–1682, http://dx.doi.org/10.1109/TCSVT.2019.2904198.

[16] T. Li, M. Xu, R. Tang, Y. Chen, Q. Xing, Deepqtmt: A deep learning approach for fast QTMT-based CU partition of intra-mode VVC, IEEE Trans. Image Process. 30 (2021) 5377–5390, http://dx.doi.org/10.1109/TIP.2021.3083447.

[17] A. Tissier, W. Hamidouche, J. Vanne, F. Galpin, D. Menard, CNN oriented complexity reduction of VVC intra encoder, in: 2020 IEEE International Conference on Image Processing (ICIP), 2020, pp. 3139–3143, http://dx.doi.org/10.1109/ICIP40778.2020.9190797.

[18] G. Tech, J. Pfaff, H. Schwarz, P. Helle, A. Wieckowski, D. Marpe, T. Wiegand, Fast partitioning for VVC intra-picture encoding with a CNN minimizing the rate-distortion-time cost, in: 2021 Data Compression Conference (DCC), 2021, pp. 3–12, http://dx.doi.org/10.1109/DCC50243.2021.00008.

[19] G. Tech, J. Pfaff, H. Schwarz, P. Helle, A. Wieckowski, D. Marpe, T. Wiegand, CNN-based parameter selection for fast VVC intra-picture encoding, in: 2021 IEEE International Conference on Image Processing (ICIP), 2021, pp. 2109–2113, http://dx.doi.org/10.1109/ICIP42928.2021.9506360.

[20] G. Tang, M. Jing, X. Zeng, Y. Fan, Adaptive CU split decision with pooling-variable CNN for VVC intra encoding, in: 2019 IEEE Visual Communications and Image Processing (VCIP), 2019, pp. 1–4, http://dx.doi.org/10.1109/VCIP47243.2019.8965679.

[21] M. Xu, T. Li, Z. Wang, X. Deng, R. Yang, Z. Guan, Reducing complexity of HEVC: A deep learning approach, IEEE Trans. Image Process. 27 (10) (2018) 5044–5059, http://dx.doi.org/10.1109/TIP.2018.2847035.

[22] X. HoangVan, S. NguyenQuang, M. DinhBao, M. DoNgoc, D. Trieu Duong, Fast QTMT for H.266/VVC intra prediction using early-terminated hierarchical CNN model, in: 2021 International Conference on Advanced Technologies for Communications (ATC), 2021, pp. 195–200, http://dx.doi.org/10.1109/ATC52653.2021.9598222.

[23] P.-C. Fu, C.-C. Yen, N.-C. Yang, J.-S. Wang, Two-phase scheme for trimming QTMT CU partition using multi-branch convolutional neural networks, in: 2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS), 2021, pp. 1–6, http://dx.doi.org/10.1109/AICAS51828.2021.9458479.

[24] Q. Zhang, R. Guo, B. Jiang, R. Su, Fast CU decision-making algorithm based on DenseNet network for VVC, IEEE Access 9 (2021) 119289–119297, http://dx.doi.org/10.1109/ACCESS.2021.3108238.

[25] G. Tech, J. Pfaff, H. Schwarz, P. Helle, A. Wieckowski, D. Marpe, T. Wiegand, Rate-distortion-time cost aware CNN training for fast VVC intra-picture partitioning decisions, in: 2021 Picture Coding Symposium (PCS), 2021, pp. 1–5, http://dx.doi.org/10.1109/PCS50896.2021.9477452.

[26] S.-h. Park, J.-W. Kang, Fast multi-type tree partitioning for versatile video coding using a lightweight neural network, IEEE Trans. Multimed. 23 (2021) 4388–4399, http://dx.doi.org/10.1109/TMM.2020.3042062.

[27] S.-H. Tsang, N.-W. Kwong, Y.-L. Chan, Fastsccnet: Fast mode decision in VVC screen content coding via fully convolutional network, in: 2020 IEEE International Conference on Visual Communications and Image Processing (VCIP), 2020, pp. 177–180, http://dx.doi.org/10.1109/VCIP49819.2020.9301885.

[28] C. Peng, X. Zhang, G. Yu, G. Luo, J. Sun, Large kernel matters — Improve semantic segmentation by global convolutional network, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 1743–1751, http://dx.doi.org/10.1109/CVPR.2017.189.

[29] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2261–2269, http://dx.doi.org/10.1109/CVPR.2017.243.

[30] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer, 2015, pp. 234–241.

[31] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778, http://dx.doi.org/10.1109/CVPR.2016.90.

[32] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Adv. Neural Inf. Process. Syst. 25 (2012).

[33] H. Touvron, P. Bojanowski, M. Caron, M. Cord, A. El-Nouby, E. Grave, G. Izacard, A. Joulin, G. Synnaeve, J. Verbeek, et al., Resmlp: Feedforward networks for image classification with data-efficient training, 2021, arXiv preprint arXiv:2105.03404.

[34] T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, Focal loss for dense object detection, IEEE Trans. Pattern Anal. Mach. Intell. 42 (2) (2020) 318–327, http://dx.doi.org/10.1109/TPAMI.2018.2858826.

[35] E. Agustsson, R. Timofte, NTIRE 2017 challenge on single image super-resolution: Dataset and study, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2017, pp. 1122–1131, http://dx.doi.org/10.1109/CVPRW.2017.150.

[36] G. Tech, V. George, J. Pfaff, A. Wieckowski, B. Bross, H. Schwarz, D. Marpe, T. Wiegand, Learning-based encoder algorithms for VVC in the context of the optimized VVenC implementation, in: Applications of Digital Image Processing XLIV, Vol. 11842, SPIE, 2021, pp. 31–42.