Parallel-Based Fast Coding Mode Decision for Intra Coding in VVC SCC

Dayong Wang, Kongqing Peng, Xin Lu, Frédéric Dufaux, Shibing Zhang, Weian Li, Hongwei Guo

PARALLEL-BASED FAST CODING MODE DECISION FOR INTRA CODING IN VVC SCC

Dayong Wang[1,2], Kongqing Peng[3], Xin Lu[4], Frederic Dufaux[5], Shibin Zhang[2], Weian Li[6], Hongwei Guo[7*]*

[1]Dayong Wang is with the Chongqing Key Laboratory on Big Data for Bio Intelligence,
Chongqing University of Posts and Telecommunications, China
[2]Advanced Cryptography and System Security Key Laboratory of Sichuan Province,
Chengdu University of Information Technology, China
[3] School of Communications and Information Engineering,
Chongqing University of Posts and Telecommunications, China
[4]Faculty of Computing, Engineering and Media (CEM), De Montfort University, UK
[5]Université Paris-Saclay, CNRS, CentraleSupélec, Laboratoire des signaux et systèmes, France
[6]School of Big Data and Computer Science, Guiyang, Guizhou Normal University, China
[7]School of Engineering, Honghe University, China
wangdayong@cqupt.edu.cn, 715769542@qq.com,
xin.lu@dmu.ac.uk, frederic.dufaux@l2s.centralesupelec.fr, cuitzsb@cuit.edu.cn,
liweian@gznu.edu.cn, ghw202@foxmail.com

ABSTRACT

In light of the growing popularity of screen content video applications, there is a increasing demand for Screen Content Coding (SCC). The latest standard, Versatile Video Coding (VVC), exhibits exceptionally high coding efficiency, albeit accompanied by a considerable coding complexity. This complexity, in turn, restricts the widespread applicability of VVC SCC. To address this issue, this paper introduces a parallel based approach to enhance the coding speed of VVC SCC Intra Coding. Specifically, we established a large-scale database and then design distinct neural networks for Coding Units (CUs) of various sizes to predict candidate Coding Modes (CMs). Subsequently, we formulate a loss function based on CM distributions and Rate Distortion(RD) costs to train the designed models. Finally, we introduce a threshold selection scheme to balance coding efficiency and coding speed. Experimental results demonstrate that the proposed method improves coding speed by an average of 36.36%, with an average increase of 0.95% in Bjøntegaard Delta Bit Rate (BDBR).

***Index Terms*—** Versatile Video Coding (VVC), Screen Content Coding (SCC), coding mode, loss function, early terminate

*Corresponding author

This work was supported in part by the Key Industry Science and Technology Service Project of Yunnan Provincial Universities under Grant FWCY-ZNT2024025; in part by the Open Fund of Advanced Cryptography and System Security Key Laboratory of Sichuan Province under Grant SKLACSS-202410; and in part by the National Natural Science Foundation of China under Grant 62461023.

1. INTRODUCTION

Recent advancements have significantly increased the deployment of screen sharing and wireless display applications. In these contexts, the video content is referred to as screen content, which typically includes images and videos generated by computers. This content is characterized by repeated patterns, expansive flat areas, a constrained color palette, and the absence of sensor noise [1]. The latest standard, Versatile Video Coding (VVC) [2], introduces several new coding tools, including the quad-tree plus multi-type tree (QTMT) coding structure, which supports up to six split modes for Coding Units (CUs). Moreover, VVC SCC necessitates the evaluation of two additional SCC CMs inherited from High Efficiency Video Coding (HEVC) [3]: Intra-Block Copy (IBC) [4] and Palette (PLT) [5]. However, the introduction of these new coding tools and the need for additional CMs evaluations have significantly increased the coding complexity of VVC SCC, hindering its widespread application. Therefore, improving the coding speed of VVC SCC is imperative.

In order to enhance the coding speed of VVC SCC, we propose a parallel-based fast CM decision algorithm for VVC SCC Intra coding. Specifically, we first design separate neural networks for CUs of various sizes. Subsequently, a loss function is formulated based on CM distributions and RD costs, followed by the introduction of a threshold selection scheme. Experimental results show that the proposed method can significantly increase the encoding speed by an average of 36.36%, with an average increase of 0.95% in BDBR.

The main contributions of this paper are summarized as follows:

- A large-scale database was established for fast coding mode decision in parallel coding of Intra coding in VVC SCC. This database is intended to facilitate the application of deep learning techniques to reduce the complexity of VVC SCC.
- Multi-scale convolutional kernel networks were developed for parallel coding. These networks simultaneously and accurately predict the coding modes of all CUs in VVC SCC.
- A loss function was formulated based on CM distributions and RD. Additionally, an effective threshold selection scheme was introduced.

2. RELATED WORK

Several approaches have recently been proposed to accelerate the decision-making process for CM selection in SCC.

In recent years, machine learning classifiers have been employed to improve coding speed. The algorithm described in [6] first categories CUs into two groups: natural content CUs and screen content CUs. Based on these categories, mode candidate sets are determined, with INTRA modes used for natural content CUs, and IBC and PLT modes for screen content CUs. In [7], decision trees and random forests are used to identify and skip unlikely CMs. In [8], online learning techniques are introduced, using colour count to skip unlikely modes and enabling early termination of CU splitting based on RD costs. However, these machine learning-based approaches rely heavily on feature extraction, which can be a challenging task. Currently, deep learning is extensively used across various areas, enabling automatic feature extraction from extensive data and achieving superior performance. In [9], Convolutional Neural Network (CNN) is employed to categorize CUs into natural, text, image, and color blocks. Subsequently, candidate modes are predicted based on CU sizes and types. However, these methods are specifically tailored for HEVC SCC. Given the substantial differences in CU sizes between HEVC SCC and VVC SCC, achieving optimal performance may still present challenges.

In the context of VVC SCC, Wang et al. [10] proposes early skipping of Intra modes, block differential pulse coded modulation, and transform selection early termination to enhance coding speed. Wang et al. [11] utilizes deep learning to predict content types for all CUs and analyze their CM distributions for mode prediction. Wang et al. [12] presents a fast coding mode selection scheme for intra prediction in VVC SCC. Jiao et al. [13] use a width-adaptive CNN to predict candidate CU partitions and exclude unlikely ones. Tsang et al. [14] proposes a U-Net-based classifier to predict content types for all 4×4 CUs within the Largest Coding Unit (LCU), inferring content types for larger CUs from their sub-CUs. While mode distributions correlate with content types, CUs with the same content type can have significantly different mode distributions, suggesting that relying solely on content to predict candidate modes may not ensure optimal performance.

To address the identified issues, this paper introduces a parallel-based fast CM decision for Intra coding in VVC SCC.

3. PROPOSED PARALLEL-BASED PREDICTION

The QTMT structure recursively divides a Coding Tree Unit (CTU) using a quadtree (QT) until further partitioning is not possible. Subsequently, the multi-type tree (MT) structure performs additional recursive partitioning on the leaf nodes generated by the quadtree. During this process, each CU is sequentially examined across multiple stages until its minimum side length reaches 4, as shown in Fig. 1.

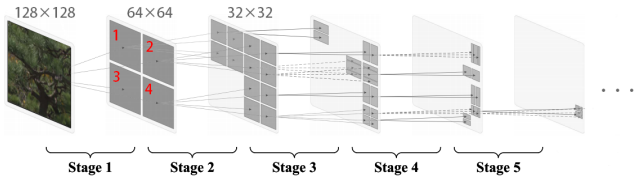

Fig. 1. Sequential checking of a CTU in multiple stages.

As shown in Fig. 1, a CTU is initially divided into a 128×128 CU, which is then subdivided into four 64×64 CUs in Stage 1. In Stage 2, each 64×64 CU is divided into four 32×32 CUs. In Stage 3, these 32×32 CUs are recursively partitioned using six split modes (SMs) until the minimum side length of 4 is achieved. Throughout this process, CUs are processed sequentially rather than simultaneously. However, the QTMT structure generates a large number of CUs, and the computational complexity of deep learning models can significantly impact efficiency when applied sequentially to predict content types for all CUs. To address this, we propose a parallel prediction method. Using parallel computing, we simultaneously predict potential SMs for all CUs within a frame while excluding unlikely SMs for each CU in every CTU, as shown in Fig. 2. This method ensures that the prediction time is limited to the maximum time required for a single CU, which is negligible.

To enable accurate parallel predictions, we established a database, designed an appropriate network architecture, defined a suitable loss function, and implemented a threshold selection scheme.

3.1. Database construction for coding modes

There are up to 17 CU types of various sizes. In order to accurately predict coding modes for all the CUs, we first investigated the mode distribution for different CU types. We used VTM-17.0 to encode 13 test sequences under the Common Test Conditions (CTC) [15]. These sequences encompass four classes: Robot in class A; EBURainFruits and Kimono1

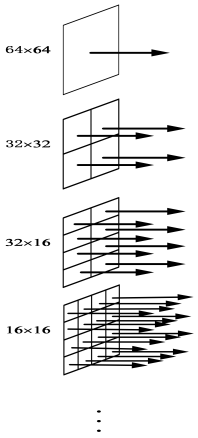

Fig. 2. Parallel prediction method for CUs

in class CC; BasketballScreen, MissionControlClip2, and MissionControlClip3 in class Mixed; and Console, Desktop, FlyingGraphics, Map, Programming, SlideShow, and WebBrowsing in class TGM. These sequences were encoded using QP values of 22, 27, 32, and 37 under the All Intra (AI) configuration. Fig. 3 illustrates the mode for CUs of various sizes.

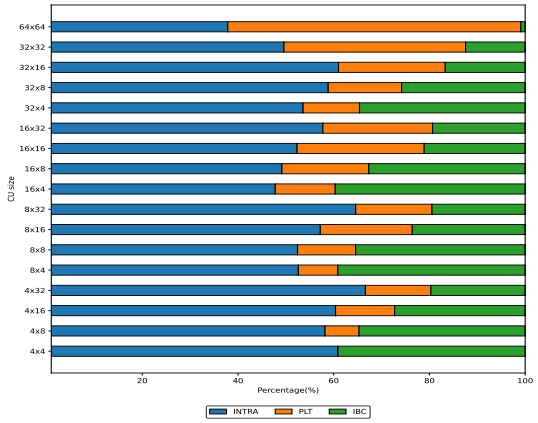

Fig. 3. Modes distribution for CUs of various sizes.

It is evident from Fig. 3 that the mode distributions for CUs of various sizes exhibit significant differences. If CUs of different sizes use the same database for training, it will inevitably be difficult to achieve optimal results. To address this, separate databases should be created for each CU size. Since there are 17 distinct CU sizes, 17 corresponding content-type databases should be constructed. Each CU should be labeled with one of the three coding modes: Intra, IBC, or PLT. The video sequences mentioned earlier will be used to build the coding mode databases. Each CU, along with its corresponding label, forms a sample in these databases, which are then divided into training and validation sets at an 8:2 ratio.

3.2. Multi-scale convolutional kernel networks design for parallel coding

Given that there are 17 different CU sizes, it is advisable to design network structures tailored to each CU to enhance coding speed.

We have designed neural networks that utilizes multi-scale convolutional kernels to adapt to CUs of various sizes. Specifically, we employ convolutional kernels of sizes 2×2, 3×3, 5×1, and 3×1 to predict potential coding modes. For square CUs, only square kernels are used, while for rectangular CUs, we apply 5×1 and 3×1 rectangular kernels to accommodate different aspect ratios. To further improve prediction accuracy, we use a stride of 1 for the rectangular kernels, which allows for finer granularity in the predictions.

According to the aspect ratio of the length to the width, these CUs can be categorized into four groups, corresponding to ratios of 1, 2, 4, and 8, respectively. We have designed the corresponding networks as follows.

For CUs with an aspect ratio of 1, where the length and width are equal, we exclusively use square convolution neural networks to predict the candidate CMs. Specifically, 3×3 convolutional kernels with a padding of 1 and a stride of 1, referred to as ConvP, are used to extract more features in the convolutional layers. Additionally, 2×2 convolution kernels with a stride of 2, denoted as Conv, are applied for non-overlapping convolutions in the convolutional layers. To reduce losses in predictions, pooling layers are omitted for these CUs.

For CUs with aspect ratios of 2, 4, and 8, where the length and width differ, a combination of square and non-square convolution kernels is employed to predict candidate CMs. Specifically, both 5×1 and 3×1 convolution kernels, each with a stride of 1, are utilized to extract additional features. These convolution layers are denoted as AsCon. For square convolution kernels, 2×2 convolution kernels with a stride of 2 are employed for non-overlapping convolutions. To minimize prediction losses, pooling layers are intentionally omitted for these CUs.

By employing a sequence of convolutional layers, we obtain an adequate number of feature maps. Subsequently, these feature maps are flattened and passed through three fully connected layers to compute the probabilities associated with the three modes. Each layer is followed by a rectified linear unit (ReLU) activation function, with a softmax activation applied at the final layer. Fig. 4 illustrates the network structures for CUs of sizes 32×32, 32×16, 32×8, and 32×4, respectively.

3.3. Loss Function Design

In conventional classification tasks, the cross-entropy function is commonly uesd as the loss function. However, it does not consider the disparity between predicted and true values. Specifically, different CMs typically exhibit distinct RD costs, which significantly affect coding efficiency. As a result, di-

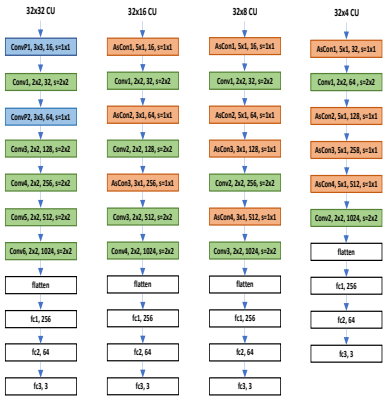

Fig. 4. The network structures for CUs of sizes 32×32, 32×16, 32×8, and 32×4.

rectly applying the cross-entropy function is insufficient to address this issue. Therefore, we have formulated a novel loss function.

Assuming the mini-batch includes N CUs, with each CU containing three CMs, the corresponding basic cross-entropy loss function can be expressed as:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{3} y_{ij}\log p_{ij}, \tag{1}$$

where y_{ij} denotes the ground-truth binary label for the i th CU which selects the j th CM as the best coding mode, while p_{ij} represents the corresponding predicted probability.

As observed in Fig. 1, the distribution of CMs is highly unbalanced. To address this issue, we have introduced a scaling factor that automatically reduces the impact of easier examples during training. This approach rapidly directs the model's attention towards more challenging examples, thus accelerating learning for these instances [16]. Consequently, the cross-entropy can be modified as follows

$$L = -\frac{1}{N}\sum_{i=1}^{N}(1-pt)\sum_{j=1}^{3} y_{ij}\log p_{ij}, \tag{2}$$

where pt is the predicted probability of the ground-truth class. pt is adjustable and is used to specify the significance weights.

As mentioned earlier, RD costs have a significant impact on coding efficiency. However, Eq. (2) does not consider the RD costs associated with CMs. In order to address this issue, RD costs are incorporated into the loss function. We first derive the penalty weight for RD costs as follows

$$weight\_cost_i = \frac{r_{ij}}{r_{ib}}, \tag{3}$$

where r_{ij} denotes the RD cost of the i th CU selecting the j th CM, while r_{ib} represents the RD cost of the best CM for the CU. When the j th CM is identified as the best one, the RD cost penalty weight is set to 1, and no penalty is imposed. Conversely, if the j th CM is not the best one, the RD cost penalty weight is greater than 1 and a penalty is applied. Moreover, a higher RD cost penalty weight results in a more severe punishment. In other words, the RD cost penalty weight is proportional to the severity of the punishment. Therefore, we can incorporate the RD cost penalty weight into Eq. (2) to derive the overall loss function as follows

$$L = -\frac{1}{N}\sum_{i=1}^{N}(1-pt)\sum_{j=1}^{3}\frac{r_{ij}y_{ij}}{r_{ib}}\log p_{ij}. \tag{4}$$

3.4. Selection of Threshold Values

Through the aforementioned process, we can obtain the trained models for CUs of various sizes. Using these models, we can calculate the probabilities for the three CMs and arrange them in descending order of their probability values. Let q_i denote the probability of the i th CM, and m is the sum of the probabilities for the first n modes

$$m = \sum_{i=1}^{n} p_i. \tag{5}$$

To enhance coding speed and maintain coding efficiency, it is advisable to exclude unlikely CMs. This can be achieved by implementing early termination of CMs selection when s is greater than or equal to a predefined threshold T. Determining the optimal threshold is crucial. To address this, various commonly used T values, such as 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, and 0.9, were tested across the 11 training sequences under the aforementioned CTC.

Fig. 5 illustrates the coding efficiency losses, as represented by BDBR [17]. A positive BDBR indicates a decrease in coding efficiency relative to the reference software, specifically reflecting the percentage of bitrate increase in bitrate required to maintain a given quality. Fig. 5 illustrates that

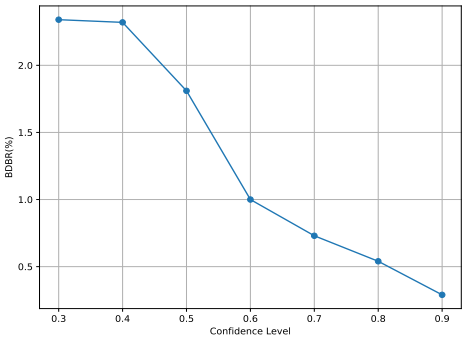

Fig. 5. T values and the corresponding coding efficiency.

Table 1. Performance comparison with existing algorithms

Categories	Sequences	Proposed		WA-CNN [13]		FastSCCNET [14]	
		BDBR(%)	ΔT(%)	BDBR(%)	ΔT(%)	BDBR(%)	ΔT(%)
A	Robot	0.59	29.87	1.44	36.05	1.98	37.33
CC	EBURainFruits	-0.02	33.43	0.95	33.75	0.07	36.44
	Kimono1	0.12	19.93	0.92	27.16	-0.07	24.66
Mixed	BasketballScreen	1.01	31.64	2.47	37.59	4.38	25.16
	MissionControlClip2	0.56	34.86	1.45	28.05	2.17	31.23
	MissionControlClip3	0.79	37.98	2.72	37.68	5.71	29.32
TGM	Console	1.29	47.70	2.17	40.37	1.98	23.69
	Desktop	1.39	52.40	2.30	41.33	2.24	32.29
	FlyingGraphics	1.87	39.43	1.66	41.82	6.66	18.51
	Map	1.60	33.00	1.45	39.11	2.28	26.87
	Programming	1.18	38.03	2.67	39.69	3.64	24.55
	SlideShow	0.79	25.30	1.09	34.41	3.77	23.19
	WebBrowsing	1.22	49.08	2.43	40.55	7.01	17.62
All	**Average**	**0.95**	**36.36**	**1.82**	**36.74**	**3.22**	**26.98**

as T increases, the BDBR decreases progressively, and vice versa. This phenomenon can be attributed to the fact that the smaller values of s result in the erroneous exclusion of more CMs, which increases the BDBR while also enhancing the coding speed. This reflects a trade-off between BDBR and improvements in coding speed. In addition, observations reveal that when T is greater than or equal to 0.6, the decrease rate of BDBR stabilizes and slows down. Consequently, we have selected 0.6 as the threshold value for T.

4. EXPERIMENTS

4.1. Training and Testing of the Neural Network

To eliminate any potential overlaps between the training and testing sets, we deliberately chose 11 training sequences that are not included in the CTC to generate the training samples. These sequences are carefully selected to cover various video content and resolutions from [18]. Specifically, the chosen training sequences are: ClearTypeSpreadsheet, KristenAndSaraScreen, MissionControlClip1, ParkScene, PcbLayout, PeopleInVehicle, PptDocXls, RealTimeData, Seeking, VideoConferencingDocSharing, and WordEditing.

The training sequences are encoded using QP values of 22, 27, 32, and 37 to derive the ground truth labels. The batch size is set to 8192. The Adam optimizer is used throughout the training process, with PyTorch used on an RTX4090 for training.

When the number of epochs is 300 or more, the training losses remain nearly constant. Consequently, we selected 300 epochs for the training process, resulting in the acquisition of trained models for all CU sizes.

4.2. Performance Evaluation

To validate the performance of our proposed algorithm, we conducted tests using VTM-17.0 on a server equipped with an AMD Ryzen9 7950x processor, 128GB of DDR5 memory, and an RTX4090 GPU. According to CTC, 13 test sequences were examined in our experiments. All these sequences are tested under the AI configuration, with the QP set of (22, 27, 32, 37). The performance of the proposed algorithm was evaluated by coding efficiency losses and coding speed improvements. Coding efficiency loss was measured using BDBR, while coding speed improvement was quantified by the percentage of coding time saved, denoted as ΔT. To evaluate the performance of the proposed algorithm, we conducted a comparative analysis with two existing algorithms: WA-CNN[13] and FastSCCNET[14]. Since the proposed algorithm focuses on modes, we select WA-CNN, which is mode-related, from the literature for comparison. To the best of our knowledge, these algorithms are recognized as the state-of-the-art benchmarks for fast intra coding in SCC. The corresponding performance comparisons are detailed in Table 1. It is important to note that the test results include the inference times for the models employed.

Table 1 presents the average coding speed improvements of the proposed algorithm, WA-CNN, and FastSCCNET, which are 36.36%, 36.74%, and 26.98%, respectively. The corresponding average BDBRs for the proposed algorithm, WA-CNN, and FastSCCNET are 0.95%, 1.82%, and 3.22%, respectively. Obviously, the proposed algorithm outperforms FastSCCNET in both coding speed and efficiency. It is worth noting that the coding speed of the proposed algorithm is essentially the same as that of WA-CNN, but the BDBR of the proposed algorithm is much smaller than that of WA-CNN. If the BDBR of the proposed algorithm were the same as that of WA-CNN, then the coding speed of the proposed algorithm would be significantly faster than that of WA-CNN.

5. CONCLUSIONS

In this paper, we presented a parallel-based fast CMs decision algorithm for intra coding in VVC SCC. Considering the significant differences in CMs distributions across CUs of different sizes, designed and trained distinct models to predict candidate CMs and exclude unlikely ones for each CU size. The experimental results demonstrate that the proposed algorithm significantly improves coding speed with negligible coding efficiency losses. Future research will focus on refining the Intra mode prediction process to further enhance coding speed.

6. REFERENCES

[1] Jizheng Xu, Rajan Joshi, and Robert A Cohen, "Overview of the emerging HEVC screen content coding extension," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, pp. 50–62, Jan 2016.

[2] Tianyi Li, Mai Xu, Runzhi Tang, Ying Chen, and Qunliang Xing, "DeepQTMT: A deep learning approach for fast QTMT-based cu partition of intra-mode VVC," *IEEE Transactions on Image Processing*, vol. 30, pp. 5377–5390, 2021.

[3] Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, pp. 1649–1668, Dec 2012.

[4] Xiaozhong Xu, Shan Liu, Tzu-Der Chuang, Yu-Wen Huang, Shaw-Min Lei, Krishnakanth Rapaka, Chao Pang, Vadim Seregin, Ye-Kui Wang, and Marta Karczewicz, "Intra block copy in HEVC screen content coding extensions," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 6, pp. 409–419, Apr 2016.

[5] Wei Pu, Marta Karczewicz, Rajan Joshi, Vadim Seregin, Feng Zou, Joel Sole, Yu-Chen Sun, Tzu-Der Chuang, Polin Lai, Shan Liu, Shih-Ta Hsiang, Jing Ye, and Yu-Wen Huang, "Palette mode coding in HEVC screen content coding extension," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 6, pp. 420–432, Apr 2016.

[6] Jianjun Lei, Dongyang Li, Zhaoqing Pan, Zhenyan Sun, Sam Kwong, and Chunping Hou, "Fast intra prediction based on content property analysis for low complexity HEVC-based screen content coding," *IEEE Transactions on Broadcasting*, vol. 63, pp. 48–58, Jan 2017.

[7] Wei Kuang, Yui-Lam Chan, Sik-Ho Tsang, and Wan-Chi Siu, "Machine learning-based fast intra mode decision for HEVC screen content coding via decision trees," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, pp. 1481–1496, May 2020.

[8] Wei Kuang, Yui-Lam Chan, Sik-Ho Tsang, and Wan-Chi Siu, "Online-learning-based bayesian decision rule for fast intra mode and cu partitioning algorithm in HEVC screen content coding," *IEEE Transactions on Image Processing*, vol. 29, pp. 170–185, 2020.

[9] Changsheng Gao, Li Li, Dong Liu, Zhibo Chen, Weiping Li, and Feng Wu, "Two-step fast mode decision for intra coding of screen content," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 8, pp. 5608–5622, 2022.

[10] Dayong Wang, Yishen Deng, Weisheng Li, Xin Lu, Frederic Dufaux, Bo Hang, and Ce Zhu, "Fast intra mode prediction algorithms for scbs in VVC SCC," ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2024, pp. 4395–4399.

[11] Dayong Wang, Junyi Yu, Xin Lu, Frederic Dufaux, Bo Hang, Hui Guo, and Ce Zhu, "Fast mode and cu splitting decision for intra prediction in VVC SCC," *IEEE Transactions on Broadcasting*, vol. 70, pp. 872–883, 2024.

[12] Dayong Wang, Junyi Yu, Xin Lu, Frederic Dufaux, Hongwei Guo, Hui Guo, and Ce Zhu, "Fast coding mode prediction for intra prediction in VVC SCC," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 2024.

[13] Chao Jiao, Huanqiang Zeng, Jing Chen, Chih-Hsien Hsia, Tianlei Wang, and Kai-Kuang Ma, "Width-adaptive cnn: Fast cu partition prediction for VVC screen content coding," *IEEE Transactions on Multimedia*, 2024, Early access.

[14] Sik-Ho Tsang, Ngai-Wing Kwong, and Yui-Lam Chan, "Fastsccnet: Fast mode decision in vvc screen content coding via fully convolutional network," 2020 IEEE International Conference on Visual Communications and Image Processing (VCIP), 2020, pp. 177–180.

[15] Chao Yung-Hsuan, Sun Yu-Chen, Xu Jizheng, and Xu Xiaozhong, "common test conditions and software reference configurations for non-4:2:0 colour formats," 2020.

[16] Jizheng Xu, Rajan Joshi, and Robert A. Cohen, "Intra block copy in HEVC screen content coding extensions," 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2399–3007.

[17] Gisle Bjøntegaard, "Calculation of average psnr differences between rd-curves," 2001.

[18] Tourapis Alexis, Michael, Singer David, and Kolarov Krasimir, "New test sequences for screen content coding," pp. 177–180, 2020.