FastSCCNet: Fast Mode Decision in VVC Screen Content Coding via Fully Convolutional Network

Sik-Ho Tsang, Ngai-Wing Kwong, and Yui-Lam Chan
Department of Electronic and Information Engineering
The Hong Kong Polytechnic University
Hung Hom, Kowloon, Hong Kong SAR
sik-ho.tsang@polyu.edu.hk, ngai-wing.kwong@connect.polyu.hk, enylchan@polyu.edu.hk

***Abstract*—Screen content coding have been supported recently in Versatile Video Coding (VVC) to improve the coding efficiency of screen content videos by adopting new coding modes which are dedicated to screen content video compression. Two new coding modes called Intra Block Copy (IBC) and Palette (PLT) are introduced. However, the flexible quad-tree plus multi-type tree (QTMT) coding structure for coding unit (CU) partitioning in VVC makes the fast algorithm of the SCC particularly challenging. To efficiently reduce the computational complexity of SCC in VVC, we propose a deep learning based fast prediction network, namely FastSCCNet, where a fully convolutional network (FCN) is designed. CUs are classified into natural content block (NCB) and screen content block (SCB). With the use of FCN, only one shot inference is needed to classify the block types of the current CU and all corresponding sub-CUs. After block classification, different subsets of coding modes are assigned according to the block type, to accelerate the encoding process. Compared with the conventional SCC in VVC, our proposed FastSCCNet reduced the encoding time by 29.88% on average, with negligible bitrate increase under all-intra configuration. To the best of our knowledge, it is the first approach to tackle the computational complexity reduction for SCC in VVC.**

Keywords—screen content coding (SCC), Versatile Video Coding (VVC), convolutional neural network (CNN), fully convolutional network (FCN), deep learning

I. Introduction

With the recent development of technologies in networking and portable devices, computer screen sharing applications have become much more popular. The applications includes remote desktop, video conferencing with document and slideshow sharing. These videos contain the mixture of camera-captured content (CC) and screen content (SC) as depicted in Fig. 1. In addition, there are also many TV programs and Internet videos which contain both CC and SC. There will be even cloud services using screen sharing technology in the coming future [1]. Efficient compression of SC is highly demanded. Thus, in January 2014, there was Call for Proposal (CfP) [2] of screen content coding (SCC) [3] as the extension of High Efficiency Video Coding (HEVC) [4] by the Joint Collaborative Team on Video Coding (JCT-VC). Two major coding modes, intra block copy (IBC) [5]-[7] and palette (PLT) [8]-[9] modes, were introduced to improve the coding efficiency. However, since the SCC extension was developed after HEVC released, SCC in HEVC is not widespread. Recently, the Joint Video Experts Team (JVET) started to develop the next-generation video coding standard, i.e. Versatile Video Coding (VVC) [10]-[11]. IBC has been included in VVC at the early stage while PLT has just been supported at recent development. The common test condition (CTC) for the compression of SC has also just been developed in April 2020 [13]. Wide range

This work is supported by a grant from the Research Grants Council of the HKSAR, China (Grant No. PolyU 152069/18E).

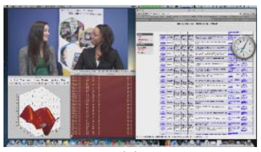
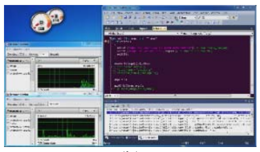

(a)

(b)

Fig. 1. Examples of screen content sequences: (a) MissionControlClip3, (b) Programming

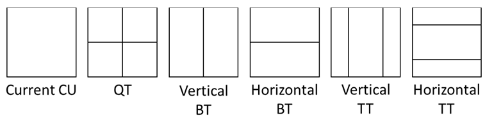


Fig. 2. Illustration of quad-tree plus multi-type tree (QTMT) coding in VVC.

of SC applications using VVC with the enabling of IBC and PLT are expected in the future when VVC is finalized and released. However, the computational complexity of VVC increases by 18 times over that of HEVC under all intra test configuration [15] because of the introduction of new coding tools in VVC. One of the major tools is the new quad-tree plus multi-type tree (QTMT) coding structure for coding unit (CU) partitioning which largely increases the computational complexity. With IBC and PLT modes, the computational complexity is even higher. Hence, it is necessary to develop the fast approach for reducing the computational complexity of VVC in order to meet the needs of practical applications.

II. SCC Intra Coding in VVC

A. *Quad-Tree plus Multi-type Tree (QTMT) Coding*

In HEVC, each video frame is divided into coding tree units (CTUs) of 64×64. Only quad-tree (QT) coding is supported for CU partitioning where each CTU can be recursively split into four equal sized square sub-CUs from the largest size of 64×64 down to 8×8. In VVC, QTMT coding is supported as in Fig. 2. Except QT, there are also binary-tree (BT) and ternary-tree (TT). For BT and TT, each of them has horizontal and vertical modes. That means, for each CU, the choices of CU partitioning have been increased from two in HEVC to six in VVC. Since each CTU can be split recursively from 128×128 down to 4×4, with such various shapes and sizes supported, the computational complexity is increased largely in VVC compared to HEVC.

B. *Conventional Intra (INTRA) Mode*

For conventional intra (INTRA) mode [10], similar to HEVC, two non-directional mode candidates, DC and planar, plus directional mode candidates are used to encode the CUs. But in VVC, the intra mode candidates are increased to 67 instead of only 35 in HEVC. The computational complexity is increased for the conventional intra mode in each CU. However, INTRA is not efficient to encode the SC using the neighbor boundary pixels since SC usually contains sharp edges with high contrast as shown in Fig. 1.

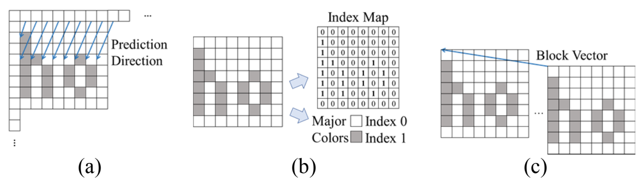


Fig. 3. Illustrations of (a) INTRA, (b) PLT, and (c) IBC modes.

TABLE I. ENCODING TIME DIFFERENCE ΔT (%) AND BD-RATE (%) COMPARED WITH THE CONVENTIONAL VVC WITH IBC AND PLT ENABLED

Sequences	IBC & PLT Off		IBC Off		PLT Off	
	ΔT	BD-Rate	ΔT	BD-Rate	ΔT	BD-Rate
ArenaOfValor	-45.75	3.39	-42.53	2.53	-3.48	0.51
BasketballScreen	-41.09	57.85	-37.19	25.40	-2.99	11.53
BirdsInCage	-33.17	0.00	-29.60	0.01	-4.72	-0.12
ChineseEditing	-37.94	83.76	-34.43	12.25	-5.56	41.68
Console	-23.23	100.06	-30.04	23.78	11.47	26.74
CrowdRun	-40.34	0.06	-38.21	0.09	-3.12	0.00
Desktop	-27.06	218.90	-30.57	53.39	0.85	38.51
EBULupoCandleLight	-40.77	0.27	-37.30	0.29	-3.37	-0.08
EBURainFruits	-43.33	-0.04	-39.62	0.01	-3.72	-0.05
FlyingGraphics	-34.73	88.76	-32.22	39.93	0.56	13.54
Glasshalf	-47.32	12.49	-43.65	3.01	-2.48	6.18
Kimono1	-30.01	-0.01	-26.91	-0.05	-3.35	0.02
Map	-48.51	20.56	-41.71	3.25	-7.39	12.51
MissionControlClip2	-41.43	112.09	-33.98	48.60	-4.36	15.03
MissionControlClip3	-37.27	114.94	-33.01	48.47	-4.28	13.93
Programming	-34.60	71.88	-31.34	33.03	-2.34	10.13
Robot	-44.15	2.32	-41.19	1.93	-3.08	0.27
SlideShow	-41.57	35.38	-36.83	7.56	-0.74	14.57
Traffic	-44.60	0.92	-41.84	0.97	-2.35	-0.09
VenueVu	-40.27	2.63	-38.83	2.58	-2.54	0.05
WebBrowsing	-34.22	133.65	-36.89	36.99	-0.45	26.67
Average	**-38.63**	**50.47**	**-36.09**	**16.38**	**-2.26**	**11.02**

C. *Intra Block Copy (IBC) and Palette (PLT) Modes*

To efficiently compress SC videos, two major coding tools, IBC and PLT modes are enabled.

For PLT [8]-[9] mode, as in Fig. 3(b), a CU is separated into color data and structural data. The color data consists of few major colors which are predicted from color table or from neighbor CUs. When the colors are not in the major color tables, they are regarded as escape colors and are explicitly coded. The structural data is then represented by an index map and the corresponding indices are entropy coded. Thus, PLT helps to encode SC which contains few colors like texts and icons.

For IBC [5]-[7] mode, as in Fig. 3(c), is a block matching technique to find the repeating patterns within the same frame which frequently occurs in SC. For instance, repeating numbers and characters can be found within a document and spreadsheet. If IBC is used by one particular CU, the CU is encoded with a block vector (BV), as well as the residual signal of that CU which is similar to an inter mode in inter-frame motion estimation. Besides, merge mode and skip mode are also performed in IBC.

With additional coding tools enabled for SC, each CU needs to check INTRA, IBC and PLT modes which increases the computational complexity. Table I tabulates the encoding time difference (ΔT) and Bjøntegaard delta bitrate (BD-Rate) [12] with IBC or PLT disabling compared with the conventional VVC with both IBC and PLT enabled. The sequences are the SC testing sequences in CTC [13]. They are encoded by VTM-9.2 [14] using quantization parameters (QPs) of 22, 27, 32, and 37 under all-intra (AI) configuration. From the table, we can see that though with both IBC and PLT disabled, the BD-rate is increased by 50.47%, which indicates that they are very important coding tools for SC videos. Yet, the encoding time difference is also reduced by large amount of 38.63%. This means that they increase the encoding time a lot when they are being used. When only PLT is disabled, i.e. only IBC is enabled, the encoding time is reduced by only 2.26% but with large increase in BD-rate of 11.02%. This shows that PLT is well optimized in VVC, which does not occupy too much time of the whole encoding process. Thus, our proposed fast prediction network does not skip PLT since PLT can guarantee certain BD-rate reduction with only little increase in encoding time.

D. *Related Works and Contributions*

To speed up the encoding process of VVC, some fast approaches are proposed recently for fast QTMT [16]-[18]. In [16], rule-based approach based on CU variance, and variance of variance is proposed to skip unnecessary partitioning. In [17], multiple random forests are proposed to decide the CU whether split or not split for each partition type, as in Fig. 2. In [18], convolutional neural network (CNN) is proposed. Multiple networks are trained for various CU sizes to decide whether the CU should be split by which kind of partitioning. There is also a fast approach to speed up the multiple transform selection process using early termination algorithm based on the rate distortion (RD) cost [19].

However, to the best of our knowledge, though there are fast SCC approaches in HEVC [20]-[27], there is no prior art to speed up the encoding process for SC videos in VVC. We are the first to speed up the SCC in VVC. In this paper, we first propose a fast prediction network using fully convolutional network (FCN) to classify CUs into natural content blocks (NCBs) or screen content blocks (SCBs). By using FCN, only one shot inference is needed to classify the current CU and all the corresponding sub-CUs. Therefore, only one model is required rather than multiple models for different CU sizes. After block classification, different subsets of coding modes are assigned according to the block type, to accelerate the encoding process. Finally, the encoding time is reduced with small increase in BD-rate for SC videos.

III. PROPOSED FAST PREDICTION NETWORK: FASTSCCNET

A. *Network Architecture*

Fully convolutional network (FCN) [27] has been a success of a deep network for different kinds of pixel-wise segmentations such as semantic segmentation, instance segmentation, and biomedical image segmentation. Among these FCNs, U-Net [29] is one of the famous FCNs for biomedical image segmentation where it consists of the contraction and expansion paths. Inspired by U-Net [29], we design a FCN, namely FastSCCNet, which consists of the contraction and expansion paths as well. Yet, instead of pixel-wise segmentation, our FastSCCNet produces block-wise labels, which as shown in Fig. 4.

Since in the current VVC, each 128×128 CTU is split into four 64×64 CUs without any mode checking, the luminance component of a 64×64 CU is used as input with scaling of [0,1]. The output is the 16×16 labels in which each label represents a 4×4 sub-block with two classes, 0 indicates natural content block (NCB) and 1 indicates screen content block (SCB).

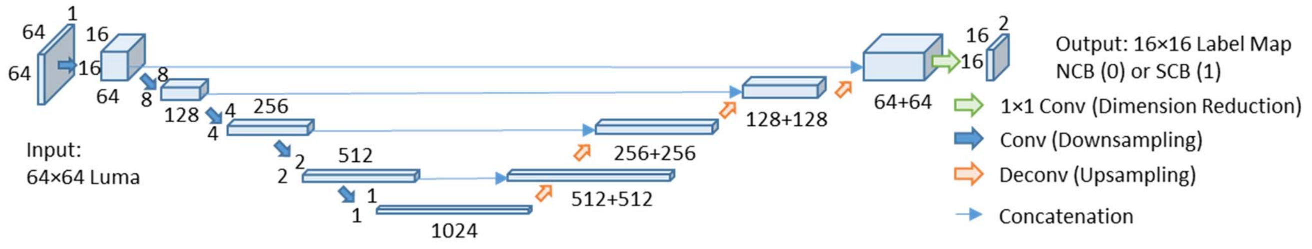


Fig. 4. FastSCCNet: Network Architecture.

At the contraction path, there are five convolutional layers (Conv1-Conv5) where the kernel size of conv1 is 4×4 and the kernel sizes of conv2 to conv5 are 2×2. The strides are set as the same size of kernel so that they are non-overlapping convolutions, in accordance with the non-overlapping CU partitioning coding structure. The number of feature maps is double after each convolution except Conv1. At the end of the contraction path, 1×1 size of 1024 feature maps are produced to extract the deep features of the 64×64 luminance input.

At the expansion path, there are four deconvolutional layers (Deconv1-Deconv4), with kernel size of 2×2 and stride of 2, to enlarge the feature maps back to the size of 16×16. Concatenation of the same-sized feature maps from the contraction path is performed so that local features and global features are fused. At each deconvolution, the number of feature maps is halved. Finally, a 1×1 convolution (Conv6) is performed to reduce the feature map number to 2, which is the output label number. Each convolution and deconvolution layer, except the last layer, is followed by batch normalization [27] and rectified linear unit (ReLU) activation function, while softmax is used at the last layer.

B. *Training Strategy*

Fifteen sequences are selected for training: BigDuck, BitstreamAnalyzer, ChineseDocumentEditing, CircuitLayoutPresentation, ClearTypeSpreadsheet, DucksAndLegs, OldTownCross, PcbLayout, PptDocXls, RealtimeData, Seeking, VideoConferencingDocSharing, Viking, and WordEditing, which are not included in CTC [13]. Thus, the training set does not overlap with the test set. One model of FastSCCNet is trained for each QP. For each training sequence, the first frame of each second is extracted. Among the extracted frames, the first 20 frames are encoded by the original VTM-9.2 [14] with QPs of 22, 27, 32 and 37 to obtain the ground truth labels. If the CU is coded as INTRA, it is NCB (0). If it is coded as IBC or PLT, it is SCB (1). By above encoding process, about 120k samples are obtained for training. The training process is implemented using TensorFlow [27] using the GeForce GTX 1080 Ti GPU. During inference, GPU is disabled, only CPU is used. Binary cross entropy loss over a mini-batch is used:

$$Loss = -\frac{1}{N}\sum_{i=1}^{N} \hat{y}_i \log y_i + (1-\hat{y}_i)\log(1-\hat{y}_i) \qquad (1)$$

where *N* is the mini-batch size of 4096. Each model is trained from scratch with 300 epochs conducted with the initial learning of 0.001 using the Adam optimizer.

C. *VVC Implementation*

CNN inference is conducted only when the CU is of the size 64×64. The predicted output for each CU, $\bar{y}_l$, with *l* being the class label (NCB or SCB), is the average of all predicted outputs, $\hat{y}(x,y,l)$, from 4×4 sub-blocks by FastSCCNet, where those 4×4 sub-blocks belong to the location and size of the corresponding CU:

$$\bar{y}_l = \frac{1}{|CU|}\sum_{(x,y)\in CU} \hat{y}(x,y,l) \qquad (2)$$

where $\hat{y}(x,y,l)$ is the predicted output label *l* for the 4×4 sub-block at position (x,y) inside the 64×64 CU, and $|CU|$ is the number of labels within the CU. Based on (2), only one-shot inference is required to get the predicted outputs $\bar{y}$, for the current 64×64 CU and all its sub-CUs with different shapes and sizes generated by QTMT CU partitioning in VVC. According to the value of $\bar{y}_l$, different sets of mode candidates are assigned to the CU:

$$M = \begin{cases} \{INTRA, PLT\}, & \text{if } \bar{y}_{SCB} < TH_{SCB} \\ \{IBC, PLT\}, & \text{if } \bar{y}_{NCB} < TH_{NCB} \\ \{INTRA, IBC, PLT\}, & \text{otherwise} \end{cases} \qquad (3)$$

where *M* is the mode candidate set to be checked for the current CU. With empirical testing, TH_{SCB} and TH_{NCB} are set to 0.05 when one of the dimensions for the current CU is smaller than 8 to have a high confidence in order for preserving the coding efficiency. For other CU sizes, TH_{SCB} and TH_{NCB} are set to 0.2 to reduce the computational complexity. As aforementioned, PLT is always kept since it only occupies little time.

TABLE II. ENCODING TIME DIFFERENCE ΔT (%) AND BD-RATE (%) OF FASTSCCNET WITH OR WITHOUT CONCATENATION COMPARED WITH THE CONVENTIONAL VVC WITH IBC AND PLT ENABLED ON VALIDATION SEQUENCE

Approaches	ΔT (%)	BD-Rate (%)
FastSCCNet (Without Concatenation)	-48.24	2.20
FastSCCNet	-48.49	1.08

IV. EXPERIMENTAL RESULTS

To evaluate the performance of our proposed FastSCCNet, sequences are encoded with QPs of 22, 27, 32, and 37 under all intra configuration according to CTC [13] using VTM-9.2 [14] with IBC and PLT enabled. Ablation study is performed to show the effectiveness of feature map concatenation using a validation sequence, English-DocumentEditing. Table II shows the encoding time difference (ΔT) and BD-rate [12] for FastSCCNet and FastSCCNet without concatenation, "FastSCCNet (Without Concatenation)", compared to the original VTM-9.2 with IBC and PLT enabled. FastSCCNet, with the feature map concatenation, much smaller increase in BD-rate is obtained with similar encoding time reduction. This shows that concatenating the early features with deep features helps to improve the performance of FastSCCNet. And the testing sequences are also encoded for complete evaluation. Table III tabulates the encoding time difference (ΔT) and BD-rate [12] for FastSCCNet compared to the original VTM-9.2 with IBC and PLT enabled. We can see that our proposed FastSCCNet can achieve 29.88% average encoding time reduction with

TABLE III. ENCODING TIME DIFFERENCE ΔT (%) AND BD-RATE (%) OF FASTSCCNET COMPARED WITH THE CONVENTIONAL VVC WITH IBC AND PLT ENABLED, ON CTC TESTING SEQUENCES

Sequences	ΔT	BD-Rate	Sequences	ΔT	BD-Rate
ArenaOfValor	-37.34	1.40	Kimono1	-24.66	-0.07
BasketballScreen	-25.16	4.38	Map	-26.87	2.28
BirdsInCage	-33.59	0.02	MissionControlClip2	-31.23	2.17
ChineseEditing	-31.89	1.37	MissionControlClip3	-29.32	5.71
Console	-23.69	1.98	Programming	-24.55	3.64
CrowdRun	-35.52	0.45	Robot	-37.33	1.98
Desktop	-32.29	2.24	SlideShow	-23.19	3.77
EBULupoCandleLight	-35.38	0.18	Traffic	-38.68	0.81
EBURainFruits	-36.44	0.07	VenueVu	-30.23	1.89
FlyingGraphics	-18.51	6.66	WebBrowsing	-17.62	7.01
Glasshalf	-33.98	2.80	**Average**	**-29.88**	**2.42**

only 2.42% increase in BD-rate. Compared to the variants of VVC with IBC or/and PLT disabling, as in Table I, the increase in BD-rate is insignificant.

V. CONCLUSIONS

In this paper, a deep learning based fast prediction network FastSCCNet was proposed to reduce the computational complexity of SCC in VVC. To avoid the exhaustive mode search in a CU, FastSCCNet outputs labels for all 4×4 sub-blocks within each 64×64 CU by one-shot inference. The current CU and all corresponding sub-CUs are classified into NCB or SCB based on the output labels. According to the classification type, different subsets of modes are checked. Unnecessary modes are skipped so that the encoding time can be reduced. Experimental results show that the proposed FastSCCNet provides an average computational complexity reduction of 29.88% with negligible increase in bitrate reduction. To the best of our knowledge, we are the first to speed up SCC encoding process in VVC. Our future work is to have the fast CU partitioning for SC videos in VVC.

REFERENCES

[1] Y. Lu, S. Li, and H. Shen, "Virtualized screen: A third element for cloudmobile convergence," *IEEE Multimedia*, vol. 18, no. 2, pp. 4–11, Feb. 2011.

[2] "Joint Call for Proposals for Coding of Screen Content," ISO/IEC JTC1/SC29/WG11, N14175, San Jose, CA, USA, Jan. 2014.

[3] J. Xu, R. Joshi, and R. A. Cohen, "Overview of the emerging HEVC screen content coding extension," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 1, pp. 50–62, Jan. 2015.

[4] G. J. Sullvian, J. Ohm,W. J. Han, and T.Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[5] S.-H. Tsang, W. Kuang, Y.-L. Chan, and W.-C. Siu, "Fast HEVC Screen Content Coding By Skipping Unnecessary Checking of Intra Block Copy Mode Based on CU Activity and Gradient," *APSIPA Annual Summit and Conf. (APSIPA ASC)*, pp.1-5, Jeju, South Korea, Dec. 2016.

[6] S.-H. Tsang, Y.-L. Chan, and W.-C. Siu, "Hash Based Fast Local Search for Intra Block Copy (IntraBC) Mode in HEVC Screen Content Coding," *APSIPA Annual Summit Conf. (APSIPA ASC)*, pp.396-400, Hong Kong, China, Dec. 2015.

[7] X. Xu, X. Li, and S. Liu, "Current Picture Referencing in Versatile Video Coding," *IEEE Conf. Multimedia Info. Process. Retrieval (MIPR)*, pp. 26-31, San Jose, CA, USA, Mar. 2019.

[8] S.-H. Tsang, Y.-L. Chan, and W.-C. Siu, "Exploiting Inter-layer Correlations in Scalable HEVC for the Support of Screen Content Videos," *Int. Conf. Digital Signal Process. (DSP)*, pp.888-892, Hong Kong, China, Aug. 2014.

[9] Y. Sun, J. Lou, Y. Chao, H. Wang, V. Seregin, and M. Karczewicz, "Analysis of Palette Mode on Versatile Video Coding," *IEEE Conf. Multimedia Info. Process. Retrieval (MIPR)*, pp. 455-458, San Jose, CA, USA, Mar. 2019.

[10] Y. Huang *et al.*, "A VVC Proposal With Quaternary Tree Plus Binary-Ternary Tree Coding Block Structure and Advanced Coding Techniques," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 5, pp. 1311-1325, May 2020.

[11] K. Choi *et al.*, "Video Codec Using Flexible Block Partitioning and Advanced Prediction, Transform and Loop Filtering Technologies," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 5, pp. 1326-1345, May 2020.

[12] G. Bjontegaard, "Calculation of Average PSNR Differences Between RDCurves," VCEG, VCEG-M33, Austin, TX, USA, Mar. 2001.

[13] Y.-H. Chao, Y.-C. Sun, J. Xu, and X. Xu, "JVET common test conditions and software reference configurations for non-4:2:0 colour formats," JVET, JVET-R2013, Teleconference, Apr. 2020.

[14] VVC Test Model Version 9.2 VTM-9.2. [Online]. Available: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM.

[15] F. Bossen, X. Li, and K. Suehring, "AHG report: Test model software development (AHG3)," JVET-K0003, Ljubljana, Slovenia, Oct. 2018.

[16] Y. Fan, J. Chen, H. Sun, J. Katto, and M. Jing, "A Fast QTMT Partition Decision Strategy for VVC Intra Prediction," *IEEE Access*, vol. 8, pp. 107900-107911, Jun. 2020.

[17] T. Amestoy, A. Mercat, W. Hamidouche, D. Menard, and C. Bergeron, "Tunable VVC Frame Partitioning Based on Lightweight Machine Learning," *IEEE Trans. Image Process.*, vol. 29, pp. 1313-1328, 2020.

[18] T. Li, M. Xu, and R. Tang, "DeepQTMT: A Deep Learning Approach for Fast QTMT-based CU Partition of Intra-mode VVC," *arXiv preprint* arXiv:2006.13125, pp. 1-11, Jun. 2020.

[19] T. Fu, H. Zhang, F. Mu, and H. Chen, "Two-Stage Fast Multiple Transform Selection Algorithm for VVC Intra Coding," *Int. Conf. Multimedia Expo (ICME)*, pp. 61-66, Shanghai, China, Jul. 2019.

[20] W. Kuang, Y.-L. Chan, S.-H. Tsang, and W.-C. Siu, "DeepSCC: Deep Learning Based Fast Prediction Network for Screen Content Coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 7, pp. 1917-1932, Jul. 2020.

[21] W. Kuang, Y.-L. Chan, S.-H. Tsang, and W.-C. Siu, "Machine Learning Based Fast Intra Mode Decision for HEVC Screen Content Coding Via Decision Trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 5, pp. 1481-1496, May 2020.

[22] W. Kuang, Y.-L. Chan, S.-H. Tsang, and W.-C. Siu, "Online-Learning-Based Bayesian Decision Rule for Fast Intra Mode and CU Partitioning Algorithm in HEVC Screen Content Coding," *IEEE Trans. Image Process.,* vol. 29, no. 1, pp. 170-185, Jan. 2020.

[23] S.-H. Tsang, Y.-L. Chan, and W. Kuang, "Mode Skipping for HEVC Screen Content Coding via Random Forest," *IEEE Trans. Multimedia*, vol. 21, no. 10, pp. 2433-2446, Oct. 2019.

[24] W. Kuang, Y.-L. Chan, and S.-H. Tsang, "Efficient Intra Bitrate Transcoding for Screen Content Coding Based on Convolutional Neural Network," *IEEE Access*, vol. 7, pp. 107211-107224, Aug. 2019.

[25] S.-H. Tsang, Y.-L. Chan, W. Kuang, and W.-C. Siu, "Reduced-Complexity Intra Block Copy (IntraBC) Mode with Early CU Splitting and Pruning for HEVC Screen Content Coding," *IEEE Trans. Multimedia*, vol. 21, no. 2, pp. 269-283, Feb. 2019.

[26] W. Kuang, Y.-L. Chan, S.-H. Tsang, and W.-C. Siu, "Fast HEVC to SCC Transcoder by Early CU Partitioning Termination and Decision Tree Based Flexible Mode Decision for Intra-Frame Coding," *IEEE Access*, vol. 7, pp. 8773-8788, Jan. 2019.

[27] S.-H. Tsang, Y.-L. Chan, and W.-C. Siu, "Fast and Efficient Intra Coding Techniques for Smooth Regions in Screen Content Coding Based on Boundary Prediction Samples," *IEEE Int. Conf. Acoustics, Speech Signal Process.*, pp.1409-1413, Brisbane, Australia, Apr. 2015.

[28] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Conf. Comput. Vision Pattern Recognition (CVPR)*, Boston, MA, USA, pp. 3431-3440, Jun. 2015.

[29] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *Springer Int. Conf. Medical Image Comput. Computer-Assisted Intervention (MICCAI)*, pp. 234-241, Munich, Germany, Oct. 2015.

[30] S. Ioffe, and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *Int. Conf. Machine Learning (ICLR)*, pp. 448–456, Lille, France, Jul. 2015.

[31] M. Abadi, *et al.*, "TensorFlow: A system for large-scale machine learning," *USENIX Symp. Operating Syst. Design Implementation (OSDI)*, pp 265−283, Savannah, GA, USA, Nov. 2016.