IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 8, 2019, accepted July 23, 2019, date of publication August 5, 2019, date of current version August 19, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2933029

Efficient Intra Bitrate Transcoding for Screen Content Coding Based on Convolutional Neural Network

WEI KUANG, (Student Member, IEEE), YUI-LAM CHAN, (Member, IEEE), AND SIK-HO TSANG, (Member, IEEE)

Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong

Corresponding author: Yui-Lam Chan (enylchan@polyu.edu.hk)

This work was supported by the Hong Kong Research Grants Council under Research Grant PolyU 152069/18E.

ABSTRACT The Screen Content Coding (SCC) extension of High Efficiency Video Coding (HEVC) is developed to improve the coding efficiency of screen content videos. To meet the diverse network requirement of different clients, bitrate transcoding for SCC is desired. This problem can be solved by a conventional brute-force transcoder (CBFT) which concatenates an original decoder and an original encoder. However, it induces high computational complexity associated with the re-encoding part of CBFT. This paper presents a convolutional neural network based bitrate transcoder (CNN-BRT) for SCC. By utilizing information from both the decoder side and the encoder side, CNN-BRT makes a fast prediction for all coding units (CUs) of a coding tree unit (CTU) in a single test. At the decoder side, decoded optimal mode maps that reflect the optimal modes and CU partitions in a CTU is derived. At the encoder side, the raw samples in a CTU are collected. Then, they are fed to CNN-BRT to make a fast prediction. To imitate the optimal mode selection in the original re-encoding part, CNN-BRT involves a loss function that takes both of the sub-optimal modes and the final optimal modes into consideration. Compared with the HEVC-SCC reference software SCM-3.0, the proposed CNN-BRT reduces encoding time by 54.86% on average with a negligible Bjøntegaard delta bitrate increase of 1.01% under all-intra configuration.

INDEX TERMS Transcoding, screen content coding (SCC), fast algorithm, convolutional neural network.

I. INTRODUCTION

Screen content videos have gained popularity with the fast development of mobile and cloud technologies, and they have many applications such as online education, video conference with document sharing, remote desktop, and wireless display [1]. Screen content videos are captured from the display screens of various electronic devices, and they usually show a mixed content of camera-captured natural image blocks (NIBs) and computer-generated screen content blocks (SCBs). Compared with NIBs, SCBs have different characteristics, such as no sensor noise, large flat areas with a single color, repeated patterns in a frame and limited colors. To improve the coding efficiency of screen content videos, the Joint Collaborative Team on Video Coding (JCT-VC) launched Screen Content Coding (SCC) extension [2] on top of High Efficiency Video Coding (HEVC) [3] in 2014.

SCC is developed based on the main framework of HEVC. Besides the traditional Intra mode that has been included in HEVC, SCC additionally induces two important coding modes, intra block copy (IBC) [4], [5] and palette (PLT) [6], to address the new SCBs. With the observation that screen content videos contain many repeated patterns within the same frame, IBC performs motion estimation for the current coding unit (CU) in the reconstructed areas of the current frame. Since a SCB usually has limited colors, PLT mode is designed to encode it with several representative colors and an index map. As a result, SCC outperforms HEVC by achieving over 50% Bjøntegaard delta bitrate (BDBR) [7] reduction for typical screen content videos [2].

SCC is expected to be widely used for many screen content based applications due to its high coding efficiency. In the real application, a single video stream cannot meet the diverse

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

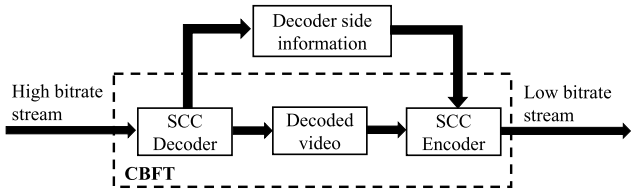

FIGURE 1. Structure of an efficient video transcoder.

network requirement of different clients. For example, a client under 3G environment prefers a video stream with a lower bitrate than a client with a Wi-Fi connection. A possible solution is to store several copies of a video with different quality levels on the server, and then sending the bitstream that well satisfies the network constraints. However, it significantly increases the storage cost in the server, and the pre-encoded video stream may not exactly match the network constraints. To solve this problem, a video can be encoded at high bitrate in a server and followed by a bitrate transcoder that bridges the server and the clients.

Bitrate transcoding is popular for digital video adaption and distribution, which transforms a high-bitrate input stream to a low-bitrate output stream within the same video format. This variable bitrate transcoding is the concern in this paper. A simple solution is to use a conventional brute-force transcoder (CBFT), which is concatenated by an original decoder and an original encoder. The input video is encoded using a low value of the quantization parameter (QP_1). CBFT first decodes the incoming bitstream to obtain the reconstructed video, and then the reconstructed video is completely re-encoded to a bitstream with lower bitrate by using a high value of QP_2. By introducing a parameter $\Delta QP = QP_2 - QP_1$, different levels of bitrate reduction can be achieved. The advantage of CBFT is that it provides high rate-distortion (RD) performance since it searches all CU partitions and mode candidates in the re-encoding process. However, the exhaustive search in the re-encoding part also induces very high computational complexity because of the flexible coding tree unit (CTU) partitioning structure and a large number of mode candidates. To solve this problem, an efficient transcoder should utilize information from both of the decoder and encoder sides to simplify the re-encoding part, as shown in Fig. 1.

In the literature, many transcoding algorithms have been proposed based on similar structures in Fig. 1, and they can be mainly divided into two categories: homogeneous transcoding and heterogeneous transcoding. Heterogeneous transcoding converts bitstream between different video formats such as MPEG-2 to H.264/AVC transcoding [8], [9], MPEG-2 to HEVC transcoding [10], H.264/AVC to HEVC transcoding [11], [12], HEVC to SCC transcoding [13]–[15], and SCC to HEVC transcoding [16]. Homogeneous transcoding refers to the conversion within the same format to meet new functionalities such as different bitrates [17], [18], different frame rates [19], [21], different spatial resolutions [22], and the insertion of error resilience layers [23]. Although the works in [17], [18] discuss the bitrate transcoding of H.264/AVC and HEVC, they are not optimal for SCC since the adoption of new modes makes the transcoding more complicated.

To reduce the computational complexity of SCC bitrate transcoding, one solution is to replace the original SCC encoder in CBFT by various fast SCC encoders [24]–[28]. In [24], a fast CU size decision method was proposed for stationary CUs in screen content videos. All modes are searched if the depth level of the current stationary CU is equal to its collocated CU. Otherwise, only PLT mode is checked for the stationary CU. In [25], three sets of decision tree-based classifiers were proposed. First, CUs are classified into NIBs and SCBs so that NIBs only check Intra mode while SCBs only check IBC and PLT modes. Then the CU partitioning process is early terminated for smooth NIBs and the direction of Intra mode is also predicted. In [26], incoming CUs are classified into NIBs and SCBs by analyzing CU content. Only Intra mode is checked for NIBs to reduce encoding time. However, all modes need to be checked for SCBs due to the low classification accuracy. Then, the bit cost and the depth information from the temporal and spatial neighboring CUs are utilized to early terminate CU partitions. In [27], Intra mode is firstly searched for all CUs with $2N\times2N$ prediction units (PUs) to collect some features. Based on these features and CU content, incoming CUs are classified into partitioning CUs and non-partitioning CUs. Partitioning CUs directly go to the next depth level while non-partitioning CUs terminate partitions in the current depth level. Furthermore, non-partitioning CUs are classified into NIBs and SCBs. Both PLT and IBC modes are searched for SCBs, while only Intra mode is checked for NIBs with $N\times N$ PUs. In [28], a sequential arrangement of decision trees was proposed to directly make mode classification for incoming CUs. The encoding time is further reduced since the different decision for PLT and SCB can be considered for a SCB. Although the complexity of the SCC encoder is reduced by these techniques, the reduction is still limited. First, the fast prediction rules in [24]–[28] rely on the assumption that the computer-generated SCBs are noise-free, but this assumption does not hold for decoded videos in the CBFT due to the lossy encoding and decoding. Second, they only utilize the information from the encoder side for computational complexity reduction but do not consider the information from the decoder side.

A more efficient solution is to design a transcoder using information from both the encoder side and the decoder side, as shown in Fig. 1. In the literature, there is only one work [29] studying the fast bitrate transcoding scheme of SCC using the structure in Fig. 1. It records the optimal mode, PU size, and CU size in the decoder side. Then several hand-crafted rules are derived to skip the unnecessary mode candidates by using the correlation between the current CU and the corresponding decoded CU. Besides, another set of hand-crafted rules are derived to skip partial PUs to further reduce the complexity of the re-encoding part. Although it shows better performance than the aforementioned fast SCC

encoder techniques [24]–[28], it still leaves large room for further improvement. Since only limited hand-crafted rules are derived in [29], it may not handle the complicated situation in SCC transcoding well. Therefore, learning based approaches utilizing a large amount of training data are desired.

In this paper, we propose a convolutional neural network (CNN) based bitrate transcoder for SCC (CNN-BRT). With the help of CNN, CNN-BRT can extract extensive features automatically by utilizing large-scale training data. Specifically, CNN-BRT makes fast predictions in the CTU level, which means all CUs in a CTU get their predicted labels in a single test. To alleviate the burden of implementation, we integrate the mode decision and CU partitioning decision into a single network, where skipping a CU level is treated as a special case of mode decision. Therefore, four classes are defined for a CU, which are ω_{Intra}, ω_{IBC}, ω_{PLT}, $\omega_{Allskip}$, i.e. selecting Intra, IBC, PLT and skipping all modes for the CU. At the decoder side, the decoded optimal mode maps that present the optimal modes and partitioning structure of a CTU is extracted. At the encoder side, the raw samples of a CTU associated with the decoded optimal mode map are fed to CNN-BRT to make a fast prediction. To imitate the optimal mode selection in the original re-encoding part, CNN-BRT involves a loss function that considers both the sub-optimal modes and the final optimal modes. The difference between our contributions and the related schemes can be summarized as 1) unlike the fast SCC encoders in [24]–[28] that only consider information from the encoder side, CNN-BRT additionally utilizes information from the decoder side. 2) Compared with the fast SCC bitrate transcoder [29], CNN-BRT employs extensive learnable parameters to replace the limited hand-crafted rules. 3) Unlike [24]–[29] only focus on the final optimal modes, CNN-BRT additionally utilizes the sub-optimal modes to guide the extraction of feature maps. With these advantages, CNN-BRT outperforms the existing approaches [24]–[29] by a large margin.

The rest of this paper is organized as follows. Section II presents the review and analysis of bitrate transcoding. Section III presents the proposed efficient bitrate transcoder CNN-BRT. The experimental results are presented in Section IV to verify the performance of the proposed CNN-BRT. Finally, Section V concludes the paper. Table 1 lists the abbreviations used in this paper.

TABLE 1. List of abbreviations.

NIB	Natural Image Block
SCB	Screen Content Block
JCT-VC	Joint Collaborative Team on Video Coding
SCC	Screen Content Coding
HEVC	High Efficiency Video Coding
IBC	Intra Block Copy
PLT	Palette
CU	Coding Unit
BDBR	Bjøntegaard Delta Bitrate
CBFT	Conventional Brute-Force Transcoder
QP	Quantization Parameter
RD	Rate-Distortion
CTU	Coding Tree Unit
PU	Prediction Unit
CNN	Convolutional Neural Network
CNN-BRT	CNN Based Efficient Bitrate Transcoder
CTC	Common Test Condition
TGM	Text and Graphics with Motion
M	Mixed Content
A	Animation
CC	Camera-Captured Content
AI	All-Intra

II. REVIEW AND ANALYSIS OF SCC BITRATE TRANSCODING

A. REVIEW ON INTRA PREDICTION IN SCC

SCC inherits the quadtree-based block partitioning scheme from HEVC. Each video frame is divided into many non-overlapped CTUs of 64×64 pixels. For each CTU, a recursive partitioning process is performed. A CU of $2N\times2N$ pixels can be partitioned into four smaller CUs of $N\times N$ pixels, and the value of $2N$ can be 64, 32, and 16 in SCC. Therefore, the size of a CU can be 64, 32, 16, and 8, and they are referred to as CUs at the depth level of 0, 1, 2, and 3, respectively. In total, a CTU have 85 different CUs (1, 4, 16, and 64 CUs at the depth level of 0, 1, 2, and 3, respectively). To select the sub-optimal mode of each CU, an exhaustive mode searching process is adopted. The RD cost is calculated for each mode x, $x \in$ {Intra, IBC, PLT}, as

$$J_x = D_x + \lambda \times R_x \tag{1}$$

where λ is a Lagrange multiplier, D_x and R_x are the distortion and bit cost of the CU being coded with a mode x. The sub-optimal mode associated with a CU is selected as the one with the smallest value of J_x. To determine the optimal CTU structure, RD costs are compared between a CU and its four sub-CUs. The partitions of a CU are removed if the sum of the RD costs of its four sub-CUs is larger than the RD cost of the CU. Otherwise, the CU is divided into four sub-CUs. Finally, the best combination of CU sizes within a CTU is determined as the one with the smallest sum of RD costs, and the corresponding sub-optimal modes become the final optimal modes of the CTU.

To model the mode decision of a CTU, a sub-optimal mode map and an optimal mode map in each depth level can be derived. Fig. 2 shows the representation of the mode decision of a CTU. Each element in a mode map contains a class label of 0 to 3, which represent $\omega_{Allskip}$, ω_{Intra}, ω_{IBC}, and ω_{PLT}, respectively. The recursive partitioning process is started at the CTU, as shown in Fig. 2(b). By comparing the RD costs among Intra, IBC and PLT modes, a sub-optimal mode is selected for each CU. After calculating the RD costs for all CUs, the sub-optimal modes of the CTU are represented by four sub-optimal mode maps, as shown in Fig. 2(a). Noted that the sub-optimal mode maps only contain class labels of 1 to 3 since a CU can only select a sub-optimal mode from class ω_{Intra}, ω_{IBC}, and ω_{PLT}. Then the RD costs are compared

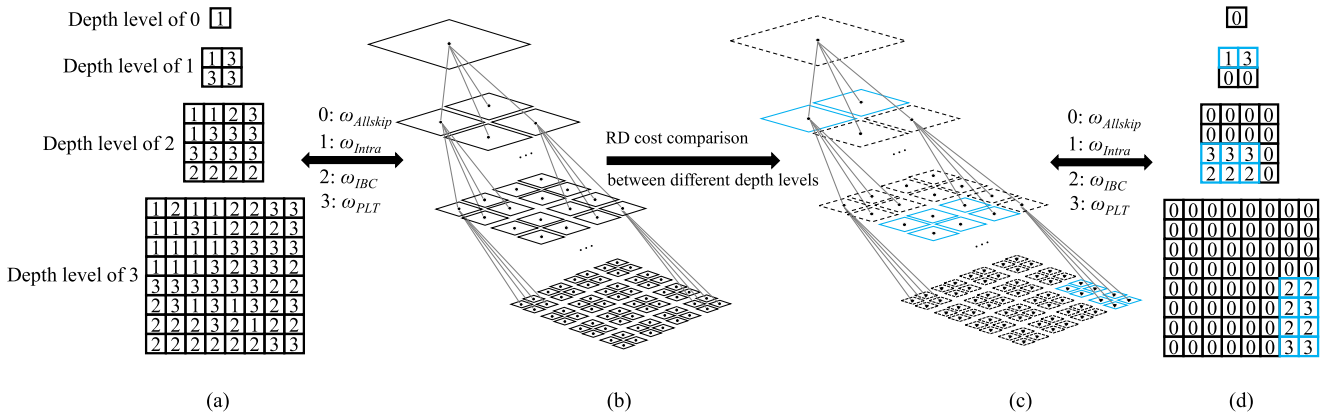

FIGURE 2. Representation of the mode decision of a CTU: (a) sub-optimal mode maps of a CTU; (b) recursive partitions in a CTU; (c) optimal partitioning structure of a CTU (denoted by blue squares); (d) optimal mode maps of a CTU.

between a CU and its four sub-CUs to decide whether the CU should be partitioned or not. By selecting the structure with the smallest sum of RD costs, the optimal partitioning structure of the CTU is shown in Fig. 2(c), which are denoted by blue squares. Then the optimal mode maps of the CTU are shown in Fig. 2(d). Since the CTU in Fig. 2 is not coded as a single 64×64 CU, the label of the mode map in the depth level of 0 is set to 0, which means that all modes are skipped in the depth level of 0. Then, the CTU contains two 32×32 CUs coded by Intra mode and PLT mode so that the corresponding labels of the mode map in the depth level of 1 are set to 1 and 3. The remaining labels of the mode map are set to 0 to denote that they are not coded as 32×32 CUs. This process is repeated until the four optimal mode maps are all generated. By using this representation, the optimal mode decision in a CTU can be modeled as a classification problem, and the computational complexity of CBFT is reduced significantly if the classification problem can be solved.

TABLE 2. Bitrate reduction by applying different values of Δ QP.

Sequences		Input Bitrate (Kbps)	Bitrate Reduction (%) ΔQP = 2	ΔQP = 4	ΔQP = 6
MissionControlClip2	$QP_1 = 22$	100993.66	15.47	30.53	41.91
	$QP_1 = 27$	65486.63	18.67	33.44	44.79
	$QP_1 = 32$	40709.48	19.25	34.23	46.20
	$QP_1 = 37$	24844.19	21.14	37.54	50.73
Map	$QP_1 = 22$	40424.86	13.14	25.61	36.71
	$QP_1 = 27$	27974.18	17.18	31.03	41.82
	$QP_1 = 32$	18054.25	17.48	31.74	42.65
	$QP_1 = 37$	11462.95	18.25	34.91	50.03
Robot	$QP_1 = 22$	37706.46	24.48	47.74	61.72
	$QP_1 = 27$	18170.50	31.01	52.02	64.22
	$QP_1 = 32$	8026.52	29.64	49.61	62.18
	$QP_1 = 37$	3764.81	30.82	50.11	62.40
EBURainFruits	$QP_1 = 22$	88085.76	18.03	36.43	48.77
	$QP_1 = 27$	52148.74	20.23	38.58	51.43
	$QP_1 = 32$	29690.58	21.57	40.95	54.26
	$QP_1 = 37$	16300.6	24.32	45.10	58.47
Average			21.29	38.72	51.14

B. ANALYSIS OF SCC BITRATE TRANSCODING

To analyze the impact of different values of ΔQP on the bitrate of the output video stream, we performed the bitrate transcoding for sequences in the common test conditions (CTC) [30] by using CBFT. In CTC, sequences are classified into four categories according to their content: text and graphics with motion (TGM), mixed content (M), animation (A) and camera-captured content (CC). A representative sequence is selected from each category for the analysis, and they are shown in Fig. 3. Each sequence was encoded and decoded by the SCC reference software HM-16.2+SCM-3.0 [31] with $QP_1 \in \{22, 27, 32, 37\}$ under all-intra (AI) configuration, and then the decoded sequence was re-encoded by HM-16.2+SCM-3.0 with QP_2, where $QP_2 = QP_1 + \Delta QP$, and $\Delta QP \in \{2, 4, 6\}$. It is noted that the value of ΔQP is limited to 6 in the proposed transcoding scheme. As suggested by [18], further bitrate reduction should be achieved by other transcoding methods such as temporal transcoding and spatial transcoding. Table 2 shows the bitrate reduction by applying different values of ΔQP. It is observed that a larger value of ΔQP leads to a larger reduction of bitrate. By setting ΔQP to 2, 4, and 6, 21.29%, 38.72% and 51.14% bitrates are reduced on average, respectively.

Table 3 presents the percentage of areas that share the same CU sizes and optimal modes between the high-bitrate input stream and the low-bitrate output stream under QP_1 of 32 and $\Delta QP \in \{2, 4, 6\}$, and similar results are observed for other values of QP_1. It is observed that there exists a strong correlation of the CU size and optimal mode between the high-bitrate stream and the low-bitrate stream, and the strength of the correlation increases as the value of ΔQP decreases. On average, 79.16%, 70.85%, and 66.65% areas share the same optimal modes between the high-bitrate and low-bitrate streams for ΔQP of 2, 4, and 6, respectively. Based on this observation, the information from the decoder side is useful to speed up the re-encoding process.

Furthermore, the re-encoding time distribution of each mode is also analyzed with $QP_1 \in \{22, 27, 32, 37\}$ and ΔQP

MissionControlClip2 (M) Map (TGM)

Robot (A) EBURainFruits (CC)

FIGURE 3. Examples of screen content sequences.

TABLE 3. Percentage of areas with the same CU sizes and optimal modes between the input and output streams.

Categories	Sequences	ΔQP = 2 (%)	ΔQP = 4 (%)	ΔQP = 6 (%)
TGM	ChineseEditing	78.18	71.53	64.56
	Console	74.42	50.78	48.52
	Desktop	85.88	83.26	81.47
	FlyingGraphics	81.88	75.81	69.76
	Map	77.38	70.35	64.63
	Programming	81.31	76.86	72.69
	SlideShow	91.39	87.21	85.01
	WebBrowsing	83.79	77.48	76.28
M	BasketballScreen	82.37	78.85	75.20
	MissionControlClip2	78.47	72.10	67.66
	MissionControlClip3	80.21	75.72	71.07
A	Robot	72.22	59.65	53.05
CC	EBURainFruits	75.12	61.83	56.76
	Kimono1	65.62	50.44	46.50
Average (ALL)		79.16	70.85	66.65

$\in \{2, 4, 6\}$. The average results are shown in Table 4 for each sequence. It is observed that Intra mode takes the largest percentage of re-encoding time, which is 58.53%, followed by IBC mode and PLT mode, which are 35.64% and 6.04%, respectively. Therefore, it is very efficient to skip unnecessary Intra mode and IBC mode, while less re-encoding time reduction can be obtained by skipping unnecessary PLT mode.

III. PROPOSED EFFICINET BITRATE TRANSCODER CNN-BRT

The computational complexity of the CBFT mainly comes from the exhaustive mode searching process in the re-encoding part. To simplify the transcoding process, we model the optimal mode selection in the re-encoding part as a classification problem. Since there exists a strong correlation of the optimal mode between the high-bitrate input stream and the low-bitrate output stream, information from the decoder side is collected to speed up the re-encoding process. Instead of deriving limited hand-crafted rules based on humans' observation as in [29], a CNN based transcoding approach is proposed. Since it contains many trainable parameters and learns extensive features, better performance can be provided.

TABLE 4. Re-encoding time distribution of each mode.

Categories	Sequences	Intra (%)	IBC (%)	PLT (%)
TGM	ChineseEditing	49.00	45.52	5.48
	Console	61.66	33.12	5.22
	Desktop	61.59	32.24	6.17
	FlyingGraphics	51.59	42.05	6.36
	Map	49.67	44.79	5.54
	Programming	58.14	34.79	7.07
	SlideShow	74.47	19.99	5.54
	WebBrowsing	61.44	32.43	6.13
M	BasketballScreen	53.61	40.03	6.36
	MissionControlClip2	56.19	36.69	7.12
	MissionControlClip3	55.62	37.21	7.17
A	Robot	58.23	34.91	6.86
CC	EBURainFruits	57.10	38.39	4.51
	Kimono1	68.24	26.78	4.98
Average (ALL)		58.53	35.64	6.04

A. NETWORK STRUCTURE

To improve the prediction efficiency, CNN-BRT is designed to predict the modes of all CUs in a CTU. For each CU, CNN-BRT outputs the probabilities of selecting each class, i.e., $P(\omega_{Intra})$, $P(\omega_{IBC})$, $P(\omega_{PLT})$, and $P(\omega_{Allskip})$. If the probability of selecting a mode x, $x \in$ {Intra, IBC, PLT}, is smaller than a threshold, x is skipped so that the encoding time is reduced. The structure of CNN-BRT is shown in Fig. 4, and the kernel sizes and feature map dimensions are also presented. CNN-BRT mainly contains convolutional layers and deconvolutional layers for feature extraction, and concatenated layers are adopted to join the extracted features maps and the decoder side information. To imitate the optimal mode selection in the original re-encoding part, CNN-BRT involves the auxiliary classifiers to predict the sub-optimal modes and the final classifiers to predict the final optimal modes for each CU. The components of CNN-BRT are described as follows.

1) CONVOLUTIONAL LAYERS

CNN-BRT contains 13 convolutional layers. The luminance component of a decoded CTU is input to CNN-BRT and it firstly goes through five convolutional layers, i.e., conv1–conv5, for feature extraction. conv1 adopts the kernel size of 4×4 and conv2–conv5 adopt the kernel size of 2×2. Since the CTU partitioning structure contains non-overlapping CUs, the strides of conv1–conv5 are set to the width of their kernel sizes to perform non-overlapping convolutions. The width and height of the feature maps in conv1–conv5 are down-sampled by half in the next layer because of the non-overlapping convolutions, and we also double the channel number of feature maps in the next layer. It is noted that the sizes of the output feature maps of conv2–conv5 in Fig. 4 are equal to the number of CUs in the depth levels of 3 to 0 in Fig. 2, which are 8×8, 4×4, 2×2, and 1×1, respectively. By using this arrangement, the receptive

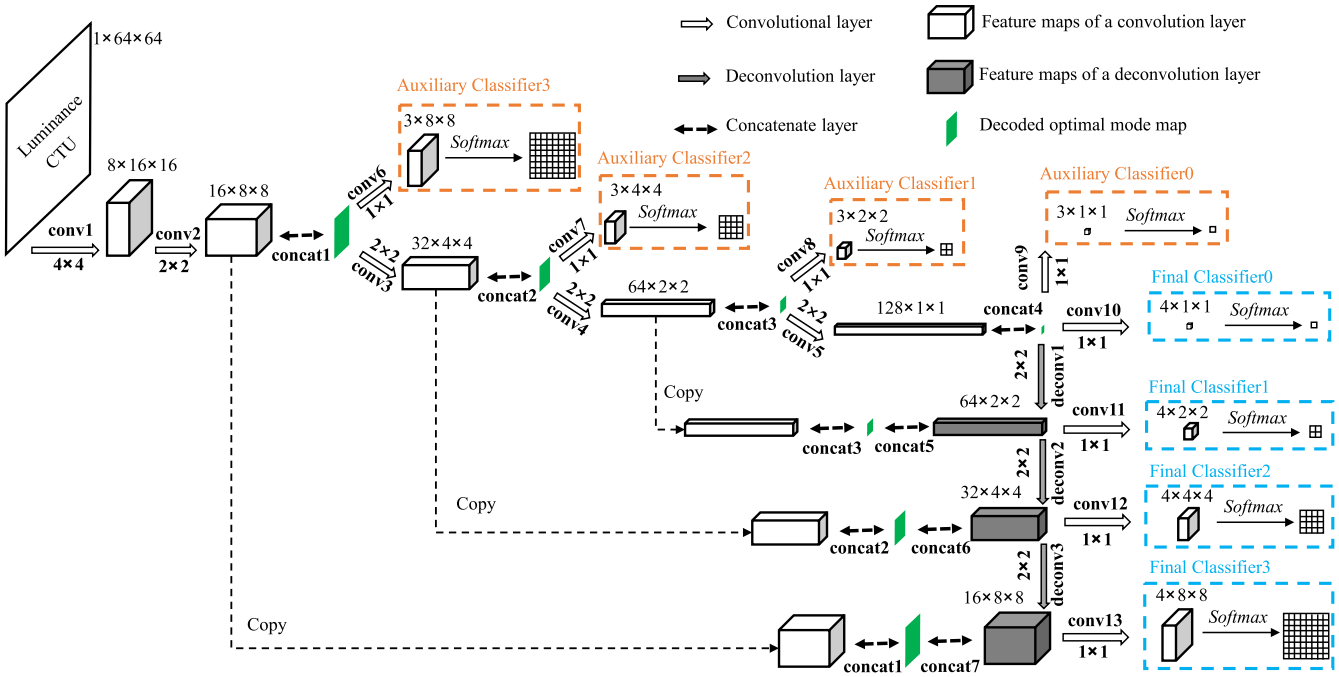

FIGURE 4. Network structure of CNN-BRT.

field of each element in a feature map is always equal to a CU size in the corresponding depth level, so that conv2–conv5 can extract local features for CUs in the depth levels of 3 to 0 without the influence of other CUs. Another two sets of convolutional layers, i.e., conv6–conv9 and conv10–conv13, are adopted to incorporate the outputs of the concatenating layers and generate the feature maps in auxiliary classifiers and final classifiers.

2) DECONVOLUTIONAL LAYERS

CNN-BRT contains three deconvolutional layers, deconv1–deconv3, to enlarge the feature maps of conv5 using the kernel size of 2×2 and stride of 2. Since the receptive field of each element in the feature maps of conv5 is the entire CTU, the receptive field of each node in the feature maps of deconv1–deconv3 also becomes the entire CTU. Therefore, deconv1–deconv3 can extract global features for CUs in the depth levels of 1 to 3 by introducing the influence of other CUs. At each feature map enlarging step, the channel number of the feature maps is halved. Finally, the global feature maps and local feature maps have the same dimension for each depth level.

3) CONCATENATING LAYERS

To join the decoder side information and the encoder side features, the concatenating layers concat1–concat4 concatenate the feature maps of conv2–conv5 to the corresponding decoded optimal mode maps before going to the next layer. Besides, the outputs of concat1–concat3 are further concatenated to the corresponding global feature maps by concat5–concat7 to join the local feature maps and global feature maps.

4) AUXILIARY CLASSIFIERS AND FINAL CLASSIFIERS

As reviewed in Section II.A, a SCC encoder first decides the sub-optimal mode of a CU by its local content and then decides the final optimal mode of a CU by comparing it with CUs in other depth levels. To imitate the original SCC re-encoding part, CNN-BRT includes both of the auxiliary and final classifiers, and the softmax function is used to predict the probability for each class. Since the receptive field of each element in the feature maps of conv6–conv9 is a local CU, Auxiliary Classifier0–Auxiliary Classifier3 are designed to predict the sub-optimal modes for CUs in the depth levels of 0 to 3, respectively. Comparatively, the receptive field of each element in the feature maps of conv10–conv13 is the entire CTU. Therefore, Final Classifier0–Final Classifier3 are designed to predict the final optimal modes for CUs in the depth level 0 to 3, respectively. Noted that the auxiliary classifiers only predict the labels of ω_{Intra}, ω_{IBC}, and ω_{PLT} since a CU can only select a sub-optimal mode from Intra, IBC, and PLT, while the final classifiers predict labels of ω_{PLT}, ω_{Intra}, ω_{IBC}, and $\omega_{Allskip}$ since a CU can select an optimal mode from Intra, IBC and PLT, or skip all modes and be coded in other depth levels. The auxiliary classifiers can guide the learning process of the training parameters by considering the loss of the sub-optimal mode prediction, and their impact will be investigated in the ablation study in Section IV.A. As seen in Fig. 4, the auxiliary classifiers and the final classifiers output 1, 4, 16, and 64

labels for CUs in the depth levels of 0 to 3, respectively, in accordance with the hierarchical CTU partitioning structure in Fig. 2. Therefore, the 85 CUs in a CTU get their predictions in a single test by running CNN-BRT.

In CNN-BRT, each convolutional or deconvolutional layer is followed by the rectified linear unit (ReLU) activation function, except for conv6–conv13, where softmax is utilized to generate the output labels.

B. TRAINING STRATEGY

To train the proposed CNN based network in CNN-BRT, 22 sequences are carefully selected from [32]–[37] to cover various video content, and they are divided into training sequences and validation sequences. Based on the same content classification criterion of CTC, we also classify those sequences into the four categories of TGM, M, A, and CC, and they are shown in Table 5. It should be noted that sequences in [32]–[37] are not included in CTC [30] so that the trained CNN-BRT does not bias the testing sequences in CTC [30], which are used to evaluate the performance of CNN-BRT and compare with other methods [24]–[29]. To generate the training data, 50 frames were extracted with an equal interval from each training sequence. They were firstly encoded by the original HM-16.2+SCM-3.0 with QP_1 of 22, 27, 32 and 37 to generate the high-bitrate streams. Second, the high-bitrate streams were decoded to get the decoded videos with the decode optimal mode maps. Finally, the decode videos were re-encoded by HM-16.2+SCM-3.0 with QP_2 to generate the low-bitrate streams while collecting the luminance component and the ground-truth labels of each CTU. In this way, 1,400,000 CTUs were collected as the training data, and a single model was obtained by using mixed training data from the four QPs.

The model of CNN-BRT was trained by Caffe [38], and a GPU of GeForce GTX 1080 Ti was used to accelerate the training process. In the training process, the trainable parameters in CNN-BRT are initialized by the "msra" filter [39], and the Adam optimizer [40] is adopted to update the trainable parameters. The learning rate policy of "step" is used with the base learning rate of 0.01, and the learning rate is multiplied by 0.1 every 10,000 iterations. To alleviate the overfitting problem, a weight decay of 0.005 is also adopted.

To imitate the original SCC re-encoding part, a loss function is derived to optimize both the sub-optimal mode decision of the auxiliary classifiers and the optimal mode decision of the final classifiers. The loss function of an *i-th* training sample in a batch is defined as the weighted summation of the losses from the final classifiers and the auxiliary classifiers

$$l_i = l_{Final} + W \cdot l_{Auxiliary} \tag{2}$$

where l_{Final} and $l_{Auxiliary}$ are the losses from the final classifiers and the auxiliary classifiers, respectively. A weighted factor W is adopted to balance these losses. Since the labels of the final classifiers are directly used for fast prediction, W is set to 0.5 so that the loss function is more focused on the final classifiers. $l_{Auxiliary}$ and l_{Final} are defined as the sum

TABLE 5. Training and validation sequences for CNN-BRT.

Categories		Sequences	Resolution	No. of Frame	Frame Rate (Hz)
Training sequences	TGM	ChineseDocumentEditing	1920×1080	300	30
		ClearTypeSpreadsheet	1920×1080	300	30
		CadWaveform	1920×1080	200	20
		PcbLayout	1920×1080	200	20
		PptDocXls	1920×1080	200	20
		RealTimeData	1920×1080	600	60
		VideoConferencing-DocSharing	1280×720	300	30
		WordEditing	1920×1080	600	60
	M	BigBuck	1920×1080	400	60
		KristenAndSaraScreen	1920×1080	600	60
		KimonoError2	2560×1440	500	60
		MissionControlClip1	2560×1440	600	60
	A	Viking	1280×720	300	30
	CC	EBULupoCandlelight	1920×1080	250	50
Validation sequences	TGM	BitstreamAnalyzer	1920×1080	300	30
		CircuitLayoutPresentation	1920×1080	300	30
		Doc	1280×720	500	10
		EnglishDocumentEditing	1920×1080	300	30
		Web	1280×720	500	10
	M	KimonoError1	2560×1440	500	60
	A	CGTwistTunne	1920×1080	300	30
	CC	Seeking	1920×1080	250	50

of cross-entropy over the 85 labels in a CTU, and they are represented by

$$l_{Auxiliary} = \sum_{j=0}^{84} f(\omega^{Auxiliary,j}, \hat{\omega}^{Auxiliary,j}) \tag{3}$$

$$l_{Final} = \sum_{j=0}^{84} f(\omega^{Final,j}, \hat{\omega}^{Final,j}) \tag{4}$$

where $\omega^{Auxiliary,j}$ and $\omega^{Final,j}$ are the ground-truth classes of the auxiliary classifiers and the final classifiers for the *j-th* CU. $\hat{\omega}^{Auxiliary,j}$ and $\hat{\omega}^{Final,j}$ are the predicted classes of the auxiliary classifiers and the final classifiers for the *j-th* CU. $f(\cdot,\cdot)$ represents the cross-entropy function between the ground-truth class and the predicted class, and it is represented as

$$f\left(\omega, \hat{\omega}\right) = -\sum\nolimits_c y\left(\omega_c = \omega\right) log(P(\omega_c = \hat{\omega})) \tag{5}$$

where c denotes the class index, and $c \in \{Intra, IBC, PLT, Allskip\}$. $y\left(\omega_c = \omega\right)$ is 1 if ω_c is the same as the ground-truth class ω, otherwise, $y\left(\omega_c = \omega\right)$ is 0. $P(\omega_c = \hat{\omega})$ denotes the probability that ω_c is the same as the predicted class $\hat{\omega}$. The loss function for all training samples in a batch is derived by averaging the loss of each sample, and it is written as

$$L = \frac{1}{N} \sum\nolimits_{i=1}^{N} l_i \tag{6}$$

where N is the batch size, and it is set to 1024. The training loss of CNN-BRT is shown in Fig. 5. It is observed that the training process of CNN-BRT converges very fast, and we terminate the training after 50,000 iterations.

C. FAST RE-ENCODING

After generating the model of CNN-BRT, it is invoked in HM-16.2+SCM-3.0 by the DNN tool of OpenCV 3.4.1 for

TABLE 6. Performance of the proposed CNN-BRT for validation sequences with different values of α.

Sequences	α=0.03		α=0.05		α=0.07		α=0.09	
	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)
BitstreamAnalyzer	0.42	-51.58	0.61	-60.55	0.88	-62.74	1.14	-66.35
CircuitLayoutPresentation	0.47	-42.57	1.39	-52.34	2.25	-59.71	2.92	-66.30
Doc	1.00	-46.55	1.29	-56.10	1.40	-59.08	1.64	-62.54
EnglishDocumentEditing	1.33	-36.42	0.81	-44.28	0.47	-45.12	0.33	-45.84
Web	0.16	-48.23	0.33	-61.89	0.72	-64.68	0.84	-68.23
KimonoError1	0.68	-40.80	1.14	-50.18	1.43	-53.11	1.62	-56.89
CGTwistTunne	0.61	-42.55	0.42	-57.09	0.90	-61.79	1.32	-66.19
Seeking	0.24	-56.38	0.48	-63.34	0.75	-66.18	1.01	-68.09
Average	0.61	-45.64	0.81	-55.72	1.10	-59.05	1.35	-62.55

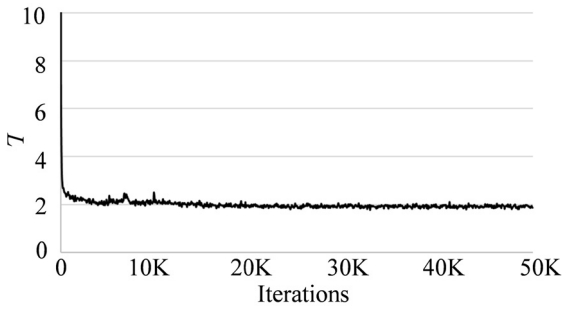

FIGURE 5. Training loss of CNN-BRT alongside iterations.

fast re-encoding. For each CTU, the final classifiers in CNN-BRT output the 85 labels for 85 CUs in it, and each label contains the probabilities of four classes, i.e., $P(\omega_c)$, $c \in \{Allskip, Intra, IBC, PLT\}$. Unlike the fast algorithms [24]–[29] that proposed different models for making fast mode decision and fast CU partitioning decision separately, CNN-BRT integrates the fast mode decision and fast CU partitioning decision into a single model by including the class $\omega_{Allskip}$. In the fast re-encoding phase, a threshold α is used to decide whether a CU needs to check a mode $x, x \in \{Intra, IBC, PLT\}$. If the probability of checking a mode x is smaller than the value of α, i.e., $P(\omega = x) < \alpha$, the mode x is regarded as unnecessary, and it is skipped to speed up the transcoder. Furthermore, if the probabilities of selecting Intra, IBC and PLT modes are all smaller than α, $\omega_{Allskip}$ will be selected, which means that the current CU size can be skipped. Before a CU continues the partitioning process, the labels of the CUs in the deeper depth levels are analyzed to perform the early CU partitioning decision. If an area of a CU always selects the class $\omega_{Allskip}$ in all deeper depth levels, the CU cannot be encoded if it continues partitioning. Therefore, CU partitions are early terminated to avoid unnecessary computation.

IV. EXPERIMETNAL RESULTS

For the simplicity of comparison, the proposed CNN-BRT has been implemented in the same reference software as in the only work of fast SCC bitrate transcoding [29], HM-16.2+SCM-3.0, and the proposed Caffe model can be found on our website [41]. Extensive experiments were conducted on a HP EliteDesk 800 G1 computer with a 64-bit Microsoft Windows 10 OS running on an Intel Core i7-4790 CPU of 3.6 GHz and 32.0 GB RAM. All experiments were conducted under AI configuration and CTC [30]. All test sequences were firstly encoded by HM-16.2+SCM-3.0 with $QP_1 \in \{22, 27, 32, 37\}$, and then the decoded sequences were re-encoded by a transcoder with the $QP_2 = QP_1 + \Delta QP$, $\Delta QP \in \{2, 4, 6\}$. By using this arrangement, three groups of QP_2 are tested for each sequence, i.e., $\{24, 29, 34, 39\}$, $\{26, 31, 36, 41\}$, and $\{28, 33, 38, 43\}$ in accordance with ΔQP of 2, 4, and 6, respectively. The coding efficiency and re-encoding time of the proposed CNN-BRT were compared with CBFT, and they are measured by BDBR and re-encoding time increase, Δ Time, in percentage (%). To calculate Δ Time, the inference time of the CNN model and the re-encoding time are both included. To calculate BDBR, the final transcoded video is compared to the uncompressed video. First, various ablation experiments were performed to decide the optimal structure of CNN-BRT by using the validation sequences in Table 5. Second, the performance of CNN-BRT was investigated from various aspects by using the testing sequences in CTC and it is compared with existing fast methods [24]–[29].

A. ABLATION STUDY

In this sub-section, various ablation experiments were conducted to investigate the performance of CNN-BRT by using the validation sequences shown in Table 5, and the average results with the three groups of QP_2 for different ablation experiments are presented in Table 6 and Table 7.

1) THRESHOLD DETERMINATION

In the fast re-encoding stage, a confidence threshold α is set to decide whether to check a mode x, $x \in$ {Intra, IBC, PLT}, and it controls the trade-off between Δ Time and BDBR. If α is set to a larger value, more mode candidates are skipped, and it leads to more re-encoding time reduction with a larger increase in BDBR. Otherwise, a smaller value of α results in less re-encoding time reduction with a smaller increase in BDBR. Therefore, the proposed CNN-BRT is complexity scalable. Table 6 shows the performance of CNN-BRT with α from 0.03 to 0.09. It is observed that CNN-BRT provides 45.64%–62.55% encoding time reduction with BDBR increased by 0.61%–1.35%. In the following sub-sections,

TABLE 7. Performance comparison of CNN-BRT and other possible structures.

Sequences	Only local feature		Only global feature		Removal of decoded optimal mode maps		Re-arrangement of decoded optimal mode maps		Removal of auxiliary classifiers		Proposed CNN-BRT	
	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)
BitstreamAnalyzer	1.64	-61.33	0.85	-52.07	3.90	-46.45	1.24	-47.53	0.77	-57.94	0.61	-60.55
CircuitLayoutPresentation	2.40	-56.50	1.18	-51.29	4.45	-34.38	1.58	-40.33	0.98	-51.99	1.39	-52.34
Doc	1.79	-55.26	2.44	-52.23	2.65	-40.43	1.57	-48.17	0.65	-52.49	1.29	-56.10
EnglishDocumentEditing	3.39	-49.61	1.26	-39.85	3.99	-36.24	2.33	-41.32	2.13	-42.01	0.81	-44.28
Web	2.25	-59.84	0.91	-57.78	4.94	-45.57	0.70	-51.41	0.43	-56.58	0.33	-61.89
KimonoError1	1.98	-43.86	1.51	-46.86	1.36	-38.08	1.35	-42.13	1.20	-46.93	1.14	-50.18
CGTwistTunne	2.33	-60.28	1.26	-52.80	1.94	-35.49	0.86	-44.98	1.40	-52.30	0.42	-57.09
Seeking	0.83	-40.86	0.53	-60.90	0.71	-48.88	0.43	-48.56	0.58	-63.39	0.48	-63.34
Average	2.08	-53.44	1.24	-51.72	2.99	-40.69	1.26	-45.55	1.02	-52.95	0.81	-55.72

α is set to 0.05 for further discussions, where 55.72% encoding time is reduced with a 0.81% increase in BDBR.

2) UTILIZING ONLY LOCAL FEATURES AND ONLY GLOBAL FEATURE

As shown in Fig. 4, CNN-BRT utilizes convolutional layers conv2–conv5 and deconvolutional layers deconv1–conv3 to extract local feature maps and global feature maps in a CTU, respectively. Then the local feature maps and the global feature maps are concatenated together to feed to the final classifiers. To show the importance of this design, experiments were performed to decouple the local feature maps and global feature maps by removing concat5–concat7. First, only the feature maps of conv2–conv5 are fed to the final classifiers so that only local feature maps are utilized. Second, only the feature maps of conv5 and deconv1–deconv3 are fed to the final classifiers so that only global feature maps are utilized. Their results are shown in Table 7, and the performance of the proposed CNN-BRT with α of 0.5 is also shown in this table for the convenience of comparison. It is observed that utilizing only local features provides a similar re-encoding reduction to the proposed CNN-BRT, but it induces a much higher increase in BDBR, which is 2.08%. On the other hand, utilizing only global features also shows worse performance than the proposed CNN-BRT, which provides a 51.72% re-encoding time reduction with BDBR increased by 1.24%. Therefore, concatenating the local features and the global features helps to improve the performance of the proposed CNN-BRT.

3) REMOVAL OF DECODED OPTIMAL MODE MAPS

To investigate the importance of the decoded optimal mode maps, experiments were performed by removing it from the proposed CNN-BRT, and the results are shown in Table 7. It is observed that the removal of the decoded optimal mode maps provides inferior performance than the proposed CNN-BRT, where 40.69% re-encoding time is reduced with BDBR increased by 2.99%. Therefore, the decoded optimal mode maps are important to the proposed CNN-BRT.

4) RE-ARRANGEMENT OF DECODED OPTIMAL MODE MAPS

In the proposed CNN-RBT, the decode optimal mode maps are concatenated to the local feature maps of conv2–conv5 before being fed to the next layer and the auxiliary classifiers. Therefore, they have an impact on the feature map generation of the next layer, the auxiliary classifiers, and the final classifiers. Alternatively, they can be input to CNN-BRT after all feature maps are generated, where they are concatenated to the local feature maps and the global feature maps before going to the final classifiers. By re-arranging the decode optimal mode maps, they only have an impact on the final classifiers, and the results are shown in Table 7. It is observed that re-arranging the feature maps provides worse performance than the proposed CNN-BRT, where 45.55% encoding time is reduced with BDBR increased by 1.26%. Therefore, inputting the decode optimal mode maps into CNN-BRT in an early stage can help its performance.

5) REMOVAL OF AUXILIARY CLASSIFIERS

In the proposed CNN-BRT, the final classifiers and the auxiliary classifiers are both included to imitate the mode decision of the original re-encoding part. The auxiliary classifiers utilize the local features to predict the sub-optimal mode while the final classifiers utilize both the local features and the global features to predict the final optimal mode. Alternatively, the auxiliary classifiers can be removed so that only final classifiers guide the learning process of the training parameters. The results are shown in Table 7. It is observed that the removal of the auxiliary classifiers provides inferior performance than the proposed CNN-BRT, where 52.95% re-encoding time is reduced with BDBR increased by 1.02%. This observation proves the importance of the auxiliary classifiers, which can guide the learning process of the training parameters.

B. PERFORMANCE EVALUATION OF CNN-BRT

Table 8 presents the detailed experimental results for each sequence with each ΔQP by using the proposed CNN-BRT. It is observed that as the value of ΔQP decreases from 6 to 2,

TABLE 8. Performance of the proposed CNN-BRT for sequences in YUV 4:4:4 form at under different QPs.

Sequences	ΔQP=2		ΔQP=4		ΔQP=6		Average (All QPs)	
	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)
ChineseEditing	0.41	-55.60	0.62	-55.23	1.15	-55.09	0.73	-55.31
Console	0.65	-56.69	0.94	-56.81	1.18	-56.61	0.92	-56.70
Desktop	0.58	-62.72	0.80	-62.15	0.88	-61.66	0.75	-62.18
FlyingGraphics	0.59	-46.88	1.10	-46.28	1.85	-45.85	1.18	-46.34
Map	1.29	-40.98	2.12	-41.20	2.98	-41.15	2.13	-41.11
Programming	0.78	-52.14	1.27	-52.10	1.72	-51.86	1.26	-52.03
SlideShow	0.80	-58.38	1.65	-58.74	2.68	-59.25	1.71	-58.79
WebBrowsing	0.63	-59.55	0.56	-59.11	0.73	-58.02	0.64	-58.89
BasketballScreen	0.58	-50.21	0.96	-49.83	1.44	-49.38	0.99	-49.81
MissionControlClip2	0.55	-49.03	0.85	-48.45	1.48	-48.26	0.96	-48.58
MissionControlClip3	0.61	-52.17	0.95	-51.09	1.63	-50.73	1.06	-51.33
Robot	0.54	-56.65	1.20	-55.51	1.73	-53.88	1.16	-55.35
EBURainFruits	0.38	-65.09	0.34	-62.76	0.61	-60.14	0.44	-62.66
Kimono1	0.17	-72.74	0.21	-68.35	0.26	-65.63	0.21	-68.91
Average (All sequences)	0.61	-55.63	0.97	-54.83	1.45	-54.11	1.01	-54.86

better performance is provided because the correlation between the final optimal mode and the decoded optimal mode increases. Specifically, for ΔQP with the values of 2, 4, and 6, re-encoding time of 55.63%, 54.86%, and 54.11% is reduced with BDBR increased by 0.61%, 0.97% and 1.45% on average, respectively. By average the results of different ΔQPs, 54.86% re-encoding time is reduced with BDBR increased by 1.01%. Table 9 shows the average results with the three groups of QP_2 by applying CNN-BRT to the training sequences, where 52.60% re-encoding time is reduced with BDBR increased by 0.89%. It is observed that similar results are provided for both training sequences and testing sequences. Therefore, CNN-BRT does not run into the overfitting problem and it is generalizable to the unseen sequences.

We also evaluate the performance of CNN-BRT by investigating the hit rate of the proposed CNN-BRT, which is calculated as the percentage of areas coded by the same mode as the original CBFT, and the results are shown in Table 10. It is observed that the hit rate of CNN-BRT is above 90% for all sequences under different values of QP_1 and ΔQP. Under a certain value of QP_1, the hit rate increases as the value of ΔQP decreases. For example, hit rates of 97.31%, 96.55%, and 96.18% are provided on average for ΔQP of 2, 4, and 6 under QP_1 of 22, respectively. It is due to the same reason that the correlation between the final optimal mode and the decoded optimal mode map increases as the value of ΔQP decreases.

Then, the performance of CNN-BRT is compared with the existing methods to evaluate its efficiency. These methods include the fast encoders in [24]–[28] and the fast transcoder in [29]. It should be noted that the fast encoders in [24]–[28] are incorporated into CBFT by replacing the original SCC encoder for re-encoding time reduction. Zhang *et al.* [29] is the only existing fast bitrate transcoding method in the literature. We re-implemented the fast encoders in [24]–[28] and the proposed CNN-BRT in the same reference software as the fast transcoder in Zhang *et al.* [29], and all experimental settings are kept the same as in Zhang *et al.* [29] for a fair comparison. The indirect comparison is given between the proposed CNN-BRT and Zhang *et al.* [29] by using the results from its original publication. The results for sequences in YUV 4:4:4 format are provided in Table 11, and the largest decrease in ΔTime and the smallest increase in BDBR are highlighted by boldface for each sequence. It is observed that the proposed CNN-BRT outperforms the existing methods by a large margin. Since the fast prediction rules in [24]–[28] do not utilize the information from the decoder side, and they rely on the assumption that the SCBs are noise-free, they show poor performance compared with the proposed CNN-BRT. Specifically, CBFT+Zhang [24], CBFT+Duanmu [25], CBFT+Lei [26], CBFT+Yang [27], and CBFT+Kuang [28] provides 31.83%, 30.15%, 31.61%, 22.00% and 41.62% with BDBR increased by 1.65%, 6.08%, 2.34%, 2.38%, and 4.59% on average, respectively. Although the fast bitrate transcoder of Zhang *et al.* [29] utilizes both the decoder side information and the encoder side information, it shows inferior performance than the proposed CNN-BRT because it only derives limited hand-crafted rules to speed up the re-encoding process. For their selected sequences, Zhang *et al.* [29] provides a 44.49% re-encoding time reduction

TABLE 9. Performance of the proposed CNN-BRT for training sequences.

Training Sequences	BDBR (%)	ΔTime (%)
ChineseDocumentEditing	0.66	-46.87
ClearTypeSpreadsheet	0.03	-57.51
CadWaveform	1.51	-59.77
PcbLayout	0.72	-49.80
PptDocXls	0.75	-53.00
RealTimeData	0.99	-49.82
VideoConferencingDocSharing	1.04	-59.39
WordEditing	0.77	-55.49
BigBuck	1.30	-49.93
KristenAndSaraScreen	1.37	-50.24
KimonoError2	0.92	-34.73
MissionControlClip1	1.05	-46.68
Viking	0.83	-61.69
EBULupoCandlelight	0.46	-61.46
Average	0.89	-52.60

TABLE 10. Hit rate of the proposed CNN-BRT.

Sequences	QP_1=22 (%)			QP_1=27 (%)			QP_1=32 (%)			QP_1=37 (%)		
	ΔQP=2	ΔQP=4	ΔQP=6	ΔQP=2	ΔQP=4	ΔQP=6	ΔQP=2	ΔQP=4	ΔQP=6	ΔQP=2	ΔQP=4	ΔQP=6
ChineseEditing	96.98	96.43	95.61	96.62	96.09	95.54	96.33	95.02	94.42	95.92	94.47	91.63
Console	97.42	97.35	96.71	97.02	96.17	95.28	95.14	91.95	92.62	94.34	94.02	92.46
Desktop	98.37	98.01	97.78	98.06	97.56	97.10	97.45	96.94	96.60	97.45	96.14	94.87
FlyingGraphics	97.49	96.86	96.16	97.55	96.69	96.01	97.41	96.57	95.61	97.02	95.68	94.24
Map	94.59	92.70	90.79	94.39	93.30	92.43	95.79	94.52	94.01	95.74	95.34	92.91
Programming	96.57	95.84	95.30	96.72	96.21	95.83	96.69	96.25	95.05	96.32	95.58	94.43
SlideShow	98.08	96.75	96.71	97.79	97.35	97.02	98.45	97.75	97.48	98.39	98.26	97.08
WebBrowsing	97.64	97.27	98.32	96.94	97.34	95.96	98.20	97.59	97.35	96.83	97.97	96.77
BasketballScreen	98.44	98.05	97.69	97.25	96.79	96.73	97.39	96.89	96.54	97.34	96.36	94.79
MissionControlClip2	97.09	96.62	96.72	97.41	97.44	97.52	97.73	97.14	96.34	97.45	96.99	95.91
MissionControlClip3	97.23	96.66	96.67	97.44	96.93	96.28	96.89	96.22	95.89	97.03	96.62	95.87
Robot	97.81	96.12	95.71	97.66	96.38	96.06	97.96	96.46	95.81	98.05	96.78	95.32
EBURainFruits	98.98	97.22	96.34	98.66	97.41	97.16	98.74	98.59	97.15	98.75	97.33	96.52
Kimono1	99.04	95.61	95.34	99.16	97.34	97.06	99.20	97.76	97.17	98.95	97.69	98.02
Average	97.31	96.55	96.18	97.07	96.52	95.98	97.12	96.11	95.64	96.82	96.18	94.69

TABLE 11. Performance evaluation of different methods compared with HM-16.2+SCM-3.0 under CTC for sequences in YUV 4:4:4 format.

Sequences	CBFT+Zhang [24]		CBFT+ Duanmu [25]		CBFT+Lei [26]		CBFT+ Yang [27]		CBFT+Kuang [28]		Zhang [29]		Proposed CNN-BRT	
	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)
ChineseEditing	1.09	-49.01	6.52	-22.33	0.92	-17.34	3.95	-14.79	2.98	-47.83			**0.73**	**-55.31**
Console	4.45	-36.42	15.93	-45.09	3.18	-21.71	4.13	-16.34	8.88	-49.71	2.50	-42.33	**0.92**	**-56.70**
Desktop	4.00	-41.80	10.08	-40.70	1.72	-20.03	2.83	-17.12	9.35	-57.97	1.55	-48.61	**0.75**	**-62.18**
FlyingGraphics	**0.90**	-7.13	6.75	-27.65	1.84	-16.60	2.93	-12.40	5.54	-46.22	1.92	-44.70	1.18	**-46.34**
Map	**0.85**	-38.31	3.40	-19.54	0.99	-21.75	1.39	-16.18	2.51	-35.89			2.13	**-41.11**
Programming	1.28	-37.13	5.21	-25.46	1.97	-23.67	3.03	-19.55	4.11	-43.89	2.05	-42.83	**1.26**	**-52.03**
SlideShow	**1.27**	-43.65	8.82	-48.08	5.38	-50.13	2.88	-44.10	7.78	-25.30	1.29	-40.39	1.71	**-58.79**
WebBrowsing	3.33	-48.44	7.96	-35.77	4.16	-25.02	2.43	-17.22	5.68	-52.72	1.16	-48.08	**0.64**	**-58.89**
BasketballScreen	1.20	-37.61	5.19	-22.90	1.40	-24.81	1.78	-18.57	5.98	-41.96			**0.99**	**-49.81**
MissionControlClip2	**0.89**	-33.30	4.50	-28.95	1.35	-31.52	1.57	-24.45	4.41	-36.03			0.96	**-48.58**
MissionControlClip3	**0.57**	-33.49	6.33	-25.41	1.70	-23.94	2.48	-17.38	4.41	-44.96			1.06	**-51.33**
Robot	0.85	-16.67	1.26	-24.88	5.20	-47.58	**0.63**	-25.26	1.95	-38.29			1.16	**-55.35**
EBURainFruits	0.61	-18.60	1.43	-26.81	1.34	-47.17	1.49	-26.12	0.48	-33.00			**0.44**	**-62.66**
Kimono1	1.79	-4.06	1.70	-28.46	1.56	**-71.33**	1.81	-38.46	**0.19**	-28.90			0.21	-68.91
Average ([29])											1.75	-44.49	**1.08**	**-55.82**
Average (ALL)	1.65	-31.83	6.08	-30.15	2.34	-31.61	2.38	-22.00	4.59	-41.62			**1.01**	**-54.86**

with a 1.75% increase in BDBR. Comparatively, the proposed CNN-BRT contains extensive learnable parameters to address the classification task, and it outperforms Zhang *et al.* [29] by providing a 55.82% re-encoding time reduction with BDBR increased by 1.01%.

In the fast re-encoding process, the proposed CNN-BRT gives the probabilities of four classes, i.e, $P(\omega_{Intra})$, $P(\omega_{IBC})$, $P(\omega_{PLT})$, and $P(\omega_{Allskip})$. These probabilities allow the adaptive skipping of Intra, IBC, PLT, or an entire CU by skipping all modes for it. To evaluate the contribution of each individual skipping, Table 12 presents the performance of CNN-BRT by enabling adaptive skipping of an individual mode or an entire CU separately. As analyzed in Table 3, Intra mode takes up the largest percentages of re-encoding time, followed by IBC and PLT modes. Similar to this trend, it is observed in Table 12 that enabling the adaptive skipping of Intra mode provides the largest re-encoding time of 28.04%, followed by the adaptive skipping of IBC and PLT modes, where 10.03% and 2.77% re-encoding time is reduced, respectively. Furthermore, enabling the adaptive skipping of an entire CU provides 15.75% re-encoding time reduction.

Table 13 shows the computational overhead of the proposed CNN-BRT for each sequence, and it is calculated as the percentage of the inference time of the CNN model to the total re-encoding time of the proposed CNN-BRT. It is observed that the computational overhead of CNN-BRT is only 4.24% on average based on a CPU. Since CNN-BRT adopts the non-overlapping convolution and it makes the predictions for all CUs in a CTU in a single test, the computational overhead of the proposed CNN-BRT is negligible.

Furthermore, we investigate the performance of the CNN-BRT applied to sequences in YUV 4:2:0 and RGB 4:4:4 formats, and the results are shown in Table 14. For sequences in YUV 4:2:0 format, the luminance component of a CTU is directly input into CNN-BRT. For sequences in RGB 4:4:4 format, color space conversion is performed to

TABLE 12. Performance of the proposed CNN-BRT by enabling the adaptive skipping of intra, IBC, PLT modes, or an entire CU.

Sequences	Skipping of Intra		Skipping of IBC		Skipping of PLT		Skipping of an entire CU	
	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)
ChineseEditing	0.09	-34.84	0.24	-9.01	0.42	-2.14	0.14	-12.24
Console	0.05	-39.37	0.49	-3.62	0.52	-1.65	0.42	-12.36
Desktop	0.01	-45.19	0.33	-5.07	0.30	-2.68	0.10	-6.24
FlyingGraphics	0.17	-31.29	0.32	-3.44	0.61	-2.97	0.14	-1.75
Map	0.85	-21.81	0.24	-8.03	1.07	-2.62	0.14	-4.45
Programming	0.12	-32.18	0.22	-4.92	0.89	-3.34	0.15	-8.46
SlideShow	0.59	-28.96	0.40	-3.21	0.86	-1.74	0.30	-36.07
WebBrowsing	0.09	-38.77	0.24	-7.12	0.28	-1.85	0.10	-6.15
BasketballScreen	0.07	-26.32	0.16	-7.85	0.62	-2.93	0.04	-8.41
MissionControlClip2	0.11	-23.34	0.25	-6.44	0.50	-3.62	0.10	-14.24
MissionControlClip3	0.01	-30.25	0.17	-6.42	0.80	-4.11	0.06	-6.74
Robot	0.44	-14.08	0.39	-24.26	0.25	-4.69	0.10	-15.56
EBURainFruits	0.24	-13.03	0.15	-29.37	0.01	-2.43	0.08	-29.16
Kimono1	0.15	-13.13	0.04	-21.68	0.03	-2.02	0.13	-58.63
Average	0.21	-28.04	0.26	-10.03	0.51	-2.77	0.14	-15.75

TABLE 13. Computational overhead of each sequence.

Sequences	Computational overhead
ChineseEditing	2.73
Console	5.44
Desktop	5.02
FlyingGraphics	3.29
Map	2.51
Programming	4.32
SlideShow	5.23
WebBrowsing	5.24
BasketballScreen	4.13
MissionControlClip2	4.30
MissionControlClip3	3.71
Robot	3.97
EBURainFruits	4.26
Kimono1	5.27
Average	4.24

TABLE 14. Performance of CNN-BRT for sequences in RGB 4:4:4 and YUV 4:2:0 formats.

Sequences	RGB 4:4:4		YUV 4:2:0	
	BDBR (%)	ΔTime (%)	BDBR (%)	ΔTime (%)
ChineseEditing	-54.03	1.10	-24.84	0.54
Console	-54.88	1.66	-30.06	0.98
Desktop	-61.03	1.26	-36.96	0.72
FlyingGraphics	-45.61	0.93	-22.66	0.47
Map	-37.04	1.32	-30.47	0.26
Programming	-50.49	0.85	-36.93	0.39
SlideShow	-62.27	1.40	-52.64	0.63
WebBrowsing	-59.18	0.60	-41.38	0.46
BasketballScreen	-48.06	0.88	-37.46	0.44
MissionControlClip2	-51.63	0.79	-38.57	0.42
MissionControlClip3	-51.22	1.01	-36.58	0.29
Robot	-56.43	0.61	-57.27	0.54
ChinaSpeed			-37.23	0.49
EBURainFruits	-62.84	0.30		
Kimono1	-74.73	0.30		
Average	-54.96	0.93	-37.15	0.51

obtain the luminance component. Although the CNN model is trained by data in YUV 4:4:4 formats, it is also generalizable to sequences in other formats. It is observed that CNN-BRT provides 54.96% and 37.15% re-encoding time reduction with 0.93% and 0.51% increase in BDBR for sequences in RGB 4:4:4 and YUV 4:2:0 formats, respectively. The time reduction is relatively smaller for YUV 4:2:0 sequences because there are many coding techniques disabled, such as cross component prediction and adaptive color transform. Since YUV 4:4:4 is the most widely used format for screen content sequences, and most existing methods in [24]–[29] do not support sequences in other formats, we cannot make the comparison for sequences in YUV 4:2:0 and RGB 4:4:4 formats.

V. CONCLUSION

In this paper, an efficient bitrate transcoder CNN-BRT was proposed by using a convolutional neural network. To reduce the computational complexity in the re-encoding part, CNN-BRT can skip the checking of unnecessary mode candidates. By modeling the mode decision from the decoder side as the decoded optimal mode maps, the decoder side information is fed to CNN-BRT together with the raw samples from the encoder side, and CNN-BRT eliminates unnecessary mode candidates for the entire CTU in a single test. To imitate the optimal mode selection in the original re-encoding part, a loss function considering both the sub-optimal modes and the final optimal modes is derived. Experimental results show that the proposed CNN-BRT outperforms all existing methods by providing 54.86% re-encoding time reduction with only 1.01% increase in BDBR on average for typical screen content sequences under AI configuration.

REFERENCES

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

[2] 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. 2016.

[3] G. J. Sullivan, J.-R. 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.

[4] X. Xu, S. Liu, T.-D. Chuang, and S. Lei, "Block vector prediction for intra block copying in HEVC screen content coding," in *Proc. Data Compress. Conf.*, Snowbird, UT, USA, Apr. 2015, pp. 273–282.

[5] X. Xu, S. Liu, T.-D. Chuang, Y.-W. Huang, S.-M. Lei, K. Rapaka, C. Pang, V. Seregin, Y.-K. Wang, and M. Karczewicz, "Intra block copy in HEVC screen content coding extensions," *IEEE J. Emerg. Sel. Topic Circuits Syst.*, vol. 6, no. 4, pp. 409–419, Dec. 2016.

[6] Z. Ma, W. Wang, M. Xu, and H. Yu, "Advanced screen content coding using color table and index map," *IEEE Trans. Image Process.*, vol. 23, no. 10, pp. 4399–4412, Oct. 2014.

[7] G. Bjontegaard, *Calculation of Average PSNR Differences Between RD-curves*, document VCEG-M33, VCEG, Austin, TX, USA, Mar. 2001.

[8] G. Fernandez-Escribano, H. Kalva, P. Cuenca, L. Orozco-Barbosa, and A. Garrido, "A fast MB mode decision algorithm for MPEG-2 to H.264 P-frame transcoding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 2, pp. 172–185, Feb. 2008.

[9] G. Fernandez-Escribano, H. Kalva, J. Martinez, P. Cuenca, L. Orozco-Barbosa, and A. Garrido, "An MPEG-2 to H.264 video transcoder in the baseline profile," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 5, pp. 763–768, May 2010.

[10] T. Shanableh, E. Peixoto, and E. Izquierdo, "MPEG-2 to HEVC video transcoding with content-based modeling," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 7, pp. 1191–1196, Jul. 2013.

[11] P. Xing, Y. Tian, X. Zhang, Y. Wang, and T. Huang, "A coding unit classification based AVC-to-HEVC transcoding with background modeling for surveillance videos," in *Proc. Vis. Commun. Image Process. (VCIP)*, Kuching, Malaysia, Nov. 2013, pp. 1–6.

[12] H. Yuan, C. Guo, J. Liu, X, Wang, and S. Kwong, "Motion-homogeneous-based fast transcoding method from H.264/AVC to HEVC," *IEEE Trans. Multimedia*, vol. 19, no. 7, pp. 1416–1430, Jul. 2017.

[13] F. Duanmu, Z. Ma, W. Wang, M. Xu, and Y. Wang, "A novel screen content fast transcoding framework based on statistical study and machine learning," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Phoenix, AZ, USA, Sep. 2016, pp. 4205–4209.

[14] W. Kuang, Y.-L. Chan, S.-H. Tsang, and W.-C. Siu, "Fast HEVC to SCC transcoding based on decision trees," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, San Diego, CA, USA, Jul. 2018 pp. 1–6.

[15] 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, 2019.

[16] F. Duanmu, Z. Ma, M. Xu, and Y. Wang, "An HEVC-compliant fast screen content transcoding framework based on mode mapping," *IEEE Trans. Circuits Syst. Video Technol.*, to be published.

[17] J. D. Cock, S. Notebaert, P. Lambert, and R. V. D. Walle, "Architectures for fast transcoding of H.264/AVC to quality-scalable SVC streams," *IEEE Trans. Multimedia*, vol. 11, no. 7, pp. 1209–1224, Nov. 2009.

[18] L. P. Van, J. De Praeter, G. Van Wallendael, S. Van Leuven, J. De Cock, and R. Van de Walle, "Efficient bit rate transcoding for high efficiency video coding," *IEEE Trans. Multimedia*, vol. 18, no. 3, pp. 364–378, Mar. 2016.

[19] T.-K. Lee, C.-H. Fu, Y.-L. Chan, and W.-C. Siu, "A new motion vector composition algorithm for fast-forward video playback in H.264," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Paris, France, Jun./May 2010, pp. 3649–3652.

[20] K.-T. Fung, Y.-L. Chan, and W.-C. Siu, "Low-complexity and high-quality frame-skipping transcoder for continuous presence multipoint video conferencing," *IEEE Trans. Multimedia*, vol. 6, no. 1, pp. 31–46, Feb. 2006.

[21] K.-T. Fung, Y.-L. Chan, and W.-C. Siu, "New architecture for dynamic frame-skipping transcoder," *IEEE Trans. Image Process.*, vol. 11, no. 8, pp. 886–900, Aug. 2002.

[22] H. Shu and L.-P. Chau, "The realization of arbitrary downsizing video transcoding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 4, pp. 540–546, Apr. 2006.

[23] Y. L. Chan, H. K. Cheung, and W. C. Siu, "Compressed-domain techniques for error-resilient video transcoding using RPS," *IEEE Trans. Image Process.*, vol. 18, no. 2, pp. 357–370, Feb. 2009.

[24] H. Zhang, Q. Zhou, N.-N. Shi, F. Yang, X. Feng, and Z. Ma, "Fast intra mode decision and block matching for HEVC screen content compression," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Shanghai, China, Mar. 2016, pp. 1377–1381.

[25] F. Duanmu, Z. Ma, and Y. Wang, "Fast mode and partition decision using machine learning for intra-frame coding in HEVC screen content coding extension," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 6, no. 4, pp. 517–531, Dec. 2016.

[26] J. Lei, D. Li, Z, Pan, Z. Sun, S. Kwong, and C. Hou, "Fast intra prediction based on content property analysis for low complexity HEVC-based screen content coding," *IEEE Trans. Broadcast.*, vol. 63, no. 1, pp. 48–58, Mar. 2017.

[27] H. Yang, L. Shen, and P. An, "An efficient intra coding algorithm based on statistical learning for screen content coding," in *Proc. IEEE Int. Conf. Image Process.*, Beijing, China, Sep. 2017, pp. 2468–2472.

[28] 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.*, to be published.

[29] L. Zhang, M. Li, and H. Zhang, "Fast intra bit rate transcoding for HEVC screen content coding," *IET Image Process.*, vol. 12, no. 5, pp. 738–744, Apr. 2018.

[30] H.-P. Yu, R. Cohen, K. Rapaka, and J.-Z. Xu, *Common Test Conditions for Screen Content Coding*, document JCTVC-X1015-r1, 24th JCT-VC Meeting, Geneva, Switzerland, May 2016.

[31] HM-16.2+SCM-3.0. *HEVC Test Model Version 16.2 Screen Content Model Version 3.0*. [Online]. Available: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.2+SCM-3.0/

[32] R. Cohen, *AHG8: 4:4:4 Game Content Sequences for HEVC Range Extensions Development*, document JCTVC-N0294, 14th JCT-VC Meeting, Vienna, Austria, Aug. 2013.

[33] A. M. Tourapis, D. Singer, and K. Kolarov, *New Test Sequences for Screen Content Coding*, document JCTVC-O0222, 15th JCT-VC Meeting, Geneva, Switzerland, Nov. 2013.

[34] H.-P. Yu, W. Wang, X. Wang, J. Ye, and Z. Ma, *AHG8: New 4:4:4 Test Sequences With Screen Content*, document JCTVC-O0256, 15th JCT-VC Meeting, Geneva, Switzerland, Nov. 2013.

[35] J. Guo, L. Zhao, and T. Lin, *Response to B1002 Call for Test Materials: Five Test Sequences for Screen Content Video Coding*, document JVET-C0044, 3rd JVET Meeting, Geneva, Switzerland, May 2016.

[36] K. Sharman and K. Suehring, *Common Test Conditions*, document JCTVC-X1100, 24th JCT-VC meeting, Geneva, Switzerland, May 2016.

[37] W. Ding, Y. Shi, and B. Yin, *YUV444 Test Sequences for Screen Content*, document JCTVC-M0431, 13th JCT-VC Meeting, Incheon, South Korea, Apr. 2013.

[38] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. ACM Int. Conf. Multimedia*, Orlando, FL, USA, Nov. 2014, pp. 675–678.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, Santiago, Chile, Dec. 2015, pp. 1026–1034.

[40] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, San Diego, CA, USA, May 2015, pp. 1–13.

[41] *Efficient Bitrate Transcoding for Screen Content Coding Based on Convolutional Neural Network*. Accessed: Aug. 7, 2019. [Online]. Available: http://www.eie.polyu.edu.hk/~ylchan/research/CNN-BRT/

WEI KUANG (S'17) received the B.S. degree from the School of Electronic and Optical Engineering, Nanjing University of Science and Technology, Nanjing, China, in 2015. He is currently pursuing the Ph.D. degree with the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University. His research interests include machine learning and deep learning in video coding and video transcoding. He serves as a Reviewer of international journals, including the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY and *KSII Transactions on Internet and Information Systems*.

YUI-LAM CHAN (S'94–A'97–M'00) received the B.Eng. (Hons.) and Ph.D. degrees from The Hong Kong Polytechnic University, Hong Kong, in 1993 and 1997, respectively.

He joined The Hong Kong Polytechnic University in 1997, where he is currently an Associate Professor with the Department of Electronic and Information Engineering. He is actively involved in professional activities. He has authored over 120 research papers in various international journals and conferences. His research interests include multimedia technologies, signal processing, image and video compression, video streaming, video transcoding, video conferencing, digital TV/HDTV, 3DTV/3DV, multiview video coding, machine learning for video coding, and future video coding standards, including screen content coding, light-field video coding, and 360o omnidirectional video coding.

Dr. Chan was the Secretary of the 2010 IEEE International Conference on Image Processing. He was also the Special Sessions Co-Chair and the Publicity Co-Chair of the 2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, and the Technical Program Co-Chair of the 2014 International Conference on Digital Signal Processing. He serves as an Associate Editor for the IEEE TRANSACTIONS ON IMAGE PROCESSING.

SIK-HO TSANG (M'10) received the Ph.D. degree from The Hong Kong Polytechnic University (PolyU), Hong Kong, in 2013.

He was a Postdoctoral Fellow, from 2013 to 2016, and involved numerous industrial projects for video coding and transcoding. He is currently a Research Fellow with PolyU. He has authored numerous international journals, conferences, and patents. His current research fields involve video coding, such as HEVC, VVC, multiview video plus depth coding, screen content coding, and immersive video coding including light field coding and 360-degree video coding. His research interests include machine learning and deep learning.

Dr. Tsang serves as a Reviewer for international journals, including the IEEE TRANSACTIONS ON IMAGE PROCESSING, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, and the *Elsevier Journal of Signal Processing: Image Communication*.