# DeepSCC: Deep Learning Based Fast Prediction Network for Screen Content Coding

Wei Kuang, *Student Member*, *IEEE*, Yui-Lam Chan, *Member*, *IEEE*, Sik-Ho Tsang, *Member*, *IEEE*, and Wan-Chi Siu, *Life Fellow*, *IEEE*

***Abstract*—Screen Content Coding is an extension of High Efficiency Video Coding (HEVC), and it is developed to improve the coding efficiency of screen content videos by adopting two new coding modes, Intra Block Copy (IBC) and Palette (PLT). However, the flexible quadtree-based coding tree unit (CTU) partitioning structure and various mode candidates make the fast algorithms of SCC extremely challenging. To efficiently reduce the computational complexity of SCC, we propose a deep learning based fast prediction network DeepSCC, which contains two parts, DeepSCC-I and DeepSCC-II. Before fed to DeepSCC, incoming CUs are divided into two categories: dynamic CTUs and stationary CTUs. For dynamic CTUs having different content as their collocated CTUs, DeepSCC-I takes raw sample values as the input to make fast predictions. For stationary CTUs having the same content as their collocated CTUs, DeepSCC-II additionally utilizes the optimal mode maps of the stationary CTU to further reduce the computational complexity. Compared with the HEVC-SCC reference software SCM-8.3, the proposed DeepSCC reduces encoding time by 48.81% on average with a negligible Bjøntegaard delta bitrate increase of 1.18% under all-intra configuration.**

***Index Terms*—Screen Content Coding (SCC), High Efficiency Video Coding (HEVC), fast algorithm, convolutional neural network, deep learning.**

## I. Introduction

SCREEN content video refers to video captured from the display screen of an electronic device, and it has been applied to many screen sharing based applications, such as online education, remote desktop, and web conferencing [1]. Besides the traditional camera-captured natural image blocks (NIBs), screen content videos contain a significant amount of stationary or dynamic computer-generated screen content blocks (SCBs). Compared with NIBs, SCBs exhibit different characteristics, including no sensor noise, large flat areas with a single color, repeated patterns in the same frame and limited colors. Leveraging on these special characteristics of screen content videos, the Joint Collaborative Team on Video Coding (JCT-VC) has developed Screen Content Coding (SCC) extension [2] on top of High Efficiency Video Coding (HEVC) [3], and it outperforms HEVC by achieving over 50% Bjøntegaard delta bitrate (BDBR) [4] reduction for typical screen content videos.

In the development of SCC, two important coding modes, intra block copy (IBC) [5] and palette (PLT) modes [6] are additionally adopted besides the Intra mode of HEVC, and they are particularly effective in addressing coding units (CUs) with repeated patterns and limited colors, respectively. However, the flexible quadtree-based coding tree unit (CTU) partitioning structure and various mode candidates bring a significant computational burden to a SCC encoder.

To simplify the encoding of HEVC, some deep learning based algorithms have been proposed recently. In [7], [8], structures of shallow convolutional neural network (CNN) were proposed to early terminate CU partitions in intra-prediction. In [9], deeper structures of CNN and long- and short-term memory (LSTM) networks were proposed to early terminate CU partitions in both intra-prediction and inter-prediction. However, these algorithms are inefficient in SCC due to the adoption of new modes. First, the new IBC and PLT modes make the CU partition decision in SCC very different from HEVC. The inhomogeneous contents can be coded as large CUs by IBC and PLT modes in SCC, while they are always coded as small CUs in HEVC. Second, the exhaustive mode searching among Intra, and the new IBC and PLT modes brings a significant complexity increase in SCC. The two new modes make all fast HEVC algorithms in [7]–[9] fail in fast mode decision of Intra, IBC and PLT, as they only consider the characteristics of NIBs without the newly introduced IBC and PLT modes. These new modes then make the fast mode decision of SCC much more challenging.

To reduce the computational complexity of a SCC encoder, the existing works include the early heuristic approaches [10]–[14] and the recent machine learning based approaches [15]–[24]. Those works simplify the encoding process of SCC in different aspects, and they are mainly divided into three categories. The first category is to simplify mode decision [11], [12], [16], [17]. In [11], the rate-distortion (RD) cost and CU activity are analyzed to skip unnecessary IBC mode to reduce computational complexity. In [12], a hash value is calculated for each block, and IBC mode only searches repeated patterns among the blocks with the same hash value as the current CU. In [15], fast mode decision is made based on the statistics of learning frames. In [16], [17], decision trees and random forests are used as classification tools to skip unnecessary modes. The second category of fast algorithms is to simplify the CU size decision of SCC [10], [14], [21]. In [10], the heuristic rules based on entropy are proposed to predict the CU partitioning

Manuscript received March 19, 2019; revised June 13, 2019, accepted July 2, 2019. This work was supported by the Hong Kong Research Grants Council under Research Grant PolyU 152069/18E.

The authors are with the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong (e-mail: wei.kuang@connect.polyu.hk; enylchan@polyu.edu.hk; sik-ho.tsang@polyu.edu.hk; enwcsiu@polyu.edu.hk).

decision, and then coding bits after checking all mode candidates are used to eliminate the incorrect decision. In [14], the depth level of the collocated CU is used to predict the depth level of the current CU if they have similar content. All modes are checked in the depth level of the collocated CU, while only PLT mode is checked in other depth levels. However, it is not efficient for sequences with many dynamic CUs, and it needs to disable the fast approach every 10 frames to avoid error propagation. In [21], neural network-based classifiers were trained by utilizing features describing CU statistics and sub-CU homogeneity, and CU sizes are adaptively checked according to the outcomes of the classifiers. However, it induces high RD performance loss because of low classification accuracy. In the third category, fast mode and fast CU size decisions are both investigated to further simplify the encoding proposes of SCC [13], [18]–[20], [22]–[24]. In [13], early CU size decision is made based on RD cost and IBC mode are adaptively skipped by checking the hash value of each block. In [18], [19], learning frames are used to derive the content-adaptive rules for fast mode decision and CU size decision. However, original encoding process needs to be performed in the learning frames, which scarifies the encoding time reduction. In [20], [22]–[24], fast mode decision is preformed based on the assumption that NIBs select Intra mode while SCBs select IBC or PLT mode. Therefore, incoming CUs are classified into SCBs and NIBs by utilizing heuristic rules [20] and traditional machine learning based on hand-crafted features [22]–[24]. In [20], the CU type classification is performed based on CU content property analysis. Smooth NIBs only check Intra mode while the remaining CUs need to check all modes because of low classification accuracy. Then the depth levels of the collocated CU and spatial neighbor CUs as well as coding bits are jointly analyzed to make fast CU size decision. However, it brings high RD performance loss because it does not consider the actual content of the current CU when utilizing information from the collocated CU and spatial neighbor CUs. In [22], decision tree-based classifiers were trained to classify CUs into SCBs and NIBs, so that NIBs only check Intra mode while SCBs check both IBC and PLT modes. Then, another two classifiers were trained to further simplify the encoding of NIBs, which predict the direction of Intra mode from 35 prediction modes and make CU size decision of NIBs, respectively. Besides, several thresholds are derived to early terminate the mode searching process for CUs with small bit cost, but it is helpful for only a small number of CUs. Similarly, decision tree-based classifiers are also utilized in [23] to make CU type classification and CU size decision separately. However, Intra mode is always checked for all CUs with 2N×2N prediction units (PUs) to get the features required by the classifiers, and it brings additional computational overhead for SCBs even though Intra mode is redundant for them. In [24], neural network-based classifiers are trained to classify CUs in NIBs and SCBs. Again, IBC, PLT modes and a subset of Intra mode are checked for SCBs, while only Intra mode is checked for NIBs. Then, various heuristic rules based on the information from spatial and temporal adjacent CUs is proposed to make fast CU partitioning decision.

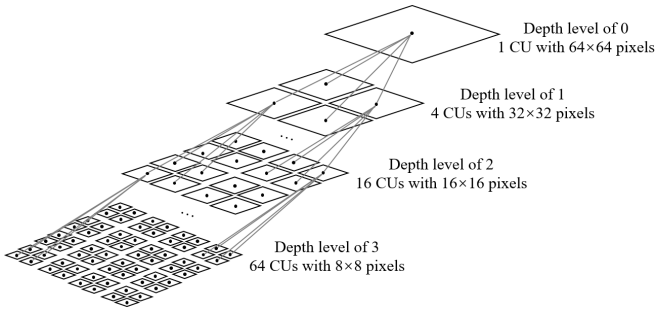


Fig. 1. Hierarchical CTU partitioning structure in SCC.

In this paper, we propose a deep learning based fast prediction network, DeepSCC, to reduce the computational complexity of SCC, and it makes fast predictions for all CUs of a CTU in a single test. Since a screen content sequence usually has many dynamic CTUs as well as stationary CTUs, the proposed DeepSCC is composed of two parts, DeepSCC-I and DeepSCC-II, which simplify the encoding of dynamic CTUs and stationary CTUs, respectively. Specifically, DeepSCC-I only takes the CTU sample values as the input, while DeepSCC-II takes both the CTU sample values and the optimal mode maps of the collocated CTU as the input. Since DeepSCC contains many trainable parameters and learns extensive features, it can make the more accurate mode decision of Intra, IBC, and PLT rather than the simple CU type classification of NIBs and SCBs. Besides, the proposed DeepSCC jointly analyzes the local features of a CU and the global features from other CUs by concatenating convolution layers and deconvolution layers to improve the prediction accuracy.

To the best of our knowledge, we are the first to use deep learning for making fast predictions of SCC. The differences between our contributions and the related schemes can be summarized as: 1) Unlike the existing fast SCC encoding algorithms in [10]–[24] which heavily rely on the limited number of hand-crafted features or heuristic rules, the proposed DeepSCC automatically learns useful features from the raw samples. Since the proposed DeepSCC contains much more trainable parameters than the traditional machine learning based approaches, it learns extensive features and avoids the risk that humans may ignore some important features during feature extraction. 2) Unlike the existing algorithms in [11], [12], [16], [17] which only design the mode decision model, and in [10], [14], [21] which only design the CU partition model, the proposed DeepSCC consider the whole fast encoding process of SCC. Although algorithms in [13], [18]–[20], [22]–[24] also optimize the whole encoding process, they employ numerous models to address mode decision and CU partitioning decision independently for each CU. Besides, a CTU contains 85 CUs, so that they need to test those models for multiple times. As a consequence, multiple models are always built for testing. On the contrary, the proposed DeepSCC makes predictions in the CTU level so that 85 CUs in a CTU get their predictions in a single test. It integrates mode decision and CU partitioning decision in the same model by treating the case of skipping all modes in a CU as a special class of mode decision. Therefore, the multiple model tests in a CTU can be avoided, and it helps to reduce the computational overhead of the proposed DeepSCC. 3) Unlike the algorithms in [14], [20], [24] which directly utilize the optimal mode and the CU depth level from the collocated

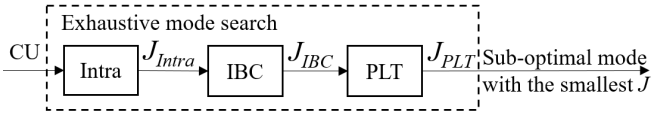


Fig. 2. Exhaustive mode search in a CU.

CU without analyzing the actual CU content, the proposed DeepSCC jointly analyzes the optimal mode maps of the collocated CTU and the content of the current CTU to avoid error propagation. 4) The proposed DeepSCC contains many trainable parameters and learns extensive features, so that it directly performs the mode decision for Intra, IBC, and PLT rather than the simple CU type classification in [20], [22]–[24]. As a result, the decision for IBC and PLT modes can be different, and many SCBs only check one mode from IBC and PLT to further reduce the computational complexity.

The rest of this paper is organized as follows. Section II presents the review and analysis of intra prediction in SCC. Section III presents the proposed fast network DeepSCC. The experimental results are presented in Section IV to verify the performance of the proposed DeepSCC. Finally, Section V concludes the paper.

## II. Review and Analysis of Intra Prediction in SCC

### A. Review on Intra Prediction in SCC

A CTU is a basic processing unit in SCC. To find the optimal CTU coding structure, a CTU is recursively partitioned into CUs in four different depth levels, i.e., depth level $d \in \{0,1,2,3\}$. As shown in Fig. 1, a CTU of 64×64 pixels is partitioned into four CUs of 32×32 pixels, and then each CU of 32×32 pixels is further partitioned into four smaller CUs, until CUs of 8×8 pixels are reached. Therefore, a CTU contains 85 CU partitions (1 + 4 + 16 + 64). In each CU, an exhaustive mode search is performed to find its sub-optimal mode, as shown in Fig. 2. Besides the Intra mode in HEVC that is used to encode the traditional NIBs, SCC additionally adopts two new modes, IBC and PLT, to improve the coding efficiency of SCBs. IBC mode is developed based on the observation that there are many repeated patterns for SCBs in the same frame. When encoding the current CU, IBC searches in the reconstructed region of the current frame to find the best-matched block for it, and the location of the best-matched block is denoted by a block vector. PLT mode is developed based on the observation that a SCB usually contains the limited number of distinct colors. PLT predicts a palette table based on the previously coded CUs, which contains several representative sample values. Then, an index map is sent to the decoder to denote the position of each representative sample value in a CU. In the exhaustive mode search, a Lagrange RD cost $J_x$ is calculated for a mode $x$

$$J_x = D_x + \lambda \times R_x \quad (1)$$

where $x \in \{\text{Intra, IBC, PLT}\}$, $\lambda$ is a Lagrange multiplier, $D_x$ and $R_x$ are the distortion and bit cost of the CU coded with a mode $x$. The sub-optimal mode for a CU is selected as the one with the smallest value of $J_x$. After calculating the RD cost $J_x$, the optimal CTU coding structure is selected as the one with the smallest value of the total RD cost. Then the corresponding sub-optimal modes of those CUs become their optimal modes, and they are involved in the final encoding bitstream.

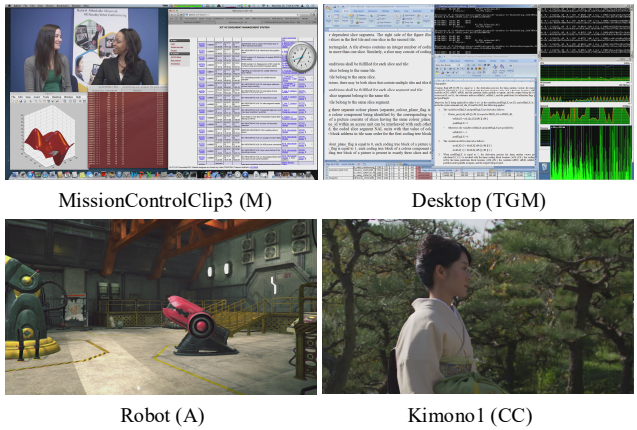


Fig. 3. Examples of testing sequences in four categories.

As shown in Fig. 1, a CTU contains 85 CU partitions, and each CU needs to check three mode candidates, except that CUs only check IBC and Intra modes in the depth level of 0. Therefore, the RD cost $J_x$ is calculated for 254 mode candidates in a CTU (1×2 + 84×3). Although the hierarchical CTU partitioning structure and the exhaustive mode search achieve the best coding performance, it brings significant computational burden to a SCC encoder. Since only parts of those modes are involved in the final encoding bitstream, which are from 1 to 64, precise prediction of the optimal modes in a CTU leads to great encoding time reduction.

### B. Analysis of Intra Prediction in SCC and Motivation of DeepSCC

To analyze the intra prediction in SCC, experiments were performed for sequences in YUV 4:4:4 format based on the HEVC-SCC reference software, Screen Content Model version 8.3 (SCM-8.3) [25]. The testing sequences were selected by the experts in the JCT-VC group, and they were encoded with quantization parameters (QPs) of 22, 27, 32, and 37 using SCM-8.3 under All Intra (AI) configuration defined in the common test conditions (CTC) [26]. Those 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). Fig. 3 shows the examples of testing sequences in four categories. Since sequences in TGM and M show mixed content of NIBs and SCBs, while sequences in A and CC only contain NIBs, we will show the average results for sequences in TGM+M and A+CC in the following sections.

Table I shows the mode distribution of each sequence, which is calculated as the percentages of Intra, IBC, and PLT coded areas in a sequence. Since sequences in A+CC only contain NIBs, it is observed that 97.46% areas of them are encoded by Intra mode on average. Therefore, the CU type classification in [20], [22]–[24] is efficient for NIBs by skipping both IBC and PLT modes. However, it is observed that the mode distributions of sequences in TGM+M are much more complicated, where all modes take up large percentages. Even although "ChineseEditing", "Console", "Desktop" and "FlyingGraphics" only contain SCBs, Intra mode still takes up 10.06%-14.56% in those sequences. Besides, IBC and PLT modes are not evenly distributed. For example, IBC mode takes up 70.93% while PLT mode only takes up 16.72% in "FlyingGraphics". Comparatively, SCBs in "Map" are more likely to select PLT

TABLE I
MODE DISTRIBUTION OF DIFFERENT SEQUENCES

| Categories | Sequences | Intra (%) | IBC (%) | PLT (%) |
|---|---|---|---|---|
| TGM | ChineseEditing | 14.56 | 47.95 | 37.49 |
| | Console | 10.06 | 67.73 | 22.21 |
| | Desktop | 13.38 | 69.07 | 17.55 |
| | FlyingGraphics | 12.35 | 70.93 | 16.72 |
| | Map | 59.84 | 14.88 | 25.28 |
| | Programming | 37.03 | 49.76 | 13.21 |
| | SlideShow | 79.95 | 13.45 | 6.60 |
| | WebBrowsing | 24.13 | 65.93 | 9.94 |
| M | BasketballScreen | 43.35 | 46.59 | 10.06 |
| | MissionControlClip2 | 55.89 | 39.72 | 4.39 |
| | MissionControlClip3 | 43.96 | 48.71 | 7.33 |
| A | Robot | 93.27 | 3.95 | 2.78 |
| CC | EBURainFruits | 99.30 | 0.67 | 0.03 |
| | Kimono1 | 99.80 | 0.19 | 0.01 |
| Average (TGM+M) | | 35.86 | 48.61 | 15.53 |
| Average (A+CC) | | 97.46 | 1.60 | 0.94 |
| Average (ALL) | | 49.06 | 38.54 | 12.40 |

TABLE II
PERCENTAGE OF STATIONARY AREAS IN DIFFERENT SEQUENCES

| Categories | Sequences | Stationary CTU (%) |
|---|---|---|
| TGM | ChineseEditing | 93.41 |
| | Console | 62.72 |
| | Desktop | 78.57 |
| | FlyingGraphics | 2.50 |
| | Map | 79.20 |
| | Programming | 48.11 |
| | SlideShow | 75.41 |
| | WebBrowsing | 96.43 |
| M | BasketballScreen | 86.80 |
| | MissionControlClip2 | 83.82 |
| | MissionControlClip3 | 73.78 |
| A | Robot | 0 |
| CC | EBURainFruits | 0 |
| | Kimono1 | 0 |
| Average (TGM+M) | | 70.98 |
| Average (A+CC) | | 0 |
| Average (ALL) | | 55.77 |

mode than IBC mode, which take up 25.28% and 14.88%, respectively. This observation shows that different SCBs may have different characteristics so that they have different preferences for IBC and PLT modes. The CU type classification in [20], [22]–[24] always treats IBC and PLT modes equally, and either both IBC and PLT modes or all modes are checked for a SCB. In fact, a SCB will only select one mode from IBC or PLT as its optimal mode. Therefore, they are not efficient for SCBs. Although it is difficult to make further classification between IBC and PLT modes by simply selecting a limited number of hand-crafted features, we believe that the recent CNN with extensive trainable parameters shows a promising way to address this problem.

Unlike the traditional camera-captured sequences only containing dynamic CTUs which show different content in adjacent frames, screen content sequences contain many stationary CTUs, i.e., the sum of absolute differences (SAD) between the current CTU and its collocated CTU is 0. Table II shows the percentage of stationary CTUs in different sequences. It is observed that sequences in A+CC only contain dynamic CTUs, while 70.98% CTUs in TGM+M sequences are stationary CTUs. To simplify the encoding of stationary CTUs, an intuitive idea is to directly encode stationary CTUs with the same optimal modes of the collocated CTUs. However, it brings high RD performance loss because whether a CU selects the same mode as its collocated CU is related to its actual content. For example, a SCB with simple texture usually has many repeated patterns within the current frame while a SCB with complex texture has few repeated patterns. If the collocated CU of a simple SCB selects IBC mode, this SCB usually select IBC mode. On the contrary, if the collocated CU of a complex SCB selects IBC mode, this SCB may select PLT mode since its very limited repeated patterns can be disappeared in the current frame. Table III shows BDBR [27] and the change in encoding time, ΔTime, brought by this approach compared with the original SCM-8.3. It should be noted that a negative value of BDBR or ΔTime denotes decrement in percentage as compared with SCM-8.3. It is observed that for sequences in TGM+M that contain many stationary CTUs, this approach provides 44.37% encoding time reduction, but it brings 6.32% increase in BDBR. Although the algorithms in [14], [20], [24] utilize some heuristic rules to reduce the RD performance loss brought by this approach, such as disabling the fast approach every 10 frames to avoid error propagation [14], and jointly analyzing the coding information from the collocated CU and spatial neighbor CUs [20], [24], they still do not achieve a good tradeoff between ΔTime and BDBR. To further improve the performance for stationary CTUs, it is desired that the optimal mode of the collocated CTU and the actual CTU content are jointly analyzed.

TABLE III
PERFORMANCE OF ENCODING STATIONARY CTUS WITH THE SAME OPTIMAL MODES OF THE COLLOCATED CTUS

| Categories | Sequences | BDBR (%) | ΔTime (%) |
|---|---|---|---|
| TGM | ChineseEditing | 3.44 | -51.81 |
| | Console | 4.20 | -47.83 |
| | Desktop | 6.16 | -57.22 |
| | FlyingGraphics | 0.13 | -1.28 |
| | Map | 4.94 | -49.39 |
| | Programming | 2.52 | -31.84 |
| | SlideShow | 13.01 | -43.26 |
| | WebBrowsing | 8.55 | -65.98 |
| M | BasketballScreen | 7.38 | -47.51 |
| | MissionControlClip2 | 12.40 | -44.42 |
| | MissionControlClip3 | 6.77 | -48.67 |
| A | Robot | 0 | 0 |
| CC | EBURainFruits | 0 | 0 |
| | Kimono1 | 0 | 0 |
| Average (TGM+M) | | 6.32 | -44.37 |
| Average (A+CC) | | 0 | 0 |
| Average (ALL) | | 4.96 | -34.95 |

## III. PROPOSED FAST PREDICTION NETWORK DEEPSCC

Generally, humans are sensitive to the difference between SCBs and NIBs, so that many approaches, such as [20], [22]–[24], have successfully differentiated SCBs from NIBs relying on a limited number of hand-crafted features. However, it is very challenging to make further classification between IBC-coded SCBs and PLT-code SCBs with hand-crafted features because humans are less sensitive to their difference. To overcome the limitation of hand-crafted features, a deep learning based fast prediction network DeepSCC is proposed, which contains two parts, DeepSCC-I and DeepSCC-II. DeepSCC-I is used to make predictions for dynamic CTUs, while DeepSCC-II is used to make predictions for stationary CTUs. It is noted that DeepSCC is disabled for the blocks located at the frame boundary and smaller than 64×64 pixels.

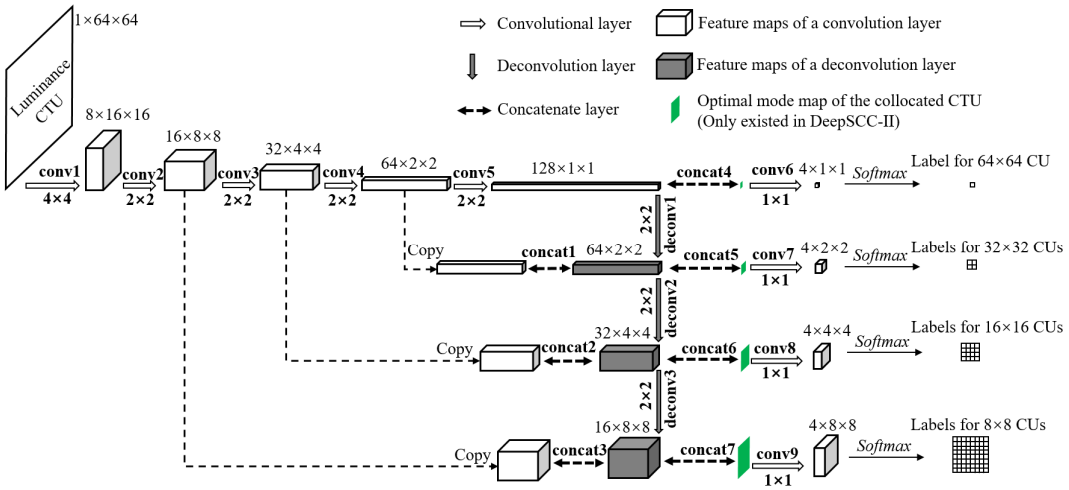


Fig. 4. Structure of DeepSCC. The optimal mode maps of the collocated CTU only exist in DeepSCC-II, which is denoted by green blocks.

Since the proposed DeepSCC contains many trainable parameters and learns extensive features, it can make the more accurate mode decision of Intra, IBC, and PLT rather than the simple CU type classification of NIBs and SCBs. The previous fast prediction approaches of SCC always make predictions in the CU level, which means the derived model is tested for multiple times to make fast prediction for a single CTU. The drawback of this strategy is that it scarifies the encoding time reduction due to the multiple tests of the derived models. To reduce the computation overhead, the proposed DeepSCC directly outputs 85 labels for 85 CUs of a CTU in a single test. Since a CU can either skip all modes or select one mode from Intra, IBC, and PLT, each predicted label contains the probabilities of four classes, i.e., *P*(*Allskip*), *P*(*Intra*), *P*(*IBC*), and *P*(*PLT*), in accordance with the probabilities for skipping all modes, and checking Intra, IBC, PLT modes, respectively. Fig. 4 illustrates the structure of the proposed DeepSCC, where the kernel sizes and feature map dimensions are also presented. The only difference between DeepSCC-I and DeepSCC-II is that the optimal mode maps of the collocated CTU are concatenated to the extracted feature maps before going through the convolution layers conv6–conv9 in DeepSCC-II, which is denoted by green color. The details of DeepSCC are given in the following sub-sections.

### *A. DeepSCC-I for Dynamic CTUs*

As shown in Fig. 4, DeepSCC-I takes the luminance component of a CTU as the input, and it is preprocessed by mean removal before fed to DeepSCC-I. Finally, DeepSCC-I outputs 85 labels for 85 CUs with different sizes, where each label shows the probabilities of selecting different modes. DeepSCC-I is composed of nine convolutional layers (conv1–conv9), three deconvolutional layers (deconv1–deconv3), and three concatenating layers (concat1–concat3). Each convolutional or deconvolutional layer is followed by a rectified linear unit (ReLU) activation function, except for conv6–conv9, where softmax is utilized to generate the output labels. The details of these layers are presented as follows.

*Convolutional layers:* At the beginning, the luminance component of a CTU goes through five convolutional layers, i.e., conv1–conv5, to generate feature maps. As shown in Fig. 4, the kernel size of conv1 is 4×4 and the kernel sizes of conv2–conv5 are 2×2. The strides of conv1–conv5 are set to the width of the kernel sizes for non-overlapping convolutions, in accordance with the non-overlapping CU partitioning structure. By using this arrangement, the receptive field of each node in a feature map is always equal to a CU size, so that the feature maps of conv2–conv5 reflect the local features of CUs from 8×8 to 64×64, respectively. At each downsampling step, we double the channel number of feature maps. After concatenating the feature maps of convolutional layers and deconvolutional layers, conv6–conv9 incorporate those feature maps and generate the last set of feature maps with the kernel size of 1×1 and stride of 1. Each layer of conv6–conv9 outputs four feature maps since each CU contains four classes. Finally, the feature maps of conv6–conv9 are used to output the predicted labels after going through a softmax function.

*Deconvolutional layers*: In contrast to the convolution layer which reduces the size of a feature map, a deconvolutional layer is used to enlarge the size of a feature map. After generating the 128 feature maps of conv5 with the size of 1×1, three deconvolutional layers i.e., deconv1–deconv3, are used to enlarge the feature maps of conv5 using the kernel size of 2×2 and stride of 2. Since the receptive field of each node 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, and they reflect the global features for CUs with size from 32×32 to 8×8, respectively. The global features help to improve the prediction accuracy because there exists spatial content correlation in a CTU. For example, if other CUs are SCBs in a CTU, the current CU is more likely to be a SCB and it would check IBC or PLT mode. On the contrary, if other CUs are NIBs in a CTU, the current CU is more likely to be a NIB and it would check Intra mode. At each feature map enlarging step, we halve the channel number of feature maps. Finally, the global feature maps and the local feature maps have the same dimension for each CU size.

*Concatenating layers*: DeepSCC-I adopts three concatenating layers, i.e., concat1–concat3, to concatenate the global feature maps and local feature maps for CUs with sizes from 32×32 to 8×8, respectively.

As shown in Fig. 4, DeepSCC-I outputs 1, 4, 16, and 64 labels for a CTU, in accordance with the hierarchical CTU

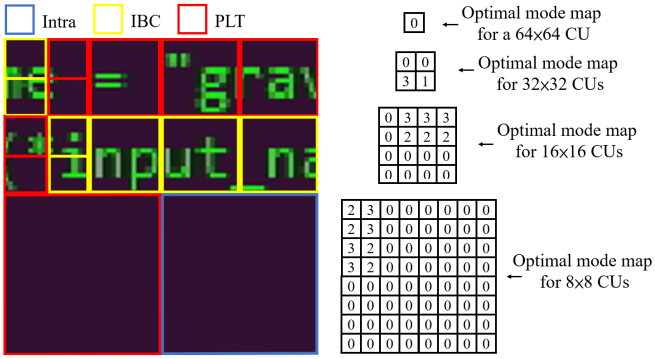


Fig. 5. A collocated CTU and its optimal mode maps.

TABLE IV
TRAINING SEQUENCES FOR DEEPSCC

| Categories | Sequences | Resolution | No. of Frame | Frame Rate (Hz) |
|---|---|---|---|---|
| TGM | ClearTypeSpreadsheet | 1920×1080 | 300 | 30 |
| | PptDocXls | 1920×1080 | 200 | 20 |
| | RealTimeData | 1920×1080 | 600 | 60 |
| | WordEditing | 1920×1080 | 600 | 60 |
| | VideoConferencingDocSharing | 1280×720 | 300 | 30 |
| M | BigBuck | 1920×1080 | 400 | 60 |
| | KristenAndSaraScreen | 1920×1080 | 600 | 60 |
| | MissionControlClip1 | 2560×1440 | 600 | 60 |
| A | Viking | 1280×720 | 300 | 30 |
| CC | EBULupoCandlelight | 1920×1080 | 250 | 50 |
| | Seeking | 1920×1080 | 250 | 50 |
| | ParkScene | 1920×1080 | 240 | 24 |

partitioning structure in Fig. 1, which contains 1 CU of 64×64 pixels, 4 CUs of 32×32 pixels, 16 CUs of 16×16 pixels, and 64 CUs of 8×8 pixels.

### B. DeepSCC-II for Stationary CTUs

As analyzed in Section II.B, directly encoding a stationary CTU with the same optimal modes of the collocated CTU leads to a very high increase in BDBR. To address this problem, the optimal mode maps of the collocated CTU are jointly analyzed with the actual CTU content to reduce the BDBR loss for stationary CTUs. By defining the indices for classes of *Allskip*, *Intra*, *IBC* and *PLT* as 0, 1, 2 and 3, an example of a collocated CTU and its optimal mode maps is shown in Fig. 5. To obtain the optimal mode maps of a collocated CTU, its optimal modes are analyzed in four depth levels. Since the CTU in Fig. 5 is not encoded as a single 64×64 CU, the optimal mode map for the 64×64 CU has the class index of *Allskip*, which means all modes are skipped in the 64×64 CU. Then, there are two 32×32 CUs encoded by PLT mode and Intra mode in the CTU, respectively, so that the class indices of the corresponding positions in the optimal mode map for 32×32 CUs are 3 and 1, which denote *PLT* and *Intra*, respectively. The other two positions in this optimal mode map still contain the index of *Allskip* since they are not encoded as 32×32 CUs. This process is repeated until the four optimal mode maps are all generated. It is noted that the four optimal mode maps of the collocated CTU have the same size as the corresponding feature maps from conv2–covn5 and deconv1–deconv3. To utilize the optimal mode correlation between the current stationary CTU and its collocated CTU, the four optimal mode maps of the collocated CTU are concatenated to the corresponding feature maps of the current CTU by using four concatenate layers concat4–concat7, as shown in Fig. 4. After using conv6–conv9 to incorporate those feature maps and the optimal mode maps, a softmax function is used to output the predicted labels.

### C. Training Strategy for DeepSCC

To avoid the overlapping between the training set and testing set, we selected 12 training sequences from [28]–[32] which are not included in CTC [26] to generate the training samples. These 12 training sequences were carefully selected to cover various video content. Based on the content classification criterion of CTC, we also classify the 12 training sequences into the four categories of TGM, M, A, and CC, and they are shown in Table IV. Then, the 14 sequences in CTC are used as the testing sequences to evaluate the performance of the proposed DeepSCC. A single model of DeepSCC is trained for QPs of 22, 27, 32, and 37 by using mixed training data from the four QPs. For each training sequence, 50 frames were extracted with an equal interval, and they were encoded by the original SCM-8.3 with QPs of 22, 27, 32 and 37 to obtain the ground truth labels. Finally, 750,000 CTUs were generated with their ground truth labels to train DeepSCC-I, while 440,000 CTUs with their ground truth labels and the optimal mode maps of the collocated CTUs were obtained to train DeepSCC-II.

The training process of DeepSCC was implemented in Caffe [33]. A GPU of GeForce GTX 1080 Ti was used to accelerate the training process, and then it was disabled in the testing phase so that only a CPU was used to evaluate the performance of DeepSCC. To make the maximum use of GPU memory, a large batch size of 1024 CTUs was adopted. The loss of an *i-th* training sample in a batch is defined as the sum of cross-entropy over all labels in four depth levels, and it is represented by

$$l_i = f(\omega^0, \hat{\omega}^0) + \sum_{j=0}^{3} f(\omega^{1\_j}, \hat{\omega}^{1\_j}) + \sum_{j=0}^{15} f(\omega^{2\_j}, \hat{\omega}^{2\_j}) + \sum_{j=0}^{63} f(\omega^{3\_j}, \hat{\omega}^{3\_j}) \quad (1)$$

where $\omega^0$ and $\omega^{1\_j}$, $\omega^{2\_j}$, $\omega^{3\_j}$ denote the ground truth classes of the CU in the depth level of 0, and the *j-th* CU in the depth levels of 1, 2, 3, respectively. Similarly, $\hat{\omega}^{0\_j}$, $\hat{\omega}^{1\_j}$, $\hat{\omega}^{2\_j}$, and $\hat{\omega}^{3\_j}$ denote the predicted classes of the corresponding CUs. $f(\cdot,\cdot)$ represents the cross-entropy function between the ground truth class and predicted class, and it is represented as

$$f(\omega, \hat{\omega}) = -\sum_c y(\omega_c = \omega) \log(P(\omega_c = \hat{\omega})) \quad (2)$$

where $c$ denotes the class index. $y(\omega_c = \omega)$ is 1 if $\omega_c$ is the same as the ground truth class $\omega$, otherwise, $y(\omega_c = \omega)$ is 0. $P(\omega_c = \hat{\omega})$ denotes the probability that $\omega_c$ is the same as the predicted class $\hat{\omega}$. By averaging the loss over all training samples in one batch, the loss function $L$ is written as

$$L = \frac{1}{N}\sum_{i=1}^{N} l_i \quad (3)$$

where $N$ is the number of training samples in one batch. All trainable parameters in DeepSCC are initialized by the "msra" filter [34]. Then, Adam optimizer [35] is adopted to update the trainable parameters in DeepSCC with the default values of momentum and momentum2, which are 0.9 and 0.999, respectively. A weight decay of 0.005 is used to alleviate the overfitting problem. Instead of using the conventional learning

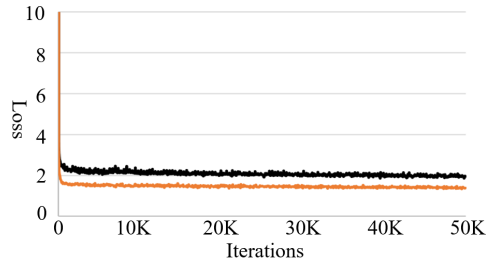


Fig. 6. Training loss of DeepSCC-I and DeepSCC-II alongside iterations.

rate policy of "Step", we adopt the learning rate policy of "Poly" as in [36], and the learning rate in each iteration (*iter*), $lr_{iter}$, is

$$lr_{iter} = lr_{base} \times (1 - \frac{iter}{max_{iter}})^{power} \quad (4)$$

where $lr_{base}$ is the base learning rate of 0.01, $power$ is set to 0.9, and $max_{iter}$ is set to 50,000.

The training losses of DeepSCC-I and DeepSCC-II calculated by (2) are shown in Fig. 6. It is observed that the training processes of DeepSCC-I and DeepSCC-II converge very fast. Besides, the final loss of DeepSCC-II is smaller than DeepSCC-I, because DeepSCC-II additionally utilizes the optimal mode maps of the collocated CTUs. Although DeepSCC-I can reduce encoding time for all CTUs by only taking sample values as the input, we only enable it for dynamic CTUs. For stationary CTUs, DeepSCC-II is enabled instead of DeepSCC-I because it has a smaller loss. The advantage of DeepSCC-II over DeepSCC-I for stationary CTUs is further discussed in Section IV.C.

### D. Content-adaptive Threshold

To make fast prediction for an input CTU, the proposed DeepSCC outputs 85 labels for 85 CUs, and each label contains four probabilities, i.e., $P(\omega)$, $\omega \in \{Allskip, Intra, IBC, PLT\}$. In the testing phase, a threshold $\alpha_x$ is used to decide whether a CU needs to check the mode $x$, $x \in \{Intra, IBC, PLT\}$. If the probability of checking a mode $x$ is smaller than the value of $\alpha_x$, i.e., $P(\omega=x)<\alpha_x$, the mode $x$ is regarded as unnecessary, and the current CU does not check it for encoding time reduction. It should be noted that the selection of the class *Allskip* is not directly decided but depended on the probabilities of checking other classes from $\{Intra, IBC, PLT\}$. If the probabilities of checking all classes from $\{Intra, IBC, PLT\}$ are smaller than $\alpha_x$, the optimal class of the CU becomes *Allskip*, and the mode checking for the CU can be skipped.

In SCC, NIBs and SCBs usually show the concentrated distribution in a frame, and there exists an optimal mode correlation in spatial neighbor CUs. Therefore, $\alpha_x$ is treated as a content-adaptive threshold, and its value is adjusted by utilizing the spatial optimal mode correlation. The mode distribution of the first frame in "Programming" is shown in Fig. 7, and it was encoded by the original SCM-8.3 with QP of 22. It is observed that many CUs select the same modes as their top or left CUs at the same depth levels. Besides, many IBC-coded CUs and PLT-coded CUs are mixed together because IBC and PLT modes are both valid mode candidates for SCBs. Therefore, the value of $\alpha_x$ for a CU is decided by the optimal modes of its top and left neighbor CUs at the same depth level

$$\alpha_x = \alpha_{base} - I_x \times \alpha_{decay} \quad (5)$$

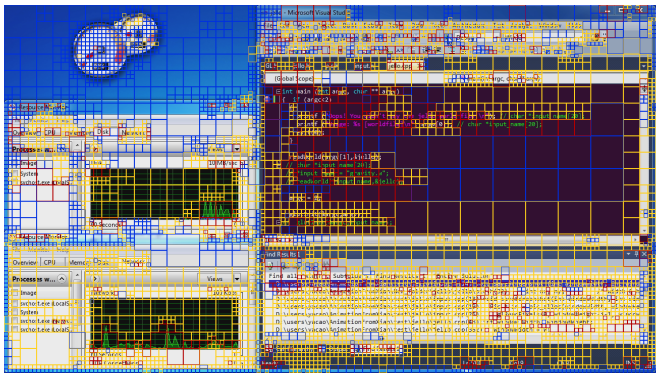

Fig. 7. Optimal mode in the first frame of "Programming". Intra, IBC and PLT coded CUs are noted by blue, yellow and red blocks, respectively.

where $I_x$ is a content-adaptive parameter, $\alpha_{base}$ and $\alpha_{decay}$ are two predefined parameters that control the value of $\alpha_x$. The impact of their values to DeepSCC is discussed in Section IV.A. Since IBC and PLT modes show a mixed distribution, they are grouped together to decide the value of $I_x$. For $x \in \{IBC, PLT\}$, $I_x$ is represented as

$$I_x = \begin{cases} 1, if\ \omega_{left} \in \{IBC, PLT\}\ or\ \omega_{top} \in \{IBC, PLT\} \\ 0, \qquad otherwise \end{cases} \quad (6)$$

For $x \in \{Intra\}$, $I_x$ is represented as

$$I_x = \begin{cases} 1, if\ \omega_{left} \in \{Intra\}\ or\ \omega_{top} \in \{Intra\} \\ 0, \qquad otherwise \end{cases} \quad (7)$$

where $\omega_{left}$ and $\omega_{top}$ are the optimal mode classes of the left and top neighbor CUs, respectively. By using the content-adaptive threshold $\alpha_x$, a CU has a larger chance to be coded by the optimal modes of its left and top CUs.

Since the proposed DeepSCC treats the case of skipping all modes as the class *Allskip* in mode decision, the CU partitioning decision is integrated into DeepSCC. If DeepSCC selects the class *Allskip* for a CU, it means that the current depth level is not optimal and the mode checking of the CU is skipped. Therefore, additional testing of another model specially designed for CU partitioning decision as in [13], [18]–[20], [22]–[24] is not necessary, and it further reduces the testing time. Before a CU in the depth level of 0, 1, or 2 continues the partitioning process shown in Fig. 1, the labels of CUs in the deeper depth levels are analyzed to perform the CU partitioning decision. If an area of a CU always selects the class *Allskip* in all deeper depth levels, the CU cannot be encoded if it continues partitioning. Therefore, we early terminate the CU partition to avoid unnecessary computation.

It should be noted that although IBC and PLT modes are grouped together to derive the content-adaptive thresholds so that $\alpha_{IBC}$ and $\alpha_{PLT}$ are the same, DeepSCC can still differentiate IBC-coded SCBs and PLT-coded SCBs by output independent probabilities of $P(IBC)$ and $P(PLT)$. For example, if $P(IBC)$ is smaller than $\alpha_{IBC}$ while $P(PLT)$ is larger than $\alpha_{PLT}$, only PLT mode will be checked. The details of mode decision distribution will be investigated in Section IV.B.

### E. Memory Overhead of DeepSCC

To make fast prediction, the trained Caffe model needs to be invoked in SCM-8.3. The memory overhead of DeepSCC

TABLE V
VALIDATION SEQUENCES FOR DEEPSCC

| Categories | Sequences | Resolution | No. of Frame | Frame Rate (Hz) |
|---|---|---|---|---|
| TGM | BitstreamAnalyzer | 1920×1080 | 300 | 30 |
| | Doc | 1280×720 | 500 | 10 |
| | Web | 1280×720 | 500 | 10 |
| M | KimonoError2 | 2560×1440 | 500 | 60 |
| CC | BirdsInCage | 1920×1080 | 600 | 60 |
| | DucksAndLegs | 1920×1080 | 300 | 30 |
| | Traffic | 2560×1440 | 60 | 30 |
| | VenueVu | 1920×1080 | 300 | 30 |

comes from two parts, which are the size of the parameters stored in the Caffe model and the size of generated feature maps when running DeepSCC. If a CTU is a dynamic CTU, the Caffe model of DeepSCC-I is invoked, which takes up 348.47KB. To store the generated feature maps, 47.66KB is needed by using the double-precision floating point which requires 8B in C language. Therefore, the memory overhead is 396.13KB for DeepSCC-I. On the other hand, the Caffe model of DeepSCC-II is invoked for stationary CTUs, which takes up 348.80KB, and the associated feature maps require 48.32KB. Therefore, the memory overhead is 397.12KB for DeepSCC-II. Comparatively, a video frame with the resolution of 2560×1440 pixels takes up 108,00KB ($2560 \times 1440 \times 3 \div 1024$). Therefore, the memory overhead percentages of DeepSCC-I and DeepSCC-II over the frame memory are only 3.67% and 3.68%, respectively.

## IV. EXPERIMENTAL RESULTS

To evaluate the performance of the proposed DeepSCC, it has been implemented in SCM-8.3 [25], and the DNN tool of OpenCV 3.4.1 is used to invoke the trained Caffe model in SCM-8.3. The trained Caffe model and the source code of the proposed DeepSCC can be found in our website [37]. The coding efficiency and computational complexity are compared with the original SCM-8.3 under all-intra (AI) configuration defined in CTC [26], and they are measured by BDBR and encoding time increase ΔTime in percentage (%). It should be noted that no GPU but only a CPU is enabled for making fair comparisons. The test platform used for simulations was 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. First, a series of ablation experiments were performed to decide the optimal structure of DeepSCC by using validation sequences [29], [31], [32], [38] in Table V. Second, the performance of DeepSCC is evaluated by comparing it with the existing fast SCC prediction algorithms. Third, the performances of the individual DeepSCC-I and DeepSCC-II are analyzed. In the following sub-sections, we highlight the largest decrease in ΔTime and smallest increase in BDBR by boldface when making comparisons between different methods.

### A. Ablation Study

In this sub-section, various experiments were performed to decide the optimal structure of the proposed DeepSCC by using the validation sequences shown in Table V.

#### 1) Threshold Determination

As aforementioned in Section III.D, a content-adaptive threshold $\alpha_x$ is used to eliminate unnecessary mode candidates in a CU, and its value is controlled by two predefined parameters $\alpha_{base}$ and $\alpha_{decay}$. A fixed value of $\alpha_{decay}$ is applied to analyze the impact of $\alpha_{base}$, and the results are shown in Fig. 8. It is observed that as the value of $\alpha_{base}$ increases, more encoding time is reduced at the cost of a larger increase in BDBR. Besides, when the gap between $\alpha_{base}$ and $\alpha_{decay}$, i.e., $\alpha_{base} - \alpha_{decay}$, is large, BDBR increases quickly. For example, when the gap between $\alpha_{base}$ and $\alpha_{decay}$ increases from 0 to 0.02, the encoding time is further reduced by 5.50% while BDBR is further increased by only 0.37%. When the gap between $\alpha_{base}$ and $\alpha_{decay}$ increases from 0.02 to 0.04, the encoding time is further reduced by 3.96% while BDBR is further increased by 0.86%. Therefore, we limit the gap between $\alpha_{base}$ and $\alpha_{decay}$ to a small value to balance the encoding time reduction and the increase in BDBR, and the results are shown in Table VI. It is observed that DeepSCC is complexity scalable and it provides 46.60%–56.34% encoding time reduction with BDBR increased by 0.48%–1.33%. In the following sub-sections, $\alpha_{base}$ is set to 0.05 and $\alpha_{decay}$ is set to 0.04 for further discussions, where 52.35% encoding time is reduced with 0.83% increase in BDBR.

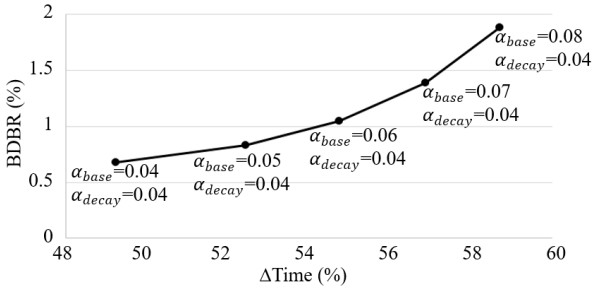


Fig. 8. Performance of DeepSCC with various values of $\alpha_{base}$ and the fixed value of $\alpha_{decay}$.

#### 2) Decoupling Local Features and Global Features

The proposed DeepSCC utilizes convolutional layers and deconvolutional layers to extract the local features and global features in a CTU, respectively. Then, they are concatenated together to predict the mode labels. To evaluate the importance of the proposed structure, two sets of experiments were performed by decoupling local features and global features, i.e., removing concat1–concat3 from DeepSCC. First, only the feature maps of conv2–conv5 are fed to concat4–concat7 so that only local features are utilized to make mode prediction. Second, only the feature maps of conv5 and deconv1–3 are fed to concat4–concat7 so that only global features are utilized to make mode prediction. Let us call them LFDeepSCC and GFDeepSCC, respectively, and their performances are shown in Table VII. It is observed that LFDeepSCC provides 36.90% encoding time reduction with 0.73% increase in BDBR. The original DeepSCC outperforms it by providing a much higher encoding time reduction of 52.35% with a similar increase in BDBR. GFDeepSCC also shows worse performance than DeepSCC by providing 50.66% encoding time reduction with 0.90% increase in BDBR. Therefore, concatenating the local features and global features helps to improve the performance of the proposed DeepSCC.

#### 3) Term Normalization in Loss Function

In (1), the loss function of a training sample is derived as the sum of cross-entropy over all labels in the four depth levels, and the terms for different depth levels are not normalized. For example, the loss function contains only one term in the depth

TABLE VI
PERFORMANCE OF THE PROPOSED DEEPSCC FOR VALIDATION SEQUENCES WITH DIFFERENT VALUES OF $\alpha_{base}$ AND $\alpha_{decay}$

| Sequences | $\alpha_{base}$=0.03 $\alpha_{decay}$=0.02 | | $\alpha_{base}$=0.03 $\alpha_{decay}$=0.01 | | $\alpha_{base}$=0.05 $\alpha_{decay}$=0.04 | | $\alpha_{base}$=0.05 $\alpha_{decay}$=0.03 | | $\alpha_{base}$=0.07 $\alpha_{decay}$=0.06 | | $\alpha_{base}$=0.07 $\alpha_{decay}$=0.05 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BDBR (%) | ΔTime (%) | BDBR (%) | ΔTime (%) | BDBR (%) | ΔTime (%) | BDBR (%) | ΔTime (%) | BDBR (%) | ΔTime (%) | BDBR (%) | ΔTime (%) |
| BitstreamAnalyzer | **0.43** | -40.74 | 0.44 | -41.49 | 0.79 | -45.29 | 0.91 | -45.43 | 1.42 | -48.60 | 1.52 | **-49.35** |
| Doc | 0.38 | -45.05 | **0.34** | -44.13 | 1.35 | -50.47 | 1.52 | -51.56 | 2.06 | -55.60 | 2.13 | **-56.70** |
| Web | **0.86** | -47.17 | 1.03 | -48.32 | 1.43 | -52.71 | 1.57 | -54.04 | 2.71 | -56.66 | 2.99 | **-57.82** |
| KimonoError2 | **0.66** | -31.63 | 0.70 | -29.75 | 0.76 | -37.24 | 0.82 | -37.14 | 0.90 | **-39.98** | 0.93 | **-39.98** |
| BirdsInCage | **0.04** | -49.37 | **0.04** | -49.95 | 0.09 | -59.07 | 0.10 | -59.03 | 0.14 | -63.01 | 0.14 | **-63.10** |
| DucksAndLegs | **0.17** | -62.23 | **0.17** | -62.22 | 0.24 | -64.21 | 0.24 | -64.48 | 0.33 | -65.51 | 0.33 | **-65.76** |
| Traffic | **0.57** | -51.02 | 0.59 | -51.33 | 0.90 | -56.96 | 0.91 | -56.64 | 1.16 | -60.05 | 1.16 | **-60.37** |
| VenueVu | **0.76** | -45.55 | 0.81 | -45.93 | 1.08 | -52.85 | 1.10 | -52.60 | 1.36 | -56.77 | 1.40 | **-57.67** |
| Average (TGM+M) | **0.58** | -41.15 | 0.63 | -40.92 | 1.08 | -46.43 | 1.21 | -47.04 | 1.77 | -50.21 | 1.89 | **-50.96** |
| Average (CC) | **0.39** | -52.04 | 0.40 | -52.36 | 0.58 | -58.27 | 0.59 | -58.19 | 0.75 | -61.34 | 0.76 | **-61.73** |
| Average (ALL) | **0.48** | -46.60 | 0.52 | -46.64 | 0.83 | -52.35 | 0.90 | -52.62 | 1.26 | -55.77 | 1.33 | **-56.34** |

TABLE VII
PERFORMANCE COMPARISON OF DEEPSCC AND OTHER POSSIBLE DESIGNS

| Sequences | LFDeepSCC | | GFDeepSCC | | DeepSCC with term normalization | | DeepSCC with element wise addition layer | | DeepSCC with DenseNet structure | | DeepSCC with "Step" | | Proposed DeepSCC | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BDBR (%) | ΔTime (%) | BDBR (%) | ΔTime (%) | BDBR (%) | ΔTime (%) | BDBR (%) | ΔTime (%) | BDBR (%) | ΔTime (%) | BDBR (%) | ΔTime (%) | BDBR (%) | ΔTime (%) |
| BitstreamAnalyzer | 0.82 | -45.20 | 0.83 | -40.86 | 1.82 | -38.81 | 0.78 | -44.69 | **0.46** | -34.05 | 1.30 | -47.50 | 0.79 | **-45.29** |
| Doc | 1.44 | -49.02 | 1.79 | -50.18 | 0.92 | -47.90 | **1.24** | -49.20 | 1.21 | -43.19 | 1.48 | -50.01 | 1.35 | **-50.47** |
| Web | 1.77 | -51.11 | 1.45 | -51.61 | **1.10** | -49.04 | 1.60 | -52.15 | 1.14 | -44.19 | 1.68 | -51.02 | 1.43 | **-52.71** |
| KimonoError2 | 0.87 | -37.28 | 0.82 | -39.75 | 0.82 | -33.19 | 0.81 | **-37.30** | **0.29** | -38.15 | 0.73 | -36.25 | 0.76 | -37.24 |
| BirdsInCage | **0.03** | -29.38 | 0.11 | -58.02 | 0.07 | -51.96 | 0.09 | **-61.91** | 0.10 | -65.08 | 0.09 | -57.68 | 0.09 | -59.07 |
| DucksAndLegs | **0.03** | -20.02 | 0.19 | -61.32 | 0.22 | -61.15 | 0.22 | **-64.27** | **0.10** | -51.48 | 0.32 | -63.11 | 0.24 | -64.21 |
| Traffic | **0.29** | -35.24 | 0.93 | -52.94 | 0.72 | -40.78 | 0.78 | -54.61 | 0.07 | -46.16 | 0.82 | -53.89 | 0.90 | **-56.96** |
| VenueVu | **0.57** | -27.93 | 1.10 | -50.58 | 0.92 | -44.79 | 1.00 | **-53.16** | 1.07 | -52.46 | 1.12 | -50.01 | 1.08 | -52.85 |
| Average (TGM+M) | 1.23 | -45.65 | 1.22 | -45.60 | 1.17 | -42.23 | 1.11 | -45.84 | **0.78** | -39.90 | 1.30 | -46.20 | 1.08 | **-46.43** |
| Average (CC) | **0.23** | -28.14 | 0.58 | -55.72 | 0.48 | -49.67 | 0.52 | **-58.49** | 0.34 | -53.80 | 0.59 | -56.17 | 0.58 | -58.27 |
| Average (ALL) | **0.73** | -36.90 | 0.90 | -50.66 | 0.82 | -45.95 | 0.82 | -52.16 | 0.56 | -46.85 | 0.94 | -51.18 | 0.83 | **-52.35** |

level of 0 while it contains 64 terms in the depth level of 4. The reason that we do not normalize the loss function to let the terms of different depth levels have equal weight is that the mode classifications in deeper depth levels are more complex. Therefore, the loss function without term normalization will be naturally more focused on the mode classification of small CUs. To prove its advantage, Table VII shows the performance of DeepSCC using loss function with term normalization. It is observed that the original DeepSCC outperforms DeepSCC with term normalization by providing 6.4% more encoding time reduction with almost the same increase in BDBR.

*4) Feature Fusion Function*

To join the convolution features, deconvolution features, and optimal mode maps of the collocated CTU, the concatenating layer is adopted in DeepSCC. It is one of the most widely used feature fusion layers and it can join feature maps with the arbitrary channel number. An alternative way is to use element wise addition layer which can only join two sets of feature maps with the equal channel numbers. Therefore, element wise addition layers can be adopted to join convolution features and deconvolution features since they have equal channel numbers, and then they are concatenated to the optimal mode maps of the collocated CTU. Table VII shows the results of DeepSCC with element wise addition layers. It is observed it almost shows the same results as the original DeepSCC. Therefore, different feature fusion functions have a minor impact on the DeepSCC.

*5) Adoption of DenseNet Structure*

Recently, many advanced CNN structures have been proposed for different tasks. For example, DenseNet [39] alleviates the vanishing-gradient problem, encourages feature reuse and substantially reduces the number of parameters. A set of experiments that adopt the DenseNet structure into DeepSCC were conducted. Each set of kernels in conv1–conv5 and deconv1–deconv3 are replaced by a 4-layer dense block in [39], and the growth rate is 1/4 of the original channel number in each layer so that the output of a 4-layer dense block has the same channel number as the one in the original DeepSCC. The results are shown in Table VII, and it is observed that the adoption of DenseNet does not help to improve the performance of DeepSCC, and it shows almost the same results as the original DeepSCC with $\alpha_{base}$=0.03 and $\alpha_{decay}$=0.01, as in Table VI.

*6) Learning Policy*

In the training process of DeepSCC, the learning rate policy of "Poly" is adopted rather than the conventional "Step". To evaluate the efficiency of this strategy, experiments were done by training DeepSCC with "Step" with the same values of $lr_{base}$ and $max_{iter}$, and the learning rate is multiplied by 0.1 every 10,000 iterations. The performance comparison is shown in Table VII. It is observed that DeepSCC with "Step" achieves 51.18% encoding time reduction with 0.94% increase in BDBR. By replacing "Step" with "Poly", DeepSCC shows a slightly better performance of 1.17% larger encoding time reduction and 0.11% smaller increase in BDBR.

*7) Number of Channels*

The proposed DeepSCC has the advantage of automatically learning useful features by using extensive learnable parameters, which is controlled by the number of channels in each layer. If a small number of channels are employed, DeepSCC may run into the underfitting problem. On the contrary, if a larger number of channels are employed, DeepSCC may run into the overfitting problem. To evaluate the impact of the channel number in DeepSCC, another four sets of experiments were

TABLE VIII
PERFORMANCE COMPARISON OF DEEPSCC WITH DIFFERENT NUMBER OF CHANNELS

| Sequences | NumChannel/4 | | NumChannel/2 | | Original DeepSCC | | NumChannel×2 | | NumChannel×4 | |
|---|---|---|---|---|---|---|---|---|---|---|
| | BDBR (%) | ΔTime (%) | BDBR (%) | ΔTime (%) | BDBR (%) | ΔTime (%) | BDBR (%) | ΔTime (%) | BDBR (%) | ΔTime (%) |
| BitstreamAnalyzer | 1.35 | -38.14 | 1.67 | -38.90 | **0.79** | -45.29 | 1.36 | -47.26 | 1.10 | **-47.93** |
| Doc | 1.36 | -49.86 | 1.32 | -50.52 | 1.35 | **-50.47** | **1.31** | -47.23 | 1.35 | -47.88 |
| Web | **1.32** | -50.67 | 1.50 | -51.76 | 1.43 | **-52.71** | 1.45 | -51.23 | 1.41 | -50.56 |
| KimonoError2 | 0.78 | -35.67 | 0.79 | **-38.04** | 0.76 | -37.24 | **0.68** | -36.72 | 0.91 | -36.82 |
| BirdsInCage | 0.08 | -55.92 | **0.07** | **-66.94** | 0.09 | -59.07 | **0.07** | -55.16 | 0.09 | -50.04 |
| DucksAndLegs | **0.17** | -43.67 | 0.18 | -62.15 | 0.24 | -64.21 | 0.23 | **-64.74** | 0.29 | -63.38 |
| Traffic | **0.53** | -49.04 | 0.79 | -56.14 | 0.90 | **-56.96** | 0.96 | -52.78 | 1.02 | -52.23 |
| VenueVu | 1.10 | -49.85 | 1.12 | **-54.27** | **1.08** | -52.85 | 1.10 | -53.85 | 1.11 | -45.24 |
| Average (TGM+M) | 1.20 | -43.59 | 1.32 | -44.81 | **1.08** | **-46.43** | 1.20 | -45.61 | 1.19 | -45.80 |
| Average (CC) | **0.47** | -49.62 | 0.54 | -59.88 | 0.58 | **-58.27** | 0.59 | -56.63 | 0.63 | -52.72 |
| Average (ALL) | 0.84 | -46.60 | 0.93 | -52.34 | 0.83 | **-52.35** | 0.90 | -51.12 | 0.91 | -49.26 |

TABLE IX
PERFORMANCE OF THE PROPOSED DEEPSCC FOR TRAINING SEQUENCES

| Training Sequences | BDBR (%) | ΔTime (%) |
|---|---|---|
| ClearTypeSpreadsheet | 1.01 | -53.59 |
| PptDocXls | 1.99 | -45.60 |
| RealTimeData | 1.04 | -40.91 |
| WordEditing | 1.40 | -53.54 |
| VideoConferencingDocSharing | 1.86 | -52.61 |
| BigBuck | 1.18 | -42.48 |
| KristenAndSaraScreen | 0.90 | -46.69 |
| MissionControlClip1 | 1.37 | -47.43 |
| Viking | 1.78 | -54.40 |
| EBULupoCandlelight | 0.25 | -53.60 |
| Seeking | 0.30 | -52.65 |
| ParkScene | 0.47 | -58.51 |
| Average (TGM+M) | 1.34 | -47.86 |
| Average (A+CC) | 0.70 | -54.79 |
| Average (ALL) | 1.13 | -50.17 |

performed, i.e., multiplying the channel number of each layer before concat4–concat7 by 1/4, 1/2, 2, 4, and they are denoted as NumChannel/4, NumChannel/2, NumChannel×2, NumChannel×4, respectively. The results are shown in Table VIII. It is observed that the original DeepSCC shows slightly better performance than the networks with the other number of channels. As the channel number increases, the performance of DeepSCC is improved first because of underfitting, and then it is dropped because of overfitting. Therefore, DeepSCC with the proposed channel number achieves a good tradeoff between ΔTime and BDBR.

### *B. Performance of DeepSCC*

Table IX shows the performance of the proposed DeepSCC for training sequences, where 50.17% encoding time is reduced with 1.13% negligible increase in BDBR. Then, to evaluate the performance of the proposed DeepSCC, it is directly compared with four state-of-the-art SCC fast intra prediction algorithms [14], [20], [22], [23]. It is noted that they were implemented in different reference software from ours in their original publications. Zhang *et al.*'s method [14], Lei *et al.*'s method [20], Duanmu *et al.*'s method [22], and Yang *et al.*'s method [23] were simulated using SCM-3.0, SCM-2.0, SCM-4.0, and SCM-5.0, respectively. There are numerous enhancements, speed-up techniques and codes clean-up in SCM-8.3 compared with the older versions. Therefore, we re-implemented them into SCM-8.3 for direct comparison. Besides, we make an indirect comparison of the proposed DeepSCC with Huang *et al.*'s method [24] because we do not have the source code of their approach. However, DeepSCC is implemented in the same reference software as Huang *et al.*'s method [24], SCM-8.3, which makes the indirect comparison to be fair. The results for 14 testing sequences in YUV 4:4:4 format are shown in Table X.

It is observed that DeepSCC outperforms the SCC fast intra prediction algorithms [14], [20], [22]–[24] by providing 48.81% encoding time reduction with only 1.18% increase in BDBR. Compared with the results in Table IX, DeepSCC provides similar performance for both training sequences and testing sequences. This shows that the proposed DeepSCC is generalizable to the unseen sequences. Zhang *et al.*'s method [14] shows similar increase in BDBR to the proposed DeepSCC, but it provides 15.62% smaller encoding time reduction than DeepSCC. Since Zhang *et al.*'s method [14] strongly relies on the CUs having similar content as their collocated CUs, it shows very limited encoding time reduction for sequences with almost only dynamic regions, such as "FlyingGraphics", "Robot", "EBURainFruits", and "Kimono1", where only 4.60%, 12.04%, 16.48%, and 0.46% encoding time is reduced. Comparatively, DeepSCC can efficiently address dynamic CTUs, and it provides 30.76%, 49.73%, 55.94% and 70.68% encoding time reduction for those sequences. Lei *et al.*'s method [20], Duanmu *et al.*'s method [22] and Yang *et al.*'s method [23] all eliminate the mode candidates for a CU by classifying it into a NIB or a SCB, and at most one mode, i.e., Intra mode, is skipped for a SCB. On the contrary, DeepSCC directly performs the mode classification rather than the simple CU type classification, so IBC and PLT modes are no longer always checked together for a SCB. As a result, DeepSCC outperforms the fast algorithms [20], [22], and [23] by providing 15.61%, 21.92% and 13.45% larger encoding time reduction with 1.18%, 0.52%, and 2.32% smaller increase in BDBR, respectively. It should be noted that we do not fine-tune the algorithms in [14], [20], [22], [23] when migrating them from their original SCM versions to SCM-8.3, and the results in Table X may not represent their best performance in SCM-8.3. However, DeepSCC outperforms them by a large margin in SCM-8.3. Furthermore, Table XI further shows the indirect comparisons between DeepSCC and algorithms in [14], [20], [22], [23] under different SCM versions where these algorithms are fine-tuned. It is observed that DeepSCC also achieves the best trade-off between ΔTime and BDBR. Although Zhang *et al.*'s method [14] has 0.14% smaller increase in BDBR than DeepSCC, it provides 10% less encoding time reduction than DeepSCC. Therefore, it is reasonable to conclude that DeepSCC has better performance than the algorithms in [14], [20], [22], [23]. Huang *et al.*'s method [24] adopts a hybrid framework of neural network-based classifiers for CU type classification and various heuristic rules to make CU partitioning decisions. Comparatively, the proposed

TABLE X
PERFORMANCE OF DIFFERENT ALGORITHMS COMPARED WITH SCM-8.3 UNDER CTC FOR SEQUENCES IN YUV 4:4:4 FORMAT

| Sequences | Zhang *et al*. [14] | | Lei *et al*. [20] | | Duanmu *et al*. [22] | | Yang *et al*. [23] | | Huang *et al*. [24] | | Proposed DeepSCC | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | BDBR (%) | ΔTime (%) | BDBR (%) | BDBR (%) | BDBR (%) | ΔTime (%) | BDBR (%) | ΔTime (%) | BDBR (%) | ΔTime (%) | BDBR (%) | ΔTime (%) |
| ChineseEditing | **0.65** | **-49.73** | 0.99 | -18.96 | 1.10 | -17.47 | 4.30 | -34.16 | | | 1.07 | -48.80 |
| Console | 3.36 | -39.35 | 2.87 | -23.40 | 1.87 | -28.12 | 7.38 | **-42.83** | | | 1.06 | -41.85 |
| Desktop | 1.95 | -47.94 | 1.97 | -23.85 | 2.19 | -26.24 | 6.27 | -35.91 | **0.84** | -46.48 | 1.00 | **-53.46** |
| FlyingGraphics | **0.84** | -4.60 | 1.72 | -18.13 | 0.98 | -20.13 | 5.47 | -31.19 | 1.10 | **-43.45** | 0.99 | -30.76 |
| Map | **0.85** | -36.95 | 1.23 | -20.05 | 1.55 | -19.16 | 2.84 | -41.66 | 1.25 | **-42.60** | 1.79 | -36.36 |
| Programming | 1.16 | -40.44 | 2.50 | -22.92 | 1.89 | -22.16 | 4.71 | -27.38 | 2.05 | **-53.66** | **0.87** | -42.74 |
| SlideShow | 1.39 | -44.15 | 2.32 | -55.58 | 2.82 | -52.47 | 3.69 | -34.45 | 1.54 | **-68.38** | 2.78 | -55.36 |
| WebBrowsing | 2.05 | -51.73 | 6.02 | -26.75 | 1.91 | -28.17 | 5.00 | -53.00 | 0.99 | **-55.33** | **0.88** | -54.09 |
| BasketballScreen | 1.06 | -41.84 | 1.46 | -24.83 | 1.25 | -22.43 | 3.00 | -31.54 | **0.87** | -39.83 | 1.27 | **-46.78** |
| MissionControlClip2 | **1.29** | -39.08 | 1.71 | -25.49 | 2.86 | -33.90 | 2.51 | -38.54 | 1.47 | -46.39 | 1.56 | **-51.16** |
| MissionControlClip3 | 1.05 | -39.91 | 1.69 | -33.81 | 2.03 | -24.61 | 2.90 | -34.15 | 1.63 | -39.42 | **1.01** | **-45.96** |
| Robot | 0.92 | -12.04 | 5.21 | -46.91 | 1.18 | -29.36 | **0.59** | -28.19 | 2.52 | -40.31 | 1.81 | **-49.43** |
| EBURainFruits | 0.71 | -16.48 | 1.76 | -48.58 | 0.88 | -26.47 | **0.17** | -25.89 | 0.67 | -50.56 | 0.29 | **-55.94** |
| Kimono1 | 0.15 | -0.46 | 1.52 | **-75.55** | 1.23 | -25.75 | **0.13** | -36.18 | 1.35 | -65.74 | 0.17 | -70.69 |
| Average (TGM+M) | 1.42 | -39.61 | 2.23 | -26.71 | 1.86 | -26.81 | 4.37 | -36.80 | **1.30** | **-48.39** | **1.30** | -46.12 |
| Average (A+CC) | 0.59 | -9.66 | 2.83 | -57.01 | 1.10 | -27.19 | **0.30** | -30.09 | 1.51 | -52.20 | 0.76 | **-58.69** |
| Average ([19]) | | | | | | | | | 1.36 | -49.34 | **1.20** | **-49.39** |
| Average (ALL) | 1.25 | -33.19 | 2.36 | -33.20 | 1.70 | -26.89 | 3.50 | -35.36 | | | **1.18** | **-48.81** |

Indirect comparison is made between [24] and DeepSCC based on SCM-8.3, and [14], [20], [22], [23] were re-implemented into SCM-8.3 for direct comparison.

TABLE XI
INDIRECT COMPARISIONS UNDER DIFFERENT SCM VERSIONS

| Methods | SCM version | BDBR (%) | ΔTime (%) |
|---|---|---|---|
| Zhang *et al*. [14] | SCM-3.0 | 1.04 | -39 |
| Lei *et al*. [20] | SCM-2.0 | 2.01 | -45 |
| Duanmu *et al*. [22] | SCM-4.0 | 1.46 | -40 |
| Yang *et al*. [23] | SCM-5.0 | 2.72 | -49 |
| Proposed DeepSCC | SCM-8.3 | 1.18 | -49 |

DeepSCC integrates the mode decision and CU partitioning decision into the same network by using a one-pass design. Therefore, DeepSCC is easier for implementation than Huang *et al.*'s method [24]. As observed in Table X, Huang *et al.*'s method [24] provides 49.34% encoding time reduction with 1.36% increase in BDBR for their selected sequences. However, the proposed DeepSCC outperforms Huang *et al.*'s method [24] by reducing nearly the same encoding time with 0.16% less increase in BDBR based on the same SCM-8.3. Besides, the training sequences of Huang *et al.*'s method [24] are partly overlapped with its testing sequences, where "WebBrowsing" and "Kimono1" are utilized for both training and testing. On the contrary, the training and testing sequences of the proposed DeepSCC are totally different, which avoids overfitting.

To understand the advantage of the proposed DeepSCC over the fast prediction algorithms [14], [20], [22], [23], the mode decision made by each algorithm is analyzed in detail. Table XII shows the distribution of mode decision in each depth level for a representative sequence "Console". It should be noted that "Console" only contains SCBs. Table I shows "Console" is the sequence with the smallest percentage of Intra mode, and we find that disabling Intra mode for "Console" leads to 31.58% encoding time reduction with only 0.69% increase in BDBR. To reduce redundant mode checking, a good prediction algorithm should let most CUs in "Console" only check IBC or PLT mode or skip all modes in a depth level. Let us call only checking IBC or PLT mode or skip all modes as Goal Mode. It can be seen from Table XII that the proposed DeepSCC obviously outperforms fast prediction algorithms [14], [20], [22], [23]. Since DeepSCC is the only algorithm in Table XII that directly makes mode decision of Intra, IBC, and PLT, it shows a more flexible combination of mode decision than others. For example, IBC mode can be checked alone or checked together with PLT mode or Intra mode. It is observed that many CUs only check IBC or PLT mode or skip all modes, and it has the highest percentages of Goal Mode among the algorithms in Table XII, which are 90.57%, 58.64%, 42.28%, and 42.88% in the depth levels of 0, 1, 2, and 3, respectively. For a CU that has similar content as its collocated CU, Zhang *et al.*'s method [14] only checks PLT mode if the depth levels of the current CU and the collocated CU are not equal. Otherwise, all modes are checked. Therefore, 9.19%–39.37% CUs only check PLT mode in the depth levels of 1–3. However, it still has 53.26%–90.81% CUs need to check all modes in the depth levels of 0–3. Lei *et al.*'s method [20] has no early mode decision for SCBs, and it only utilizes some rules to skip all modes in a CU. Hoverer, it has 38.89%–91.11% CUs need to check all modes in the depth levels of 0–3. Duanmu *et al.*'s method [22] manually disables IBC mode for all CUs in the depth level of 0, and all SCBs need to redundantly check Intra mode in the depth level of 0. Although some thresholds are derived to skip remaining mode candidates for CUs with small bit cost, it is observed that 52.50%–88.13% CUs need to check both IBC and PLT modes due to the simple CU type classification. Yang *et al.*'s method [23] always checks Intra mode for CUs with 2N×2N PUs to obtain the features required by classifiers, so that Intra mode takes a very large percentage even though "Console" only contains SCBs. Besides, it falsely skips IBC and PLT modes for many CUs, which explains the reason for the very high increase in BDBR of 7.38% for "Console".

Table XIII shows the hit rate of *Intra*, *IBC* and *PLT* predicted by the proposed DeepSCC, which is calculated as the percentage of the areas encoded by the same mode as the original SCM-8.3. Besides, the hit rate of *Allskip* is also given by calculating as the percentage of CUs whose all modes are correctly skipped compared with the original SCM-8.3. Since that sequences in A+CC barely select IBC and PLT modes, only the hit rate of Intra mode is shown for them. It is observed that the prediction hit rate of DeepSCC is very high and it varies from 87.23% to 99.45% for different class, different sequences and different QPs. Therefore, the proposed DeepSCC induces a negligible increase in BDBR. In addition, the hit rate is maintained stable for testing sequences with QPs of 22, 27, 32

TABLE XII
COMPARISON OF THE MODE DECISION DISTRIBUTION DECIDED BY DIFFERENT ALGORITHMS OF "CONSOLE"

| Algorithm | Depth level | Console (%) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Intra only | IBC only | PLT only | Intra+IBC | Intra+PLT | IBC+PLT | Intra+IBC+PLT | Skip All Modes | Goal Mode |
| Proposed DeepSCC | 0 | 5.58 | 5.46 | | 3.85 | | | | 85.11 | 90.57 |
| | 1 | 1.08 | 16.62 | 21.42 | 5.93 | 2.39 | 22.37 | 9.59 | 20.60 | 58.64 |
| | 2 | 2.54 | 26.23 | 3.18 | 7.99 | 1.16 | 31.53 | 14.50 | 12.87 | 42.28 |
| | 3 | 1.57 | 24.97 | 0 | 9.31 | 0 | 20.93 | 25.31 | 17.91 | 42.88 |
| Zhang *et al.* [11] | 0 | 0 | 0 | | 53.26 | | | | 46.74 | 46.74 |
| | 1 | 0 | 0 | 39.37 | 0 | 0 | 0 | 60.63 | 0 | 39.37 |
| | 2 | 0 | 0 | 25.75 | 0 | 0 | 0 | 74.25 | 0 | 25.75 |
| | 3 | 0 | 0 | 9.19 | 0 | 0 | 0 | 90.81 | 0 | 9.19 |
| Lei *et al.* [12] | 0 | 0 | 0 | | 38.89 | | | | 61.11 | 61.11 |
| | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 91.11 | 8.89 | 8.89 |
| | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 93.00 | 7.00 | 7 |
| | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 60.17 | 39.83 | 39.83 |
| Duanmu *et al.* [14] | 0 | 100 | 0 | | 0 | | | | 0 | 0 |
| | 1 | 3.40 | 0 | 0 | 0 | 0 | 88.13 | 7.61 | 0.86 | 0.86 |
| | 2 | 3.10 | 0.23 | 0 | 0.04 | 0 | 77.29 | 7.76 | 11.58 | 11.81 |
| | 3 | 0.21 | 0.09 | 0 | 0.02 | 0 | 52.50 | 21.47 | 25.71 | 25.8 |
| Yang *et al.* [15] | 0 | 100 | 0 | | 0 | | | | 0 | 0 |
| | 1 | 72.11 | 0 | 0 | 0 | 0 | 0 | 27.77 | 0.12 | 0.12 |
| | 2 | 33.94 | 0 | 0 | 0 | 0 | 0 | 62.89 | 3.17 | 3.17 |
| | 3 | 21.55 | 0 | 0 | 0 | 0 | 36.26 | 23.20 | 18.99 | 18.99 |

TABLE XIII
HIT RATE OF THE PROPOSED DEEPSCC

| Sequences | QP=22 (%) | | | | QP=27 (%) | | | | QP=32 (%) | | | | QP=37 (%) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *Intra* | *IBC* | *PLT* | *Allskip* | *Intra* | *IBC* | *PLT* | *Allskip* | *Intra* | *IBC* | *PLT* | *Allskip* | *Intra* | *IBC* | *PLT* | *Allskip* |
| ChineseEditing | 98.23 | 98.50 | 94.86 | 95.81 | 97.91 | 98.38 | 94.84 | 95.87 | 97.47 | 98.29 | 94.23 | 95.60 | 95.79 | 97.80 | 93.85 | 95.91 |
| Console | 98.07 | 98.03 | 96.45 | 97.21 | 97.68 | 98.17 | 96.39 | 96.98 | 97.09 | 98.17 | 96.34 | 96.67 | 94.43 | 97.45 | 96.08 | 95.86 |
| Desktop | 98.18 | 98.33 | 96.57 | 96.65 | 98.04 | 98.22 | 96.35 | 96.85 | 97.46 | 98.07 | 96.20 | 96.44 | 96.22 | 97.59 | 95.59 | 95.78 |
| FlyingGraphics | 96.48 | 98.91 | 93.76 | 94.20 | 96.50 | 99.00 | 93.83 | 95.20 | 96.34 | 99.02 | 93.95 | 95.73 | 95.75 | 98.93 | 94.85 | 96.20 |
| Map | 99.45 | 98.26 | 90.84 | 99.07 | 99.38 | 97.63 | 90.95 | 98.73 | 99.13 | 97.41 | 91.04 | 98.75 | 98.62 | 97.70 | 91.59 | 98.90 |
| Programming | 97.04 | 96.37 | 94.13 | 95.21 | 97.92 | 96.56 | 93.71 | 97.65 | 98.09 | 96.20 | 93.77 | 98.16 | 97.79 | 96.31 | 93.39 | 98.48 |
| SlideShow | 98.08 | 92.03 | 92.61 | 98.84 | 98.50 | 95.52 | 92.76 | 99.45 | 99.00 | 96.62 | 94.09 | 99.74 | 99.06 | 95.15 | 94.75 | 99.84 |
| WebBrowsing | 98.76 | 98.67 | 96.70 | 98.16 | 98.74 | 98.03 | 95.76 | 97.79 | 98.93 | 97.80 | 96.32 | 98.41 | 98.37 | 97.69 | 95.29 | 98.70 |
| BasketballScreen | 98.44 | 96.02 | 92.12 | 95.98 | 99.02 | 95.97 | 91.41 | 97.70 | 99.05 | 97.83 | 91.35 | 98.34 | 98.70 | 96.77 | 91.97 | 98.68 |
| MissionControlClip2 | 95.83 | 96.19 | 90.65 | 94.47 | 97.71 | 96.92 | 90.78 | 97.51 | 98.70 | 96.51 | 90.85 | 98.79 | 98.22 | 97.05 | 90.38 | 99.23 |
| MissionControlClip3 | 97.25 | 97.79 | 93.38 | 95.92 | 98.13 | 97.86 | 93.07 | 98.06 | 98.51 | 98.25 | 93.22 | 98.78 | 98.30 | 97.69 | 93.20 | 98.90 |
| Robot | 97.33 | | | 87.23 | 99.19 | | | 93.83 | 99.39 | | | 97.85 | 98.50 | | | 99.40 |
| EBURainFruits | 97.33 | | | 95.58 | 98.12 | | | 98.02 | 98.75 | | | 99.07 | 98.59 | | | 99.61 |
| Kimono1 | 96.66 | | | 97.49 | 98.48 | | | 99.13 | 98.78 | | | 99.53 | 98.77 | | | 99.70 |
| Average (TGM+M) | 97.80 | 97.19 | 93.82 | 96.50 | 98.14 | 97.48 | 93.62 | 97.44 | 98.16 | 97.65 | 93.76 | 97.76 | 97.39 | 97.28 | 93.72 | 97.86 |
| Average (A+CC) | 97.11 | | | 93.43 | 98.60 | | | 96.99 | 98.97 | | | 98.82 | 98.62 | | | 99.57 |
| Average (ALL) | 97.65 | 97.19 | 93.82 | 95.84 | 98.24 | 97.48 | 93.62 | 97.34 | 98.34 | 97.65 | 93.76 | 97.99 | 97.65 | 97.28 | 93.72 | 98.23 |

TABLE XIV
HIT RATE OF DEEPSCC TRAINED BY DATA FROM QP OF 37

| Sequences | QP=22 (%) | | | | QP=37 (%) | | | |
|---|---|---|---|---|---|---|---|---|
| | *Intra* | *IBC* | *PLT* | *Allskip* | *Intra* | *IBC* | *PLT* | *Allskip* |
| ChineseEditing | 96.56 | 97.70 | 94.89 | 95.03 | 97.04 | 97.40 | 94.89 | 96.71 |
| Console | 96.26 | 98.28 | 96.01 | 96.80 | 96.99 | 98.48 | 96.01 | 96.74 |
| Desktop | 96.28 | 98.51 | 94.54 | 97.37 | 96.76 | 98.07 | 94.54 | 96.56 |
| FlyingGraphics | 96.92 | 98.59 | 91.05 | 94.79 | 96.73 | 98.83 | 91.05 | 95.07 |
| Map | 98.10 | 97.44 | 90.01 | 95.59 | 98.77 | 97.57 | 90.01 | 98.45 |
| Programming | 94.18 | 92.54 | 92.50 | 91.89 | 97.40 | 96.79 | 92.50 | 97.75 |
| SlideShow | 92.97 | 89.16 | 91.07 | 96.43 | 98.60 | 95.48 | 91.07 | 99.70 |
| WebBrowsing | 98.00 | 97.15 | 92.53 | 95.91 | 98.55 | 97.35 | 92.53 | 98.78 |
| BasketballScreen | 92.00 | 95.05 | 92.19 | 84.32 | 98.71 | 96.90 | 92.19 | 98.50 |
| MissionControlClip2 | 81.53 | 79.17 | 89.13 | 86.65 | 97.79 | 96.78 | 89.13 | 98.83 |
| MissionControlClip3 | 89.93 | 96.52 | 92.80 | 90.11 | 98.39 | 97.77 | 92.80 | 98.52 |
| Robot | 72.14 | | | 53.39 | 96.80 | | | 96.43 |
| EBURainFruits | 88.00 | | | 85.47 | 98.14 | | | 98.75 |
| Kimono1 | 91.80 | | | 94.90 | 98.70 | | | 99.59 |
| Average (TGM+M) | 93.88 | 94.56 | 92.43 | 93.17 | 97.79 | 97.40 | 92.43 | 97.78 |
| Average (A+CC) | 83.98 | | | 77.92 | 97.88 | | | 98.25 |
| Average (ALL) | 91.76 | 94.56 | 92.43 | 89.90 | 97.81 | 97.40 | 92.43 | 97.88 |

and 37. It is due to the reason that the model of DeepSCC is trained using mixed data generated by QPs of 22, 27, 32 and 37. Furthermore, we also investigate the hit rate of DeepSCC if we train a model by data only from QP of 37 and then apply it to test sequences with QPs of 22 and 37. The results are shown in Table XIV. It is observed that for testing sequences with QP of 37, DeepSCC trained by data from QP of 37 has similar hit rate to the proposed DeepSCC trained by the mixed data from QPs of 22, 27, 32 and 37. However, DeepSCC trained by data from QP of 37 has lower hit rate than the proposed DeepSCC for QP of 22. Therefore, a single model trained by the mixed data from QPs of 22, 27, 32 and 37 can cover a wider QP range of testing sequences while providing similar hit rate to the model trained and tested by data from one QP.

Fig. 9 shows the RD curve and ΔTime for four sequences over different QPs by using DeepSCC, and it is noted that other sequences have similar results. It is observed that the RD curves of DeepSCC are very close to those of the original SCC encoder, which indicates that DeepSCC has negligible influence on video quality. Besides, ΔTime varies little over different QPs for all sequences. Therefore, DeepSCC provides stable performance in both high and low bitrate cases.

Fig. 10 shows the computational overhead of the proposed DeepSCC, which is calculated as the ratio of running DeepSCC to the total encoding time of the proposed fast encoder. Since DeepSCC adopts non-overlapping convolutions and outputs 85

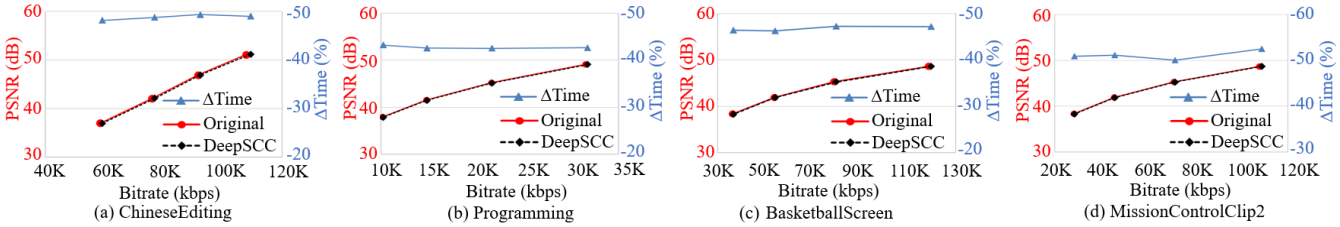


Fig. 9. RD curve and ΔTime of the proposed DeepSCC for "ChineseEditing", "Programming", "BasketballScreen", and "MissionControlClip2".

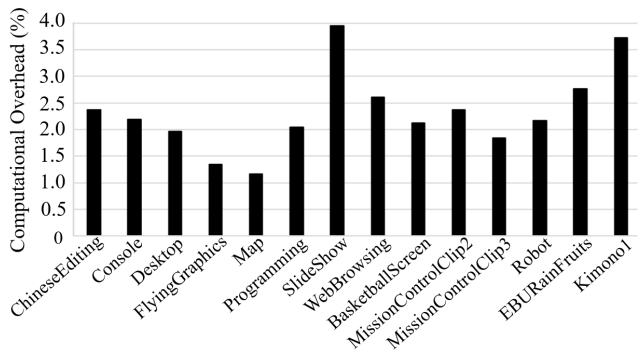


Fig. 10. Computational overhead of the proposed DeepSCC.

TABLE XV
PERFORMANCE OF DEEPSCC FOR SEQUENCES IN RGB 4:4:4 AND YUV 4:2:0

| Sequences | RGB 4:4:4 | | YUV 4:2:0 | |
|---|---|---|---|---|
| | BDBR (%) | ΔTime (%) | BDBR (%) | ΔTime (%) |
| Average (TGM+M) | 1.29 | -42.62 | 1.27 | -42.69 |
| Average (A+CC) | 0.54 | -60.68 | 1.40 | -49.23 |
| Average (ALL) | 1.13 | -46.49 | 1.29 | -43.69 |

TABLE XVI
PERFORMANCE COMPARISONS UNDER LD AND RA CONFIGURATIONS.

| Methods | LD | | RA | |
|---|---|---|---|---|
| | BDBR (%) | ΔTime (%) | BDBR (%) | ΔTime (%) |
| Zhang *et al.* [14] | 0.14 | 1.53 | 0.14 | 1.25 |
| Lei *et al.* [20] | 1.96 | 4.52 | 2.73 | 5.94 |
| Duanmu *et al.* [22] | 3.17 | 7.65 | 2.29 | 10.47 |
| Yang *et al.* [23] | 4.03 | 12.45 | 3.80 | 13.25 |
| Proposed DeepSCC | 0.90 | 11.92 | 0.58 | 10.89 |

labels in a single test, the computational overhead is very low, which varies from 1.17% to 3.94% of the total encoding time for all test sequences. It is noted that the computational overhead is included to calculate the total encoding time of the proposed DeepSCC for all simulations in this paper.

Table XV shows the performance of DeepSCC applied to sequences in RGB 4:4:4 and YUV 4:2:0 formats. While the luminance samples of sequences in YUV 4:2:0 format are directly input to DeepSCC, color space conversion is performed for sequences in RGB 4:4:4 format to get the luminance samples. It should be noted that DeepSCC is only trained by sequences in YUV 4:4:4 format. However, DeepSCC shows good generalization for sequences in YUV 4:2:0 and RGB 4:4:4 formats, where 46.49% and 43.69% encoding time is reduced with only 1.13% and 1.29% increase in BDBR, respectively. Since most existing fast SCC prediction algorithms 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.

Since intra-prediction also exists in low-delay (LD) and random-access (RA) configurations, Table XVI shows the performance comparisons between DeepSCC and algorithms in [14], [20], [22], [23] applied to these configurations. It should be noted that all algorithms are not fine-tuned for LD and RA configurations. It is observed DeepSCC achieves the best trade-off between ΔTime and BDBR, where 11.92% and 10.85% encoding time is reduced with BDBR increased by 0.90% and 0.58% under LD and RA configurations, respectively.

TABLE XVII
PERFORMANCE OF THE INDIVIDUAL DEEPSCC-I AND DEEPSCC-II

| Sequences | DeepSCC-I | | DeepSCC-II | | Proposed Overall DeepSCC | |
|---|---|---|---|---|---|---|
| | BDBR (%) | ΔTime (%) | BDBR (%) | ΔTime (%) | BDBR (%) | ΔTime (%) |
| ChineseEditing | **0.69** | -35.49 | 6.11 | -43.62 | 1.07 | **-48.80** |
| Console | **0.83** | -33.95 | 4.92 | -26.79 | 1.06 | **-41.85** |
| Desktop | **0.64** | -42.75 | 4.08 | -44.78 | 1.00 | **-53.46** |
| FlyingGraphics | **0.98** | -30.48 | 6.58 | -0.78 | 0.99 | **-30.76** |
| Map | **1.62** | -25.32 | 4.10 | -33.52 | 1.79 | **-36.36** |
| Programming | **0.76** | -33.65 | 5.40 | -26.75 | 0.87 | **-42.74** |
| SlideShow | 3.19 | -51.07 | 8.73 | -40.50 | **2.78** | **-55.36** |
| WebBrowsing | **0.49** | -42.58 | 7.13 | -51.63 | 0.88 | **-54.09** |
| BasketballScreen | **0.88** | -37.05 | 1.00 | -38.29 | 1.27 | **-46.78** |
| MissionControlClip2 | **1.36** | -40.62 | 3.44 | -39.25 | 1.56 | **-51.16** |
| MissionControlClip3 | **0.66** | -36.94 | 2.20 | -32.25 | 1.01 | **-45.96** |
| Robot | 1.81 | -49.78 | **1.19** | 0 | 1.81 | **-49.43** |
| EBURainFruits | **0.29** | -55.67 | 0.37 | 0 | **0.29** | **-55.94** |
| Kimono1 | **0.17** | -70.69 | 0.21 | 0 | **0.17** | **-70.69** |
| Average (TGM+M) | **1.10** | -37.27 | 4.88 | -34.38 | 1.30 | **-46.12** |
| Average (A+CC) | 0.76 | -58.71 | **0.59** | 0 | 0.76 | **-58.69** |
| Average (ALL) | **1.03** | -41.86 | 3.96 | -27.01 | 1.18 | **-48.81** |

### C. *Performance of Individual DeepSCC-I and DeepSCC-II*

The proposed overall DeepSCC utilizes DeepSCC-I and DeepSCC-II to make separate predictions for dynamic CTUs and stationary CTUs. To show the advantage of this arrangement, two sets of experiments were performed by only enabling DeepSCC-I and DeepSCC-II for all CTUs, respectively. The results are shown in Table XVII. When applying DeepSCC-II to all CTUs, a very high increase in BDBR of 3.96% is brought since the mode correlation between the current CTU and the collected CTU is not guaranteed. Although some sequences contain very high percentages of stationary CTUs, they still suffer from very high increases of BDBR. For example, "ChineseEditing" contains 93.41% stationary CTUs, and it shows 6.11% increase in BDBR by implementing the individual DeepSCC-II for all CTUs. When applying DeepSCC-I to all CTUs, it provides 41.86% encoding time reduction with 1.03% increase in BDBR. It proves that DeepSCC-I can address both dynamic CTUs and stationary CTUs by only take the luminance samples as the input. However, it shows less encoding time reduction compared with the overall DeepSCC, especially for sequences with many stationary CTUs. For example, the proposed overall DeepSCC shows 13.91% larger encoding time reduction for "ChineseEditing" than the individual DeepSCC-I. Therefore, the proposed overall DeepSCC which integrates DeepSCC-I and DeepSCC-II together helps to improve coding performance.

## V. Conclusion

In this paper, a deep learning based fast prediction network DeepSCC was proposed to reduce the computational complexity of SCC. To avoid the exhaustive mode search in a CTU, DeepSCC outputs 85 labels for 85 CUs of the CTU in a single test. For dynamic CTUs, DeepSCC-I was designed to take the luminance samples of a CTU as the input. For stationary CTUs, DeepSCC-II additionally utilizes the optimal mode maps of the collocated CTUs for further performance improvement. Compared with the traditional fast SCC prediction algorithms heavily relying on the limited number of hand-crafted features or heuristic rules, the proposed DeepSCC automatically learns useful features from the input. With extensive trainable parameters, DeepSCC can make direct mode decision for Intra, IBC, and PLT rather than the simple CU type classification. Experimental results showed that the proposed DeepSCC provides an average computational complexity reduction of 48.81% with a negligible increase in BDBR of 1.18%, and the computational overhead of DeepSCC is less than 4% of the total encoding time. This paper only investigated the fast decision in the granular of the CU level, and it can be treated as a baseline for other CNN approaches in the future. Future works may include fast algorithms making a fast decision in the granular of PU and transform unit (TU) levels for more encoding time reduction. Besides, more advanced CNN structures in SCC could also be a point for our future investigation.

## References


[1] Y. Lu, S. Li, and H. Shen, "Virtualized screen: A third element for cloud-mobile convergence," *IEEE Multimedia*, 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. 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] G. Bjontegaard, "Calculation of average PSNR differences between rd-curves," document VCEG-M33, VCEG, Austin, Texas, USA, Mar. 2001.

[5] X. Xu *et al.*, "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] Z. Liu, X. Yu, Y. Gao, S. Chen, X. Ji, and D. Wang, "CU partition mode decision for HEVC hardwired intra encoder using convolution neural network," *IEEE Trans. Image Process.*, vol. 25, no. 11, 5088–5103, Aug. 2016.

[8] T. Li, M. Xu and X. Deng, "A deep convolutional neural network approach for complexity reduction on intra-mode HEVC," in *Proc. IEEE Int. Conf. Multimedia and Expo*, Hong Kong, China, Jul. 2017, pp. 1255-1260.

[9] M. Xu, T. Li, Z. Wang, X. Deng and Z. Guan, "Reducing complexity of HEVC: A deep learning approach," *IEEE Trans. Image Process.*, vol. 27, no. 10, 5044–5059, Oct. 2018.

[10] M. Zhang, Y. Guo, and H. Bai, "Fast intra partition algorithm for HEVC screen content coding," in *Proc. IEEE Vis. Commun. Image Process.*, Valletta, Malta, Dec. 2014, pp. 390–393.

[11] D.-K. Kwon, and M. Budagavi, "Fast intra block copy (IntraBC) search for HEVC screen content coding," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Melbourne VIC, Australia, Jun. 2014, pp. 9–12.

[12] 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," in *Proc. APSIPA ASC*, Jeju, Korea, Dec. 2016, pp.1–5.

[13] S.-H. Tsang, W. Kuang, Y.-L. Chan 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.

[14] 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.

[15] W. Kuang, S.-H. Tsang, Y.-L. Chan, and W.-C. Siu, "Fast Mode Decision Algorithm for HEVC Screen Content Intra Coding," in *Proc. of Int. Conf. on Image Process.*, pp. 2473-2477, Beijing, China, Sep. 2017.

[16] W. Kuang, Y.-L. Chan, S.-H. Tsang, and W.-C. Siu, "Machine Learning Based Fast Intra Encoding Decision for HEVC Screen Content Coding Via Decision Trees," *IEEE Trans. Circuits Syst. Video Technol.*, early access, 2019.

[17] S.-H. Tsang, Y.-L. Chan, W. Kuang, and W.-C. Siu, "Mode Skipping for HEVC Screen Content Coding via Random Forest," early access, *IEEE Trans. Multimedia*, 2019.

[18] W. Kuang, Y.-L. Chan, S.-H. Tsang, and W.-C. Siu, "Fast Intraprediction for High-Efficiency Video Coding Screen Content Coding by Content Analysis and Dynamic Thresholding," *J. Electron. Imaging*, vol. 27, no. 5, pp. 053029-1-053029-18, Oct. 2018.

[19] 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," early access, *IEEE Trans. Image Process.*, 2019.

[20] 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.

[21] F. Duanmu, Z. Ma, and Y. Wang, "Fast CU partition decision using machine learning for screen content compression," in *Proc. IEEE Int. Conf. Image Process.*, Quebec, QC, Canada, Sep. 2015, pp. 4972–4976.

[22] 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. Topic Circuits Syst.*, vol. 6, no. 4, pp.517–531, Dec. 2016.

[23] 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.

[24] C. Huang, Z. Peng, F. Chen, Q. Jiang, G. Jiang and Q. Hu, "Efficient CU and PU Decision Based on Neural Network and Gray Level Co-Occurrence Matrix for Intra Prediction of Screen Content Coding," *IEEE Access*, vol. 6, pp. 46643- 46655, Aug. 2018.

[25] HM-16.12+SCM-8.3, HEVC test model version 16.12 screen content model version 8.3, [Online], available at: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.12+SCM-8.3/.

[26] H. -P. Yu, R. Cohen, K. Rapaka, and J. -Z Xu, "Common test conditions for screen content coding", 24th JCT-VC meeting, document JCTVC-X1015-r1, Geneva, Switzerland, May. 2016.

[27] G. Bjontegaard, "Calculation of average PSNR differences between rd-curves," document VCEG-M33, VCEG, Austin, Texas, USA, Mar. 2001.

[28] R. Cohen, "AHG8: 4:4:4 game content sequences for HEVC range extensions development", 14th JCT-VC meeting, document JCTVC-N0294, Vienna, Austria, Aug. 2013.

[29] A. M. Tourapis, D. Singer, and K. Kolarov, "New test sequences for screen content coding", 15th JCT-VC meeting, document JCTVC-O0222, Geneva, Switzerland, Nov. 2013.

[30] H. -P. Yu, W. Wang, X. Wang, J. Ye, and Z. Ma, "AHG8: New 4:4:4 test sequences with screen content", 15th JCT-VC meeting, document JCTVC- O0256, Geneva, Switzerland, Nov. 2013.

[31] J. Guo, L. Zhao, and T. Lin, "Response to B1002 Call for test materials: Five test sequences for screen content video coding", 3th JVET meeting, document JVET-C0044, Geneva, Switzerland, May. 2016.

[32] K. Sharman, and K. Suehring, "Common test conditions", 24th JCT-VC meeting, document JCTVC-X1100, Geneva, Switzerland, May. 2016.

[33] 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, Florida, USA, Nov. 2014, pp. 675–678.

[34] 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. Computer Vision*, Santiago, Chile, Dec. 2015, pp. 1026–1034.

[35] D. P. Kingma and J. L. Ba, "Adam: a method for stochastic optimization," in *Proc. Int. Conf. Learning Represen*tations, San Diego, USA, May 2015, pp. 1–13.

[36] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell*., vol. 40, no. 4, pp. 834–848, Apr. 2018.

[37] DeepSCC: Deep learning based fast prediction network for screen content coding. Available at: http://www.eie.polyu.edu.hk/~ylchan/research/DeepSCC/.

[38] W. Ding, Y. Shi, and B. Yin, "YUV444 test sequences for screen content", 13th JCT-VC meeting, document JCTVC-M0431, Incheon, South Korea, Apr. 2013.

[39] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks". In *Proc. IEEE Int. Conf. Computer Vision*, Hawaii, USA, Mar. 2017, pp. 4700-4708.

**Wei Kuang** (S'17) received the B. S. degree in School of Electronic and Optical Engineering from Nanjing University of Science and Technology, Nanjing, China, in 2015. Now, he is currently pursuing the Ph.D. Degree in the Department of Electronic and Information Engineering at 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 360-degree omnidirectional video coding.

Dr. Chan serves as an Associate Editor of IEEE TRANSACTIONS ON IMAGE PROCESSING. He 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.

**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 in 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 also includes machine learning and deep learning.

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

Wan-Chi Siu (S'77-M'77-SM'90-F'12-Life-F'16) received the MPhil and PhD degrees from The Chinese University of Hong Kong in 1977 and Imperial College London in 1984. He is Life-Fellow of IEEE and Fellow of IET, and Immediate-Past President (2019-2020) of APSIPA (Asia-Pacific Signal and Information Processing Association). Prof. Siu is now Emeritus Professor, and was Chair Professor, Founding Director of Signal Processing Research Centre, Head of Electronic and Information Engineering Department and Dean of Engineering Faculty of The Hong Kong Polytechnic University. He is an expert in DSP, transforms, fast algorithms, machine learning, and conventional and deep learning approaches for super-resolution imaging, 2D and 3D video coding, object recognition and tracking. He has published 500 research papers (over 200 appeared in international journal papers), and edited three books. He has also 9 recent patents granted. Prof. Siu was an independent non-executive director (2000-2015) of a publicly-listed video surveillance company and convenor of the First Engineering/IT Panel of the RAE(1992/93) in Hong Kong. He is an outstanding scholar, with many awards, including the Best Teacher Award, the Best Faculty Researcher Award (twice) and IEEE Third Millennium Medal (2000). Prof. Siu has been Guest Editor/Subject Editor/AE for IEEE Transactions on Circuits and System II, Image Processing, Circuit & System for Video Technology, and Electronics Letters, and organized very successfully over 20 international conferences including IEEE society-sponsored flagship conferences, such as TPC Chair of ISCAS1997 and General Chair of ICASSP2003 and General Chair of ICIP2010. He was Vice-President, Chair of Conference Board and Core Member of Board of Governors (2012-2014) of the IEEE Signal Processing Society, and is now a member of the IEEE Educational Activities Board, IEEE Fourier Award for Signal Processing Committee and some other IEEE committees.