Content-Aware Convolutional Neural Network for In-Loop Filtering in High Efficiency Video Coding

Chuanmin Jia, *Student Member, IEEE*, Shiqi Wang, *Member, IEEE*, Xinfeng Zhang, *Member, IEEE*, Shanshe Wang, Jiaying Liu, *Senior Member, IEEE*, Shiliang Pu, and Siwei Ma, *Senior Member, IEEE*

***Abstract*— Recently, convolutional neural network (CNN) has attracted tremendous attention and has achieved great success in many image processing tasks. In this paper, we focus on CNN technology combined with image restoration to facilitate video coding performance and propose the content-aware CNN based in-loop filtering for high-efficiency video coding (HEVC). In particular, we quantitatively analyze the structure of the proposed CNN model from multiple dimensions to make the model interpretable and optimal for CNN-based loop filtering. More specifically, each coding tree unit (CTU) is treated as an independent region for processing, such that the proposed content-aware multimodel filtering mechanism is realized by the restoration of different regions with different CNN models under the guidance of the discriminative network. To adapt the image content, the discriminative neural network is learned to analyze the content characteristics of each region for the adaptive selection of the deep learning model. The CTU level control is also enabled in the sense of rate-distortion optimization. To learn the CNN model, an iterative training method is proposed by simultaneously labeling filter categories at the CTU level and fine-tuning the CNN model parameters. The CNN based in-loop filter is implemented after sample adaptive offset in HEVC, and extensive experiments show that the proposed approach significantly improves the coding performance and achieves up to 10.0% bit-rate reduction. On average, 4.1%, 6.0%, 4.7%, and 6.0% bit-rate reduction can be obtained under all intra, low delay, low delay P, and random access configurations, respectively.**

***Index Terms*— High-efficiency video coding (HEVC), in-loop filter, convolutional neural network.**

Manuscript received September 20, 2018; revised January 12, 2019; accepted January 22, 2019. Date of publication January 31, 2019; date of current version May 22, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61632001 and Grant 61872400, in part by the National Basic Research Program of China (973 Program) under Grant 2015CB351800, in part by the High-Performance Computing Platform of Peking University, Hong Kong RGC Early Career Scheme, under Grant 9048122 (CityU 21211018), in part by the City University of Hong Kong under Grant 7200539/CS, in part by the National Natural Science Foundation of China under Grant 61772043, and in part by the Beijing Natural Science Foundation under Grant L182002 and Grant 4192025. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Xin Li. *(Corresponding author: Shanshe Wang.)*

C. Jia, S. Wang, and S. Ma are with the Institute of Digital Media, Peking University, Beijing 100871, China (e-mail: cmjia@pku.edu.cn; sswang@pku.edu.cn; swma@pku.edu.cn).

S. Wang is with the Department of Computer Science, City University of Hong Kong, Hong Kong (e-mail: shiqwang@cityu.edu.hk).

X. Zhang is with the School of Computer and Control Engineering, University of the Chinese Academy of Sciences, Beijing 100049, China (e-mail: zhangxinf07@gmail.com).

J. Liu is with the Institute of Computer Science and Technology, Peking University, Beijing 100871, China (e-mail: liujiaying@pku.edu.cn).

S. Pu is with the Hikvision Research Institute, Hangzhou 310052, China (e-mail: pushiliang@hikvision.com).

Digital Object Identifier 10.1109/TIP.2019.2896489

I. Introduction

THE block-based compression framework has been widely adopted in many existing image/video coding standards, e.g., JPEG [1], H.264/AVC [2], AVS2 [3] and High Efficiency Video Coding (HEVC) [4]. However, the block-based prediction and quantization [1], [2], [4] in these existing compression framework will introduce the discontinuities along the block boundary, error information around texture contour, as well as the loss of high frequency details, which correspond to the blocking, ringing and blurring artifacts, respectively. The strengths of these artifacts are important determinants of video quality. Therefore, in-loop filtering plays a significant role to promote the reconstruction quality of decoded video, and the majority of state-of-the-art in-loop filtering algorithms are investigated with such purpose.

To suppress the blocking artifacts in video coding, the in-loop deblocking filters are investigated over the past several decades [3], [5]–[9], the philosophy of which mainly lies in designing low-pass filters to restrain the blocking artifacts by adaptively smoothing the boundary pixels. Moreover, the strength of deblocking filtering can also be determined by comparing the discontinuities between adjacent block boundaries with certain thresholds. The deblocking filters are originally applied for 4×4 block boundaries in H.264/AVC [2], [6]. The advanced deblocking filter is then designed in HEVC [4], [5], which is able to accommodate the quad-tree block partition process. Although deblocking filters can reduce the blocking artifacts efficiently by smoothing the boundary pixels, its application scope is restricted due to the design philosophy that only boundary pixels are processed while inner ones within a block have been largely ignored. Obviously, it is difficult to be applied in handling other kinds of artifacts (e.g., ringing and blurring). To compensate the artifacts induced by block-based transform and coarse quantization, several novel in-loop filtering methods such as sample adaptive offset (SAO) [10], adaptive loop filtering (ALF) [11] and image denoising based method [12]–[17] were investigated, all of which could efficiently remove the artifacts and obtain better coding performance.

Recently, deep learning (DL) [18], especially CNN, has been bringing revolutionary breakthrough in many tasks such as visual and textural data processing. Deep learning based

image restoration and denoising methods [19], [20] have also achieved state-of-the-art performances. Regarding image and video coding, which tend to become intelligence originated as well, DL based coding tools have been widely investigated, e.g., in the end-to-end image compression framework [21], sub-pixel interpolation [22], [23], inter-prediction [24], in-loop filtering [25], [26].

In this work, we aim to achieve the content-aware in-loop filter via the multiple CNN models and the corresponding discriminative network to well adapt to images with different content characteristics. In summary, the contributions of this paper are summarized as follows.

- A fully convolutional network architecture with inception structure [27] is analyzed and designed to enhance the quality of the reconstructed frame in video coding. With the proposed CNN structure, a content-aware loop filtering scheme based on multiple CNN models is proposed for high efficiency video coding.
- We employ the discriminative network to adaptively select the CNN model for each CTU. As such, the content adaptive selection of the appropriate filtering parameter is casted into a classification problem and solved with the data-driven DL approach.
- We investigate an iteratively training strategy to learn the multiple CNN models, which achieves simultaneous learning of the near-optimal model parameters as well as the content category. Extensive validations provide useful evidence regarding the effectiveness of the proposed approach.

The remainder of this paper is organized as follows. Section II elaborates the related work of the in-loop filtering technique and CNN based image restoration methods. Section III makes quantitative analysis on the proposed single CNN architecture. In Section IV, we present the proposed content-aware in-loop filter based on multiple CNN models. Extensive experimental results are reported in Section V. In Section VI, we conclude this paper.

II. Related Work

In this section, we briefly review the previous work related to the proposed approach. In particular, the existing in-loop filtering algorithms in the current video coding standards are firstly discussed, following which we detail the CNN based image restoration methods.

A. In-Loop Filtering in Video Coding

The quality degradation of the reconstructed video results from the lossy video compression framework. In particular, various kinds of artifacts are introduced in the block-based transform coding framework. To alleviate those kinds of distortions in the compression process, many in-loop filtering approaches have been studied to enhance both the objective and subjective quality. In this manner, better prediction efficiency can also be provided in the predictive video coding framework, in which previously filtered frames are used to predict the current one. Existing in-loop filtering techniques can be summarized from the following aspects.

- **Deblocking Filter.** Due to the quantization of block-based coding, the prediction error cannot be completely compensated. Therefore, the discontinuity often appears along the block boundaries, especially under low bit-rate coding circumstances. The design philosophy of deblocking filter [28], [29] is low-pass filtering the block boundaries to smooth the jagged and discontinuous edges or boundaries. Hence, deblocking filter has been adopted as a core coding tool since the video coding standards H.263+ [3], [5]–[9]. For low bit-rate video coding, Kim *et al.* [29] proposed an algorithm with two separate filtering modes, which are selected by pixel behavior around the block boundary. Recently, deblocking algorithms [30], [31] with higher flexibilities were also presented to determine the filter strength by content complexity estimation instead of boundary-pixel thresholds.
- **SAO and ALF.** The deblocking filters cannot adequately restore the quality degraded frame, since the inner pixels have been ignored in the deblocking process. In view of this, more in-loop filtering algorithms such SAO [10] and ALF [11] were proposed, which potentially take all pixels within each CTU into consideration. SAO belongs to the statistical algorithm, and the key idea is to compensate for the sample distortion by classifying the reconstructed samples into different categories. Inspired by Wiener filter theory [32]–[34], ALF aims to minimize distortions between the original and reconstructed pixels and trains the low-pass filter coefficients at the encoder. The coefficients are further transmitted to the decoder. To reduce the overhead of filter coefficients, temporal merge mode [35] and geometry transformation [36] were further investigated to boost the coding performance of ALF.
- **Image Prior Models.** The natural image statistical modeling has also pushed the horizon of in-loop filtering techniques. Krutz *et al.* [37] utilized the high order compensated temporal frames to improve the quality of the current coding picture. Zhang *et al.* [14] proposed a low rank based in-loop filter model which estimates the local noise distribution for coding noise removal. Ma *et al.* [13] investigated the non-local prior of natural images to generate GSR for collaborative filtering. To reduce the cost of transform coefficients by smoothing the residuals in JEM, Galpin *et al.* [38] modeled the errors induced by the clipping process and designed component-wise clipping bounds for each coded picture. Zhang *et al.* [39] explored a near optimal filtering mechanism for high efficiency image coding by iteratively labeling each pixel with near-optimal filters.

B. CNN for Image Restoration and Compression

Recently, efforts have been devoted to CNN based image restoration, which has been proved to achieve the state-of-the-art performance [40]. Zhang *et al.* [19] proposed a deep CNN approach with residual learning for image denoising. In [41], CNN was employed for ill-posed inverse problems in medical imaging. It is also shown that CNN models also achieved obvious quality enhancement for JPEG compressed images [20].

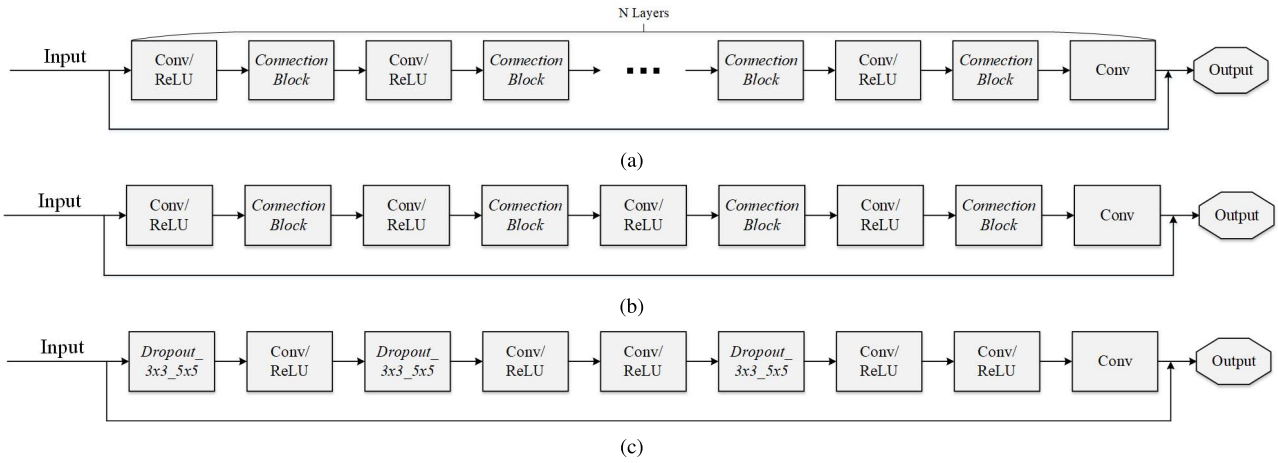


Fig. 1. Illustration of (a) network structure for depth analysis, N equals 7, 8, 9, 10, 11; (b) network structure of different connection unit types, each connection block is replaced by the four connection units in Fig. 2; (c) the proposed single CNN structure, which is the parameter reduction version of (b) with no performance degradation.

In addition to image restoration, preliminary yet very promising progress has been made in CNN based image/video compression [25], [26], [42], [43]. It has been incroporated into various modules in the hybrid video coding framework [22], [44]. To enhance the inter-prediction accuracy in video coding, Yan *et al.* [22] utilized a shallow CNN model for half-pixel interpolation. Post-processing methods for reconstructed video were explored in [25] and [43], all of which utilized CNN for quality enhancement outside of the coding loop. More specifically, Dai *et al.* [25] proposed VRCNN model as post-processing for HEVC intra coding. Yang *et al.* [43] investigated quality enhancement CNN (QE-CNN) under very low bit-rate coding configuration and achieved significant quality enhancement. The IFCNN [42], which firstly utilizes the CNN as the in-loop filtering for video coding. However, the unseparated training data and test data in [42] may result in a lack of generalization ability. Moreover, the CNNs could also be deployed for the view synthesis and restoration of immersive media content [45]. In [26], a spatial-temporal residual network (STResNet) was integrated into HEVC for in-loop filtering for coding performance enhancement.

III. Proposed Single CNN Model

In this section, we provide the designation philosophy and multi-dimensional analysis for the detail architecture of our proposed single CNN model. First, the extensive analysis for network depth is provided. Second, the connection unit type of CNN model is analyzed with respect to the aforementioned network depth, where the inception structure with variable kernel size is adopted. The performance of the proposed unit outperforms the other connection units proposed in [25] and [46]. Finally, we propose the single CNN model by reducing the number of parameters with no performance degradation on the top of the analysis.

A. Network Depth Analysis

In general, the DL community shares the common and simple philosophy that the deeper of the network, the better performance can be achieved, especially for the high level vision tasks [47], [48]. However, in the low-level problems such as image restoration and in-loop filtering, the depth should be determined by the network structure and specific application. To better understand how the depth of network influences the in-loop filtering performance, we conduct the empirical analysis to explore the network performance with different depths. We design several sets of plain networks with different depth (N) which are composed of a stack of convolutional (conv.) layers and Rectified Linear Unit (ReLU) [49] as activation function (except for the last layer), as illustrated in Fig. 1. It should be noted that there are 64 channels for the first $N-1$ layers for feature extraction and 1 channel for the last layer to reconstruct the image. In particular, the N-layer fully conv. network could be formulated as follows.

$$F_0(X) = X, \tag{1}$$

$$F_i(X) = ReLU(W_i * F_{i-1}(X) + b_i), \quad i = 1, 2, \ldots, N-1 \tag{2}$$

$$F_N(X) = W_N * F_{N-1} + b_N + X, \tag{3}$$

where X is the input patch, and W_i and b_i denote the weights and bias of the i-th conv. layer respectively. The $*$ represents conv. operation. In particular, it should be noted that there is no pooling layers in the restoration model since the pooling may cause irreversible loss of information which damages the reconstructed quality.

Within our analysis, it is worth mentioning that for fair comparison, except for the various depth of these networks, we control all other variables, including the training set, patch size, training hyper-parameters and hardware environment to be identical. Without loss of generality, we set quantization parameter (QP) to be 37 in our analysis. We first compressed the BSDS-500 dataset [50] using HEVC with QP=37 then randomly selected 380 images for training, 10 images for validation and the remaining for testing. Each image is partitioned into 38×38 patches with stride 22 for training and validation. Additionally, we deploy the trained model to recover the HEVC-compressed test images. The restoration ability of networks with various depth is shown in Table I,

TABLE I

AVERAGE RESTORATION GAIN (dB) OF DIFFERENT NETWORK DEPTH (N)

Connection Unit	N=7	N=8	N=9	N=10	N=11
Average PSNR	0.262	0.271	**0.292**	0.277	0.265

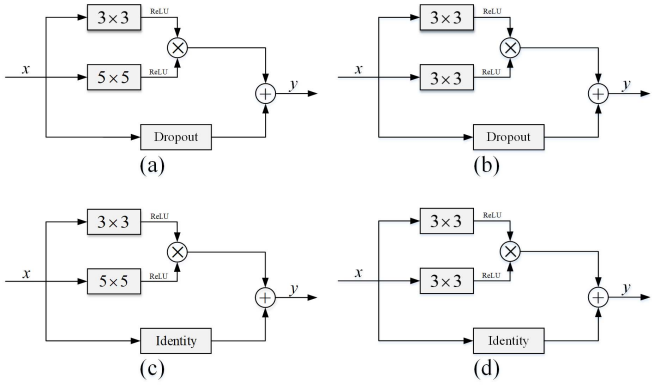


Fig. 2. Different connection unit type: (a) $dropout\_3 \times 3\_5 \times 5$: variable kernel size for inception with dropout; (b) $dropout\_3 \times 3\_3 \times 3$: identical kernel size for inception with dropout; (c) $identity\_3 \times 3\_5 \times 5$: variable kernel size for inception with identity connection; (d) $identity\_3 \times 3\_3 \times 3$: identical kernel size for inception with identity connection. It should be noted that the *product* in each subfigure denotes the concatenation operation.

from which we could learn that the networks with depth from seven to eleven conv. layers have similar restoration ability. However, the network of nine layers outperforms the others with little margin. Hence, we choose the depth of network to be nine in our model.

B. Proposed Connection Units

We illustrate the proposed connection unit as well as three typical types of connection units from literatures in Fig. 2, all of which contain inception structure. The major difference for such four connection units lies in the kernel size of inception structure and the residue connection type between the input and output of the unit. Regarding the designation of connection unit type, three major factors are considered during designation. The first design philosophy is inspired by the inception structure [27] which introduced the variable conv. kernel size for different branches to extract variable-size features. Second, following [51], relatively small receptive fields (3×3 and 5×5) are adopted within our model for more non-linearity. In addition, we further deploy the residue connection [47] from the input to realize residual learning, such that convergence speed during the training stage can be accelerated.

As shown in Fig. 2(a), our proposed connection unit has 3×3 and 5×5 conv. kernels for two branches respectively. Subsequently, the two branches are concatenated before output (we deployed *same* padding mechanism for the conv. to guarantee the consistent spatial resolution as the input such that concatenation could be easily realized). Meanwhile, the dropout connection is deployed between the input and output of the unit (we define dropout probability to be 1). The same kernel size (both 3×3) is utilized for two branches in Fig. 2(b). For Fig. 2(c) and (d), the kernel sizes for conv. layers keep the same as Fig. 2(a) and (b) respectively. The only difference is the residue connection becomes identity connection (shortcut) between input and output of each unit.

TABLE II

AVERAGE RESTORATION GAIN dB OF FOUR DIFFERENT TYPES OF CONNECTION UNITS IN FIG. 2

Connection Unit	Fig. 2(a)	Fig. 2(b)	Fig. 2(c)	Fig. 2(d)
Average PSNR	**0.41**	0.35	0.38	0.11

To validate the effectiveness of proposed connection units against the other three different types, we conduct comparisons for the restoration ability of the four different connection types. For the fair comparison as in the previous stage, all other variables are controlled except for the connection type. The network depth is set to be 9 according to previous analysis. As depicted in Fig. 1(b), each connection block is replaced by the aforementioned units for restoration ability comparison. The average PSNR gain of four models could be found in Table II. It is clear that our proposed connection unit type shows the best restoration ability in terms of PSNR gain. Hence, this category of analysis shows the efficiency of proposed connection unit type.

C. Parameter Reduction

Based on the previous two categories of quantitative comparisons, we have demonstrated that the network described in Fig. 1(b), which is with depth of 9 and the multi-scale inception structures ($dropout\_3 \times 3\_5 \times 5$ connection unit type) can achieve the highest restoration performance.

However, since there are too many parameters and feature maps in the optimal choice (64 channels for each layer), there must be redundant kernels and it is not appropriate to utilize excessive number of parameters for the in-loop filtering. Hence, it is urgent and necessary for us to reduce the number of parameters to achieve similar restoration performance. We first reduce the number of feature map channels from 64 to 32 in each branch of $droppout\_3 \times 3\_5 \times 5$ unit and re-train the model. It is worth noting that there is no max pooling in our single CNN model since the max pooling operation will lose the information and make it difficult for signal restoration. Subsequently, the locations of first two $dropout\_3 \times 3\_5 \times 5$ units are changed into the 1st and 3rd layer. Finally, the 4th $dropout\_3 \times 3\_5 \times 5$ unit in Fig. 1(b) is reduced into single 3×3 conv. layer to further reduce the number of conv. kernel parameter. Hence, the structure of the proposed single CNN model is shown in Fig. 1(c).

Analogous to the previous two subsections, in this analysis, we also control all other variables and the only difference is the network structure in Fig. 1(b) and Fig. 1(c). We plot the restoration performance with different iterations for the two models in Fig. 3. When evaluating the restoration performance on the testing images, we could observe that the parameter-reduced version (green curve in Fig. 3) only has negligible performance loss comparing with the original optimal model in Fig. 1(b) (red curve in Fig. 3).

TABLE III
PARAMETER SETTINGS OF THE PROPOSED CNN MODEL

Index	Layer1		Layer2	Layer3		Layer4	Layer5	Layer6		Layer7	Layer8	Layer9
Layer Type	Conv1(1)	Conv1(2)	Conv2	Conv3(1)	Conv3(2)	Conv4	Conv5	Conv6(1)	Conv6(2)	Conv7	Conv8	Conv9
Receptive Field	3×3	5×5	3×3	3×3	5×5	3×3	3×3	3×3	5×5	3×3	3×3	3×3
Feature Map Number	32	32	64	32	32	64	64	32	32	64	64	64
Stride	1											
Padding	1	2	1	1	2	1	1	1	2	1	1	1

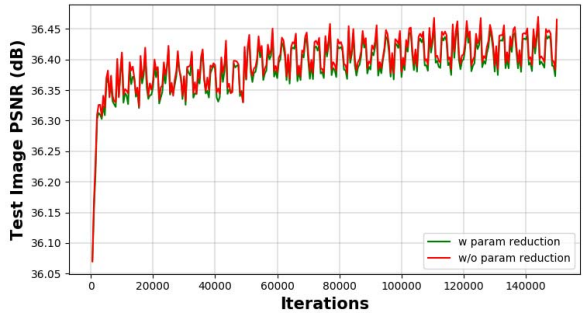


Fig. 3. Performance comparison with/without parameter reduction.

TABLE IV
HYPER-PARAMETERS SETTING FOR SINGLE CNN MODEL TRAINING

Hyper-param \ Interval	QP=22	QP=27	QP=32	QP=37
Base Learning Rate (base_lr)	0.01	0.01	0.01	0.1
Gamma	0.5			
Step size (Epochs)	15			
λ	0.001			
Momentum/Momentum2	0.9/0.99			
Training Epochs	50			

IV. CONTENT-AWARE CNN BASED IN-LOOP FILTERING

In this section, the proposed content-aware in-loop filtering based on CNN is detailed, as shown in Fig. 4. In particular, the CNN model applied for each CTU is derived totally adaptive according to CTU content. As such, better coding performance can be achieved with the locally adaptive in-loop filtering. First, the network architecture of a single CNN model for in-loop filtering is presented. Subsequently, the detailed network configuration of each layer and the hyper-parameters during training are presented. Due to the inherent limitation of single CNN model, the multiple CNN model based in-loop filtering is proposed to adapt different texture characteristics. To optimally select the appropriate CNN model, a discriminative network is learned to infer the CNN model for each CTU. Moreover, to simultaneously learn the network parameters and model category, an iteratively training mechanism is further introduced.

A. From Single Model CNN to Multiple Model CNN

The network architecture of the proposed single CNN model is shown in Fig. 5. To provide a more intuitive interpretation of the single model, the network configurations for each conv. layer are illustrated in Table III. The sliding step (stride) value is 1 pixel for each layer. To maintain the resolution consistency before and after each conv. layer, zero padding is utilized. In particular, the padding is 1 pixel for 3×3 conv. layers and 2 pixels for 5×5 conv. layers respectively. Each conv. layer adopts ReLU for non-linearity activation except for the last conv. layer (Eqn.(3)).

1) Single CNN Model Training: We utilize the BSDS-500 dataset for our single CNN model training. All images in BSDS-500 are compressed by the standard HEVC[1] (HM-16.9) with common test condition (CTC) in all intra (AI) configuration. To distinguish different quality levels, the networks are individually trained in terms of different QP intervals. In particular, we train the optimal CNN models for each QP interval by employing all the training data compressed with every QP in the corresponding interval. Specifically, each QP interval contains 5 consecutive QP values. For simplicity, we utilize the smallest QP value to denote the corresponding QP interval in this paper. During inference, the model is selected by mapping the QP value of current slice to the closest QP interval, i.e., when the QP of current slice is 25, then the model of QP=22 is used. The total dataset is splitted into two folds: 380 randomly-selected images for training and 20 images for validation. To generate the training samples for single CNN model, all compressed images are cropped into 38×38 small patches with stride 22.

Here, let (x_i, y_i) denote the i-th training sample, where x_i is the HEVC compressed patch and y_i denotes the corresponding pristine one. Therefore, the objective function of the CNN training is to minimize the Euclidean-loss function,

$$Ł(\Theta) = \frac{1}{N}\sum_{i=1}^{N} ||\Phi(x_i|\Theta) - y_i||_2^2 + \beta||\Theta||_2^2, \tag{4}$$

where Θ encapsulates the weights and biases of the network and $\Phi(x_i|\Theta)$ denotes the single CNN model. Moreover, L_2 norm regularization term in Eqn.(4) is introduced to prevent overfitting during training, and β is the penalty factor. The first-order gradient based method Adam [52] is chosen to optimize the objective function, which is able to adaptively adjust the gradient and updating the results for each parameter.

The widely adopted DL framework Caffe [53] is utilized to train our models. Hyper-parameter settings of our proposed single CNN model are listed in Table IV. The progressive training methodology is adopted during training. Specifically, we first train the model for interval QP=37 completely from scratch and use it as the initialized network parameters to train the model for smaller QP intervals, etc. The base learning rate (base_lr) is set to be 0.1 for QP=37 while 0.01 for other QP intervals. The hyper-parameter Gamma is the degradation factor for base learning rate, which implies that the learning rate

[1]The HEVC-compressed BSDS-500 dataset for training is provided at http://jiachuanmin.site/Projects/CNNLF/TrainingData/

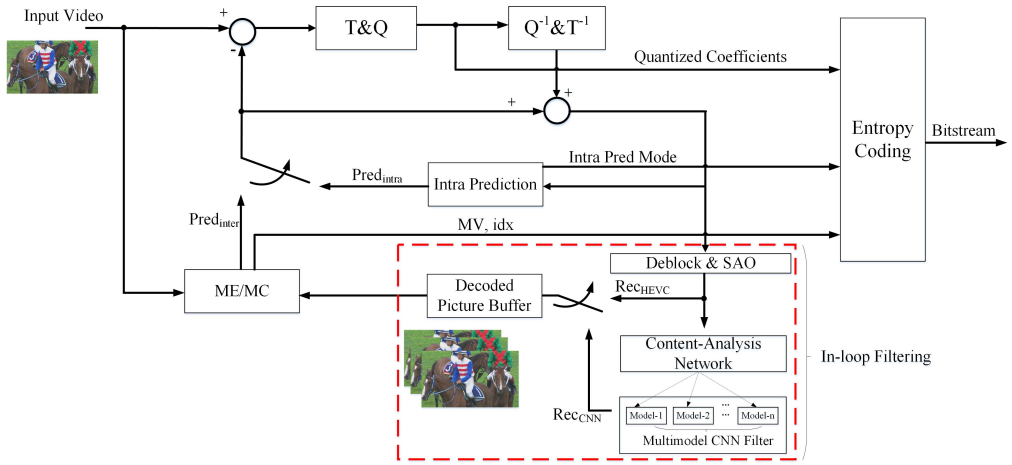


Fig. 4. The framework of the proposed content-aware CNN based in-loop filtering within the hybrid coding framework of HEVC.

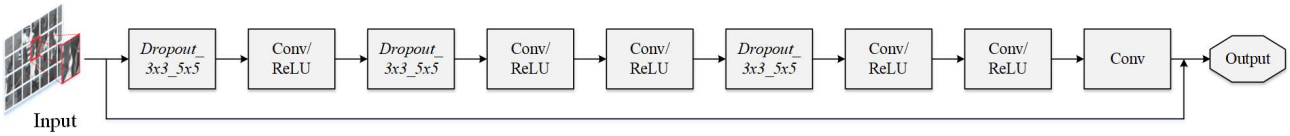


Fig. 5. Network structure of the proposed single CNN model where the *droppout*_3 × 3_5 × 5 unit has two separate branches with 3 × 3 and 5 × 5 kernel size respectively.

discounts 50% every 15 epochs (Step size) during training. The *Momentum* and *Momentum*2 are two hyper-parameters for Adam when adaptively calculating gradients during training. During the network training process, we first train the model of each interval with a fixed learning rate (base_lr) for 50 epochs and then use stepping learning rate (base_lr starts from 0.1) for another 50 epochs until its convergence.

2) Drawbacks of Single CNN: For each CTU, the rate-distortion (R-D) performance of the in-loop filtering can be expressed as follows,

$$J_{CTU} = \Delta D_{CTU} + \lambda R, \tag{5}$$

$$\Delta D_{CTU} = D'_{CTU} - D_{CTU}, \tag{6}$$

where D'_{CTU} and D_{CTU} denote the distortion after and before CNN based in-loop filtering, ΔD_{CTU} is the distortion variation after the CNN based in-loop filtering, R indicates the coding bits for signaling the loop filter control flag, and the Langrange multiplier λ controls the tradeoff between rate and distortion. As such, the performance of the loop filtering is reflected by J_{CTU}, and lower J_{CTU} indicates better restoration performance. In Fig. 6, the quality variation map is illustrated, where both quality improvement and degradation can be observed. It is obvious that one single model is not able to handle the diverse content in a frame, and there are also some areas suffering from performance loss with the single CNN model.

To address this issue and improve the adaptability of the in-loop filtering, we propose an advanced in-loop filtering technique with multiple CNN candidate models. As such, each CTU can adaptively select the optimal CNN model to achieve better restoration performance. To offline learn the CNN models, an iterative training scheme is proposed. More specifically, the training samples can be classified into several

Fig. 6. Illustrations of the limitations of a single CNN model. Green CTUs indicate performance improvement after filtering, and red ones correspond to performances loss after filtering.

categories in a data-driven manner, leading to the multiple CNN models that cover a wide range of content characteristics. In order to derive the appropriate model for each CTU at both encoder and decoder sides, a discriminative network is adopted to infer the optimal CNN model, such that explicitly signaling of the model index can be also avoided.

B. Discriminative Network for Model Selection

The discriminative network (Discrimnet) for CNN model selection is implemented based on a light-weighted modification of Alexnet [48]. The network structure of Discrimnet is depicted in Fig. 7. More specifically, there are 5 conv. layers (variable receptive fields, 11 × 11, 5 × 5 and 3 × 3) and 2 max-pooling layers with kernel size 3 × 3. Batch normalization [54] is also used after each pooling layer for faster convergence. The numbers of feature map for each conv. layer are 96, 96, 192, 192, 128. It should be noted that we add ReLU as the activation function after all conv. layers and fully connected (fc) layers (except for the last fc layer). The stride value for four conv. layers are 4, 2, 1, 1 and there is no padding for the first two layers. More details regarding the descrimnet can be found in Table V.

TABLE V
PARAMETER SETTINGS OF THE DISCRIMNET

Index	Layer1	Layer2	Layer3	Layer4	Layer5	Layer6	Layer7	Layer8	Layer9	Layer10
Layer Type	Conv1	Pool2	Conv2	Pool2	Conv3	Conv4	Conv5	fc6	fc7	fc8
receptive Field	11×11	5×5	5×5	3×3	3×3	3×3	3×3	–	–	–
Feature Map Number	96	96	96	128	192	192	128	–	–	–
Stride	4	1	2	2	1	1	1	–	–	–
Padding	0	0	3	1	1	1	1	–	–	–

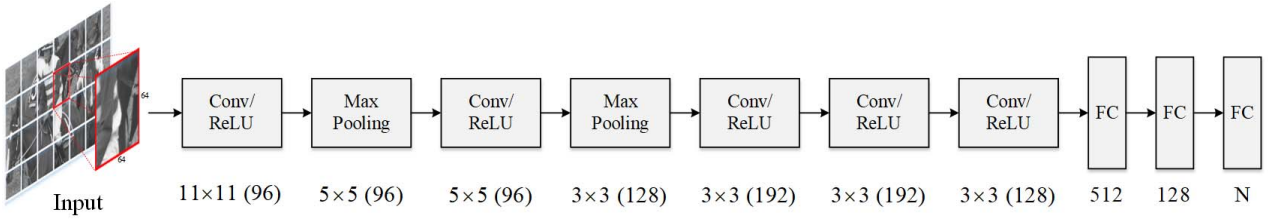


Fig. 7. The network structure for the discriminative neural network (N denotes the number of multimodels). The conv. kernel size and the corresponding channel number (non-fc layer) or vector dimension (fc layer) are listed in the bottom.

Discrimnet takes each CTU as the input and produces an N-dimension feature vector (i.e., $(\alpha_1, \alpha_2, \ldots, \alpha_N)$) for classification, where N denotes the number of CNN models. To choose the best model from the candidate ones, softmax operation ($f(\alpha_i) = \frac{e^{\alpha_i}}{\sum_{j=1}^{N} \alpha_j}, \quad i = \{1, \ldots, N\}$) is then applied for each element of the N-dimension feature vector. And optimal CNN model is determined by the index of largest element after softmax. With the help of Discrimnet, the model selection is casted into the classification problem and solved in a data-driven manner. The training details of Discrimnet will be introduced in the next subsection.

C. Multimodel Iterative Training Mechanism

To obtain the multiple CNN models, a novel training scheme is proposed to iteratively optimize the CNN model parameters and model categories simultaneously. The initialization process of the proposed training mechanism is firstly introduced, which includes the single CNN model training, quality ranking of the restored images, and the fine-tuning.

- **Single CNN Model Training.** An initial single CNN model is first trained with the training data from BSDS-500, as described in Section III-A.
- **Quality Ranking of the Restored Images.** The learned single CNN model is used to restore all training samples generated from the BSDS-500 dataset. The quality difference in terms of peak-signal-noise-ratio (PSNR) before and after single CNN filtering is calculated as follows,

$$\Delta\psi_i = \bar{\psi}_i - \psi_i, \tag{7}$$

where $\bar{\psi}_i$ denotes the PSNR value of the $i-$th training sample x_i after single CNN model filtering, and ψ_i denotes the PSNR before filtering. As such, all training samples can be ranked by $\Delta\psi_i$ in the descending order.
- **Fine-Tuning.** To generate N initialized multimodels, the ranked training samples are equally partitioned into N-folds. Each fold of the partitioned training samples is utilized to fine-tune the single CNN model. The hyper-parameters for fine-tuning remain the same as the single CNN model training. Hence, we can obtain N initialized CNN models after the convergence of the fine-tuning process.

Algorithm 1: Iterative Training Mechanism

Input: Initialized N CNN models, training samples (x_i, y_i) generated from BSDS-500, iteration time K.
Output: N fine-tuned models.
$m = 0$;
while $m < K$ **do**
  for i **in** *all training samples* **do**
    Filtering x_i with each CNN model;
    Obtain $Idx(x_i)$ according to Eqn.(8);
    Label x_i with index $Idx(x_i)$ for training sample clustering;
  end
  for j **from** 1 **to** N **do**
    Generate the $j-$th fold of training samples by clustering samples with same index $Idx(x_i)$;
    Fine-tune and update the $j-$th model with $j-$th fold of clustered training samples;
  end
  $m = m + 1$;
end

As such, we can obtain the N initialized CNN filter models, based on which each training sample can be labeled with an index,

$$Idx(x_i) = \arg\max_j [\Delta\psi^j], \quad j = 0, \ldots, N-1. \tag{8}$$

The index $Idx(x_i)$ denotes the appropriate CNN model that should be choosen to accommodate the content characteristics of x_i. It is also worth mentioning here only the distortion is considered as the coding bit in R-D optimization can be regarded as the constant when using different selected CNN models.

1) Iterative Training Mechanism: Based on the initialization process, the proposed iteratively training mechanism is achieved by fine-tuning the multiple CNN models and labeling the index simultaneously within each iteration. The algorithm is summarized in **Algorithm 1**.

TABLE VI
SYNTAX ELEMENT DESIGN OF THE PROPOSED IN-LOOP FILTER

MMCNN_parameter_Set()	Descriptor
Y_MMCNN_on	1 bit
Cb_MMCNN_on	1 bit
Cr_MMCNN_on	1 bit
if(Y_MMCNN_on){	
for (i=0;i<NumOfCTUs; ++i){	
CTU_MMCNN_on[i]	1 bit
}	
}	

TABLE VII
HYPER-PARAMETERS SETTING FOR DISCRIMNET TRAINING

Hyper-param \ Interval	QP=22	QP=27	QP=32	QP=37
Base Learning Rate (base_lr)	0.1	0.1	0.1	0.1
Gamma	0.1			
Step size (Epochs)	10			
λ	0.005			
Momentum/Momentum2	0.9/0.99			
Training Epochs	50			

More specifically, in each iteration, the N CNN models are fine-tuned by the adaptively partitioning the training samples, such that the fine-tuned models can be optimized with different texture characteristics. After the iterative training process, the corresponding N CNN models can be obtained.

2) Training Discrimnet: Different from CNN models training, the Discrimnet adopts uncompressed color image dataset (UCID) [55] to generate training samples. UCID contains $1,328$ images, and we randomly choose $1,200$ for training and the remaining for validation. Again, all the test images are first compressed by standard HEVC intra coding and then cropped into 64×64 small patches. The compressed images and the corresponding labels derived in Eqn.(8) are used for training the Discrimnet. To train the Discrimnet, we directly adopt the stepping learning rate strategy, starting from 0.1 with a degradation factor 0.1 for every 10 epochs. The hyper-parameter settings for Discrimnet are listed in Table VII.

D. Syntax Element Design for CTU Control Flag

To ensure the optimal performance in the R-D sense, the syntax element control flags in CTU-level and frame-level are particularly designed for the proposed in-loop filtering. For CTU-level control, a flag is added for each CTU to enable the proposed in-loop filtering to provide better local adaptation, i.e., CTU_MMCNN_on[i] for the i-th CTU. In particular, when the rate-distortion performance of the filtered CTU becomes better, the corresponding control flag is enabled, indicating that the proposed in-loop filtering is applied to this CTU. Otherwise, the flag is disabled and the proposed scheme is not applied to this CTU. After the determination of all the CTUs in one frame, the frame-level RD cost reduction for each color channel is calculated as

$$J_c = D_c + \lambda R_c, \quad \bar{J}_c = \bar{D}_c + \lambda \bar{R}_c, \tag{9}$$

where D_c and $\bar{D}_c$ indicate the frame-level distortion before and after the proposed in-loop filtering, respectively. R_c and $\bar{R}_c$ denote the coding bits of the two scenarios, and λ is the Lagrange multiplier. If $J_c > \bar{J}_c$, the frame-level flag is enabled, indicating that the proposed in-loop filtering is applied to the current frame. As such, the corresponding frame-level and CTU-level control flags are signaled into the bitstream. Otherwise, the frame-level flag is disabled, and the CTU-level control flags will not be further encoded.

Considering both the coding efficiency and complexity, we propose to adopt the CTU-level control for luminance component and frame-level control for chroma components in the proposed scheme. The syntax element structure of the control flags is shown in Table VI. All of the frame-level flags are encoded within the slice header of the bitstream after the syntax elements for SAO, and the CTU-level flags are embedded into each corresponding CTU syntax. It should be noted that if the frame-level flag for luma channel is false, we do not send the CTU flags anymore since the filtering for the entire frame is switched off.

V. EXPERIMENTAL RESULTS

To validate the efficiency of the proposed scheme, we integrate the proposed content-aware in-loop filtering into the HEVC reference software HM-16.9. In this section, both objective evaluations and subjective visualizations are firstly provided, and the BD-rate [56] performance of proposed method is illustrated and compared with other CNN based in-loop filter algorithms. Subsequently, the encoding and decoding complexity as well as the run-time GPU memory bandwidth usage are presented. Finally, we present the impact of frame-level syntax by empirical analysis.

A. Testing Conditions

The Caffe [53] library is integrated into HM-16.9 to perform the in-loop filtering with the CNN models. More specifically, the proposed in-loop filtering is incroporated after SAO process. The multiple CNN models are applied in the luminance channel only, and the single CNN model is adopted by the two chroma channels. Moreover, models for different color components are trained independently in the proposed method.

Four typical QP values are tested, including 22, 27, 32, 37. The experiments are conducted under HEVC CTC with all intra (AI), low-delay (LDB), low-delay P (LDP) and random access (RA) configurations. The anchor for all experiments is HEVC reference software (HM-16.9) with both deblocking and SAO enabled. The HEVC test sequences [57] from Class A to Class E are used, and the total length sequences are compressed for performance validation. Moreover, the iteration time K is set to be 2.

B. Objective Evaluations

The in depth objective evaluations are conducted in this subsection from multiple perspectives. The BD-rate reductions of the proposed single CNN model are shown in the first row of Table VIII, where it can be observed that 3.0%, 3.9%, 3.7% and 3.9% bit-rate savings can be achieved for AI, LDB, LDP and RA configurations, respectively. The performances with

TABLE VIII

OVERALL PERFORMANCE OF THE PROPOSED MULTIPLE CNN MODEL ($N = 1, 2, 4, 6$) IN-LOOP FILTERING (ANCHOR: HM-16.9)

Sequences	AI			LDB			LDP			RA		
	Y	U	V	Y	U	V	Y	U	V	Y	U	V
N=1	-3.0%	-2.8%	-3.4%	-3.9%	-2.1%	-2.3%	-3.7%	-1.0%	-1.6%	-3.9%	-2.1%	-2.3%
N=2	-3.3%	-3.2%	-4.2%	-4.9%	-2.8%	-3.3%	-4.3%	-1.1%	-1.6%	-4.9%	-3.1%	-3.7%
N=4	-3.7%	-3.1%	-3.8%	-5.4%	-2.5%	-3.0%	-4.6%	-0.9%	-1.1%	-5.4%	-2.9%	-3.4%
N=6	-3.9%	-3.5%	-4.2%	-5.9%	-2.4%	-3.0%	-4.7%	-0.7%	-1.3%	-5.8%	-2.6%	-3.3%

TABLE IX

PERFORMANCE OF THE PROPOSED MULTIPLE CNN MODEL ($N = 8$) IN-LOOP FILTERING (ANCHOR: HM-16.9)

Sequences	AI			LDB			LDP			RA		
	Y	U	V	Y	U	V	Y	U	V	Y	U	V
Class A	-4.7%	-3.3%	-2.6%	-6.7%	-2.6%	-1.9%	-3.5%	0.2%	0.3%	-6.6%	-3.4%	-3.0%
Class B	-3.5%	-2.8%	-3.0%	-5.7%	-1.6%	-2.2%	-4.5%	-0.5%	-1.1%	-6.5%	-2.5%	-2.7%
Class C	-3.4%	-3.5%	-5.0%	-5.0%	-3.4%	-5.0%	-4.4%	-1.9%	-3.0%	-4.5%	-3.3%	-4.5%
Class D	-3.2%	-4.7%	-6.0%	-3.8%	-1.7%	-2.6%	-3.5%	-0.8%	-0.9%	-3.3%	-2.6%	-3.6%
Class E	-5.8%	-4.1%	-5.2%	-8.6%	-5.2%	-5.6%	-7.7%	-1.7%	-0.9%	-9.0%	-4.2%	-5.3%
Overall	-4.1%	-3.7%	-4.1%	-6.0%	-2.9%	-3.5%	-4.7%	-1.0%	-1.2%	-6.0%	-3.2%	-3.8%

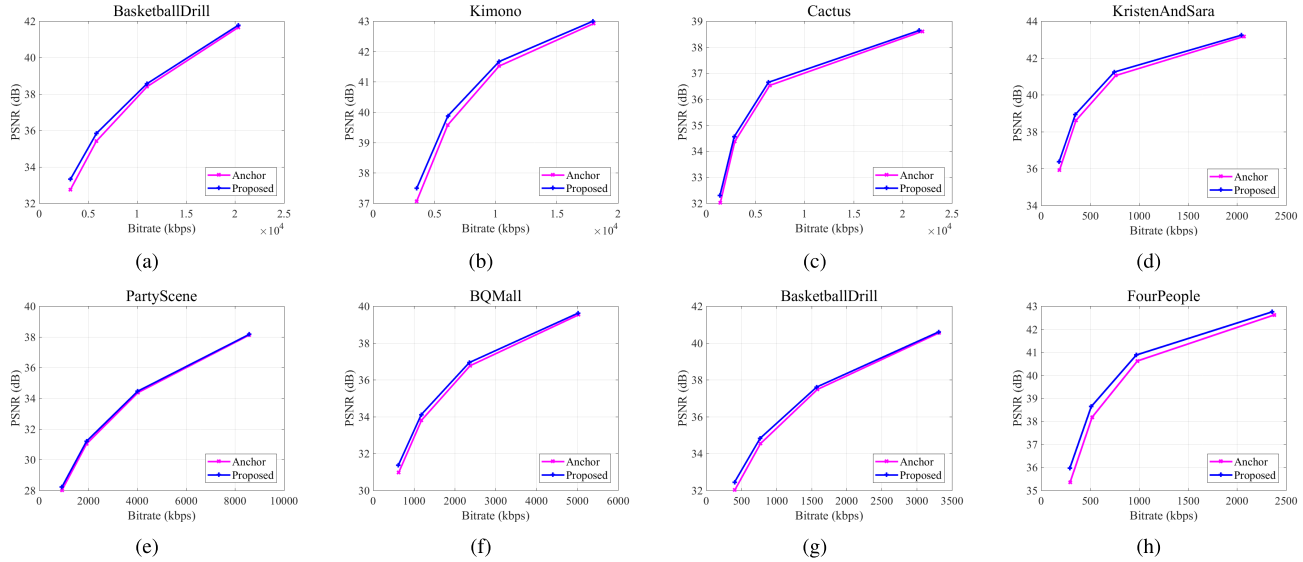


Fig. 8. Performance comparisons for different sequences with $N = 8$ (Anchor: HM-16.9). (a) *BasketballDrill*, AI; (b) *Kimono*, AI; (c) *Cactus*, LDB; (d) *KristenAndSara*, LDB; (e) *PartyScene*, RA; (f) *BQMall*, RA; (g) *BasketballDrill*, RA; (h) *FourPeople*, RA.

the increasing of the number of CNN models ($N = 2, 4, 6$) are reported in the second to the last rows of Table VIII. We should note that the performance in Table VIII are the overall average performance of Class A-E. It is obvious that with a certain range of N, the coding performance grows with the increasing number of CNN models, and the performance becomes saturated when the number of models is larger than 4. Moreover, the performance of the proposed scheme ($N = 8$) with CTU level control for each sequence is shown in Table IX. In particular, 4.1%, 6.0%, 4.7% and 6.0% bit-rate reductions on average for luma channel under four different coding configurations respectively. Based on Tables VIII and IX, the effectiveness of the proposed CNN based in-loop filtering method could be proved. With the growing number of N, the proposed iterative training method also expresses the content aware capability to adapt to different content characteristics. Moreover, more than 10% bit-rate reduction can be achieved in inter-coding for the sequence *FourPeople*.

In Fig. 9, the performance of the proposed scheme as well as the upper-bound performance are plotted in terms of the number of CNN models adopted (N). Here, the upper-bound is achieved by selecting the best model through full R-D based decision for all models at the encoder side. Under such circumstance, the best model index is not signaled to approach the ideal performance that can be obtained when the discriminative model is 100% accuracy. From this figure we can observe that the performance of the proposed scheme turns out to be consistent when $N > 6$. Moreover, the performance of the proposed scheme is quite close to the upper-bound performance, which further provides evidence that the Discrimnet

TABLE X
THE RATE-DISTORTION PERFORMANCE COMPARISON WITH VRCNN [25] AND VDSR [58] IN RA CONFIGURATION (ANCHOR: HM-16.9)

Sequences	VDSR [58]			VRCNN [25]			Proposed (N=8)		
	Y	U	V	Y	U	V	Y	U	V
Class A	-0.9%	-0.2%	0.0%	-1.1%	-0.2%	-0.1%	-6.6%	-3.4%	-3.0%
Class B	-2.1%	0.1%	-0.3%	-2.8%	0.0%	-0.2%	-6.5%	-2.5%	-2.7%
Class C	-2.3%	-1.4%	-2.1%	-3.0%	-1.3%	-2.4%	-4.5%	-3.3%	-4.5%
Class D	-2.1%	-1.2%	-1.4%	-2.3%	-0.5%	-1.0%	-3.3%	-2.6%	-3.6%
Class E	-4.7%	-0.3%	-0.6%	-6.1%	-0.7%	-0.5%	-9.0%	-4.2%	-5.3%
Overall	-2.5%	-0.5%	-1.0%	-3.1%	-0.5%	-0.9%	**-6.0%**	**-3.2%**	**-3.8%**

TABLE XI
THE LUMA PERFORMANCE COMPARISON BETWEEN ALF [11] AND PROPOSED METHOD ($N = 8$) (ANCHOR: HM-16.9)

Sequences	HEVC+ALF vs. HEVC				HEVC+Proposed vs. HEVC				HEVC+Proposed+ALF vs. HEVC+ALF			
	AI	LDB	LDP	RA	AI	LDB	LDP	RA	AI	LDB	LDP	RA
Class A	-3.9%	-4.9%	-7.1%	-4.8%	-4.7%	-6.7%	-3.5%	-6.6%	-2.7%	-3.2%	-3.2%	-3.1%
Class B	-2.1%	-3.0%	-6.2%	-3.4%	-3.5%	-5.7%	-4.5%	-6.5%	-1.6%	-2.5%	-2.9%	-2.7%
Class C	-1.5%	-1.8%	-2.3%	-2.1%	-3.4%	-5.0%	-4.4%	-4.5%	-3.4%	-4.0%	-3.8%	-3.7%
Class D	-0.7%	-1.4%	-0.7%	-2.2%	-3.2%	-3.8%	-3.5%	-3.3%	-3.2%	-3.4%	-3.7%	-3.4%
Class E	-3.1%	-3.5%	-4.7%	-3.6%	-5.8%	-8.6%	-7.7%	-9.0%	-4.3%	-5.8%	-6.2%	-5.3%
Overall	-2.0%	-2.7%	-4.0%	-3.0%	-4.1%	-6.0%	-4.7%	-6.0%	-2.9%	-3.7%	-3.8%	-3.6%

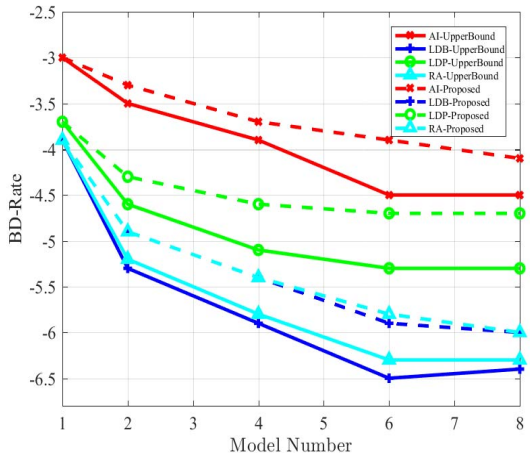


Fig. 9. Variations of the BD-Rate performance with the number of models. Dash line: performance of the proposed scheme; solid line: the corresponding upper-bound performance.

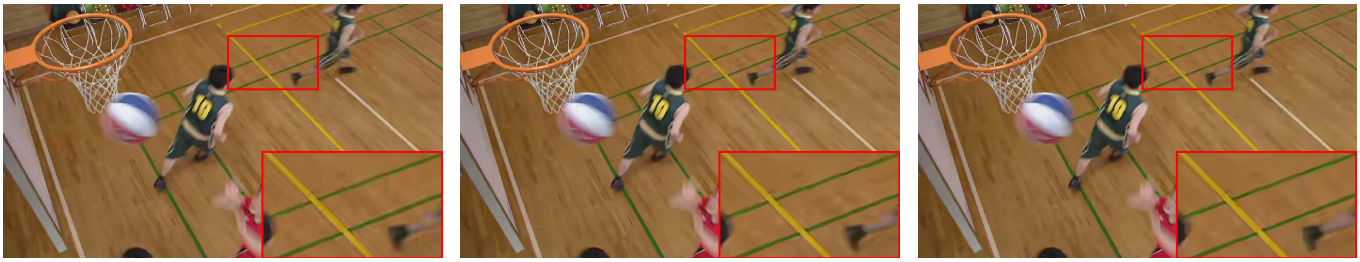
is effective in selecting the appropriate CNN model for in-loop filtering. The rate-distortion performance comparisons are also depicted in Fig. 8, which show that the proposed method achieves good generalization ability and obtains consistent coding performance under different configurations.

Furthermore, we compare the proposed scheme with other CNN structures using as in-loop filtering algorithms, including VRCNN [25] and VDSR [58]. In particular, VRCNN is a five-layer fully conv. network (FCN) and VDSR is with very deep FCN intentionally designed for single image super-resolution (SISR). From Table X, the proposed method achieves 6.0% coding gain for luma component while the approaches in [25] and [58] obtain 3.1% and 2.5% BD-rate reduction respectively. It is clear that our proposed scheme outperforms these two representative CNN based algorithms under RA configurations.

In addition, the performance comparison with non-CNN based in-loop filter approach (such as ALF) is also conducted to show the effectiveness of the proposed method. Specifically, we compare the coding performance of ALF and proposed method in terms of BD-rate reduction. It is worth mentioning that the simulation platform is HM-16.9 with CTC and the coding performance of luma channel is reported. From the Table XI, we could learn that ALF obtains 2.0%, 2.7%, 4.0% and 3.0% BD-rate reduction for the four different configurations respectively while the proposed multimodel method saves 4.1%, 6.0%, 4.7% and 6.0% BD-rate for each case. Obviously, the proposed approach outperforms the ALF and achieves better coding efficiency. To further verify the performance for both ALF and CNN base loop filter, we conduct another set of experiments to show that the proposed method could also achieve 2.9%, 3.7%, 3.8% and 3.6% BD-rate reductions when ALF is also enabled (we should note that the CNN is located between SAO and ALF in this category of experiment), which convinces us that the proposed method is a competitive coding tool for next generation video coding standard. From the last four columns of Table XI, we could learn that the CNN based loop filtering algorithm has the compatibility with conventional Wiener filter based approach since it also achieves promising coding gain on the top of ALF.

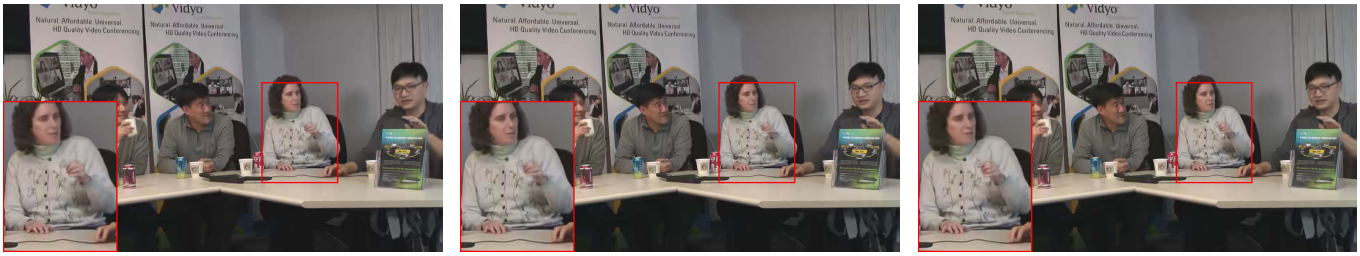
C. Subjective Evaluations

The subjective comparisons is usually necessary for evaluating the loop filtering algorithms in video coding. In this subsection, we further compare the visual quality of reconstructed frames in Fig. 10 and 11. Two frames from $BasketballDrill$ and $FourPeople$ are used for illustration. The frame is also cropped for better visualization. It is obvious that our proposed scheme can efficiently remove different kinds of compression artifacts due to the generalization ability of deep learning



(a) (b) (c)

Fig. 10. Visual quality comparison for *BasketballDrill* with RA configuration, where the 9th frame is shown (QP=37). (a) Original; (b) VRCNN; (c) The proposed scheme ($N = 8$).



(a) (b) (c)

Fig. 11. Visual quality comparison for *FourPeople* with LDB configuration, where the 16th frame is shown (QP=37). (a) Original; (b) VRCNN; (c) The proposed scheme ($N = 8$).

based approaches. Moreover, scrupulous observers may find that the degraded structures during block-based coding can also be recovered by the proposed scheme, e.g., the texture of floor and straight-lines. The underlying reason lies in that our knowledge-based scheme is able to recover the missing information. Moreover, the proposed network has the capability of effectively representing high order information of the visual signals and reconstructing the missing details, due to larger parameter quantity and variable receptive field layers. As such, the frames processed by the proposed scheme have better visual quality than the state-of-the-art methods by a clear margin.

D. Complexity

In this subsection, the encoding and decoding complexity as well as the GPU memory consumption of the proposed scheme are reported. To evaluate the time complexity of our algorithm, we test the proposed algorithm with hyper-threading off and record the encoding and decoding (enc/dec) time. The testing environment is Intel i7 4770k CPU and the latest version of Caffe is incroporated [53]. We utilize the NVIDIA GeForce GTX TITAN X GPU for testing and the GPU-memory is 12 GB. Moveover, the operating system is Windows 10 64-bit home basic and the memory for the PC is 24 GB. The HEVC reference software and Caffe are compiled with the Visual Studio 2013 ultimate version.

When evaluating the coding complexity overhead, the ΔT is calculated as,

$$\Delta T = \frac{T' - T}{T}, \tag{10}$$

where T is original enc/dec time of HEVC reference software (HM-16.9) and T' is the proposed enc/dec time. All of the complexity evaluations are conducted with GPU acceleration, such that the forward operation of the CNN filtering and Discrimnet is operated by GPU, and the remaining operations are performed by CPU. From Table XII,we can observe that on average the encoding complexity overhead is 113%, while the decoding overhead is 11656%. The proposed scheme greatly influences the decoding time because of the forward operation in network and CPU-GPU memory copy operation. Moreover, the fully-connected layer within Discrimnet also imposes great complexity to the decoding process. The encoding complexity for VDSR [58] and VRCNN [25] is 135% and 110% respectively while the ALF [11] only increases 4% encoding time with respect to HEVC baseline. And the decoding complexity for the three methods are 13753%, 4459%, 123% respectively.

The CNN model storage consumption and run-time GPU memory bandwidth are also listed in Table XII. The model size consists of three parts, Discrimnet, multi CNN models of luma and single model for two chroma channels. It is also worth noting that the model sizes are identical for AI, LDB, LDP and RA configurations. Specifically, the size of each CNN model is 1.38MB while each Discrimnet model is 10.80MB. The total model size equals ($N \times 1.38 + 1.38 + 1.38 + 10.8$MB) where N is the model numbers. Hence, it takes 14.94$\sim$20.6MB to store the trained models for each QP interval. Regarding VDSR [58] and VRCNN [25], the model size is 2.54MB and 0.21MB respectively. Since the proposed method has another Discrimnet for each case, which contains fc. layers for classification. Hence, the average model size of the proposed method is larger but still in a reasonable range than the two existing method. As for GPU memory bandwidth usage, 370$\sim$1428MB run-time GPU memory are needed when N ranging from 1 to 8. VDSR [58] consumes 1022 MB run-time GPU resources while that value of VRCNN [25] is 155 MB. Further optimization can be realized by pruning as well as decomposing the weight matrixes of the trained deep models to speed up network forward operation.

TABLE XII
ENCODING AND DECODING COMPLEXITY AND GPU MEMORY CONSUMPTION OF THE DIFFERENT IN-LOOP FILTERING APPROACHES

Model Number	ΔT_{enc}	ΔT_{dec}	Model Size	GPU-Memory
$N = 1$	111%	8891%	14.94 MB	370 MB
$N = 2$	113%	9820%	16.32 MB	496 MB
$N = 4$	114%	11497%	19.08 MB	778 MB
$N = 6$	113%	13061%	21.84 MB	1053 MB
$N = 8$	114%	15010%	24.60 MB	1428 MB
Average	113%	11656%	—	
Approach	ΔT_{enc}	ΔT_{dec}	Model Size	GPU-Memory
VDSR [58]	135%	13753%	2.54 MB	1022 MB
VRCNN [25]	110%	4459%	0.21 MB	155 MB
ALF [11]	104%	123%	—	

TABLE XIII
THE IMPACT OF FRAME-LEVEL SYNTAX

Configuration	AI	LDB	LDP	RA
Class A	0.0%	0.3%	0.3%	0.3%
Class B	0.0%	0.2%	0.2%	0.2%
Class C	0.0%	0.1%	-0.1%	0.0%
Class D	0.0%	0.0%	0.1%	0.0%
Class E	0.0%	2.4%	1.9%	1.8%
Average	0.0%	0.5%	0.4%	0.4%

Regarding the potential investigations of fast algorithms in the future work, there are particularly two major approaches for decreasing the complexity. First, inspired by the recent advances in neural network inference acceleration [59], [60], we could deploy the related emerging techniques such as pruning, weights quantization and matrix decomposition to avoid the float-point operation in our current model, which could significantly reduce the decoder run-time. Second, the inference of out trained CNN models relied on the third-party DL framework. In our future work, we will reduce the overhead introduced by the interaction between the HEVC codec and the DL framework. With the two above methods, the run-time complexity for the decoder is expected to be reduced.

E. Impact of Frame-Level Syntax

In the proposed method, the frame-level flags are used for each channel. There are two major reasons for introducing frame-level syntax for CNN based in-loop filtering. First, we conducted empirical analysis to illustrate the effectiveness of frame-level syntax. We keep the CTU-level flags the same as the proposed method and always turn on the frame-level syntax. The reference is the proposed single model in our paper. As shown in Table XIII, if the frame-level syntax is always turned on, the luma channel coding performance loss will be 0.0%, 0.5%, 0.4% and 0.4% for AI, LDB, LDP and RA configurations respectively. Moreover, we could observe that the BD-rate keeps the same for AI configuration. This is because no motion compensation is used in intra coding hence there is no error propagation for the ill-filtered frames. However, regarding the inter-coding cases, the previously coded frames are used as reference frames for motion compensation. Hence, we need the frame-level flag mechanism to stop error propagation. From the experimental results, we could also observe that the frame-level syntax is more useful and necessary in the motion-compensation based coding configurations including LDB, LDP and RA. Second, since the SAO, which is an existing in-loop filtering coding tool in HEVC, utilizes the frame-level to ensure the coding efficiency. Similar rules and syntax designation should also be applied for CNN based in-loop filtering techniques to keep the design consistency with the coding tools in existing standards. Hence, the frame-level flag is necessary.

VI. CONCLUSIONS

In this paper, we propose a content-aware CNN based in-loop filtering algorithm for HEVC. The novelty of this paper lies in that multiple CNN models are offline trained and applied in the adaptive in-loop filtering process, and a discriminative deep neural network is also trained to select the optimal CNN model. Moreover, we provide the corresponding quantitative analysis on the design philosophy of our proposed filter network structure. The iterative training scheme is further proposed to learn the optimal CNN models and the corresponding content characteristics category for different content simultaneously. Extensive experimental results demonstrate that the proposed content-aware CNN based in-loop filtering algorithm outperforms other DL based approaches, achieving the state-of-the-art performance under HEVC CTC configurations.

ACKNOWLEDGMENT

The authors would like to thank the associate editor and anonymous reviewers for their constructive comments that significantly helped in improving the presentation of the manuscript of this paper and thank Dai *et al.* [25] and Kim *et al.* [58] for kindly sharing their implementation for performance comparison. The authors would also like to express their gratitude to Shurun Wang and Shuaiyu Liang for the help of running simulations.

REFERENCES

[1] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Trans. Consum. Electron.*, vol. 38, no. 1, pp. xviii–xxxiv, Feb. 1992.
[2] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, 2003.
[3] S. Ma, T. Huang, C. Reader, and W. Gao, "AVS2? Making video coding smarter [standards in a nutshell]," *IEEE Signal Process. Mag.*, vol. 32, no. 2, pp. 172–183, Mar. 2015.
[4] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.
[5] A. Norkin *et al.*, "HEVC deblocking filter," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1746–1754, Dec. 2012.
[6] P. List, A. Joch, J. Lainema, G. Bjontegaard, and M. Karczewicz, "Adaptive deblocking filter," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 614–619, Jul. 2003.
[7] G. Cote, B. Erol, M. Gallant, and F. Kossentini, "H.263+: Video coding at low bit rates," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 7, pp. 849–866, Nov. 1998.
[8] S.-M. Lei, T.-C. Chen, and M.-T. Sun, "Video bridging based on H.261 standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, no. 4, pp. 425–437, Aug. 1994.
[9] L. Fan, S. Ma, and F. Wu, "Overview of AVS video standard," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, vol. 1, Jun. 2004, pp. 423–426.

[10] C.-M. Fu *et al.*, "Sample adaptive offset in the HEVC standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1755–1764, Dec. 2012.

[11] C.-Y. Tsai *et al.*, "Adaptive loop filtering for video coding," *IEEE J. Sel. Topics Signal Process.*, vol. 7, no. 6, pp. 934–945, Dec. 2013.

[12] J. Zhang, D. Zhao, and W. Gao, "Group-based sparse representation for image restoration," *IEEE Trans. Image Process.*, vol. 23, no. 8, pp. 3336–3351, Aug. 2014.

[13] S. Ma, X. Zhang, J. Zhang, C. Jia, S. Wang, and W. Gao, "Nonlocal in-loop filter: The way toward next-generation video coding?" *IEEE MultiMedia*, vol. 23, no. 2, pp. 16–26, Apr./Jun. 2016.

[14] X. Zhang *et al.*, "Low-rank-based nonlocal adaptive loop filter for high-efficiency video compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 10, pp. 2177–2188, Oct. 2017.

[15] X. Zhang, R. Xiong, X. Fan, S. Ma, and W. Gao, "Compression artifact reduction by overlapped-block transform coefficient estimation with block similarity," *IEEE Trans. Image Process.*, vol. 22, no. 12, pp. 4613–4626, Dec. 2013.

[16] X. Zhang, W. Lin, R. Xiong, X. Liu, S. Ma, and W. Gao, "Low-rank decomposition-based restoration of compressed images via adaptive noise estimation," *IEEE Trans. Image Process.*, vol. 25, no. 9, pp. 4158–4171, Sep. 2016.

[17] X. Zhang, R. Xiong, W. Lin, S. Ma, J. Liu, and W. Gao, "Video compression artifact reduction via spatio-temporal multi-hypothesis prediction," *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 6048–6061, Dec. 2015.

[18] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[19] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3142–3155, Jul. 2017.

[20] C. Dong, Y. Deng, C. C. Loy, and X. Tang, "Compression artifacts reduction by a deep convolutional network," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 576–584.

[21] J. Ballé, V. Laparra, and E. P. Simoncelli. (2016). "End-to-end optimized image compression." [Online]. Available: https://arxiv.org/abs/1611.01704

[22] N. Yan, D. Liu, H. Li, and F. Wu. (2017). "A convolutional neural network approach for half-pel interpolation in video coding." [Online]. Available: https://arxiv.org/abs/1703.03502

[23] J. Liu, S. Xia, W. Yang, M. Li, and D. Liu, "One-for-all: Grouped variation network-based fractional interpolation in video coding," *IEEE Trans. Image Process.*, vol. 28, no. 5, pp. 2140–2151, May 2019.

[24] Y. Wang, X. Fan, C. Jia, D. Zhao, and W. Gao, "Neural network based inter prediction for HEVC," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2018, pp. 1–6.

[25] Y. Dai, D. Liu, and F. Wu, "A convolutional neural network approach for post-processing in HEVC intra coding," in *Proc. Int. Conf. Multimedia Modeling*. Reykjavik, Iceland: Springer, 2017, pp. 28–39.

[26] C. Jia, S. Wang, X. Zhang, S. Wang, and S. Ma. (2017). "Spatial-temporal residue network based in-loop filter for video coding." [Online]. Available: https://arxiv.org/abs/1709.08462

[27] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.

[28] B. Ramamurthi and A. Gersho, "Nonlinear space-variant postprocessing of block coded images," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 34, no. 5, pp. 1258–1268, Oct. 1986.

[29] S. D. Kim, J. Yi, H. M. Kim, and J. B. Ra, "A deblocking filter with two separate modes in block-based video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 1, pp. 156–160, Feb. 1999.

[30] H. Jo, S. Park, and D. Sim, "Parallelized deblocking filtering of HEVC decoders based on complexity estimation," *J. Real-Time Image Process.*, vol. 12, no. 2, pp. 369–382, 2016.

[31] H. Karimzadeh and M. Ramezanpour, "An efficient deblocking filter algorithm for reduction of blocking artifacts in HEVC standard," *Int. J. Image, Graph. Signal Process.*, vol. 8, no. 11, pp. 18–24, 2016.

[32] T. Kailath, "A view of three decades of linear filtering theory," *IEEE Trans. Inf. Theory*, vol. IT-20, no. 2, pp. 146–181, Mar. 1974.

[33] D. Slepian, "Linear least-squares filtering of distorted images," *J. Opt. Soc. Amer.*, vol. 57, no. 7, pp. 918–922, 1967.

[34] S. O. Haykin, *Adaptive Filter Theory*, vol. 2. Upper Saddle River, NJ, USA: Prentice-Hall, 2002, pp. 478–481.

[35] X. Zhang, R. Xiong, S. Ma, and W. Gao, "Adaptive loop filter with temporal prediction," in *Proc. IEEE Picture Coding Symp. (PCS)*, May 2012, pp. 437–440.

[36] M. Karczewicz, L. Zhang, W.-J. Chien, and X. Li, "Geometry transformation-based adaptive in-loop filter," in *Proc. IEEE Picture Coding Symp. (PCS)*, Dec. 2016, pp. 1–5.

[37] A. Krutz, A. Glantz, M. Tok, M. Esche, and T. Sikora, "Adaptive global motion temporal filtering for high efficiency video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1802–1812, Dec. 2012.

[38] F. Galpin, P. Bordes, and F. Racape, "Adaptive clipping in JEM," in *Proc. IEEE Data Compress. Conf. (DCC)*, Apr. 2017, pp. 33–41.

[39] X. Zhang, S. Wang, Y. Zhang, W. Lin, S. Ma, and W. Gao, "High-efficiency image coding via near-optimal filtering," *IEEE Signal Process. Lett.*, vol. 24, no. 9, pp. 1403–1407, Sep. 2017.

[40] L. Zhang and W. Zuo, "Image restoration: From sparse and low-rank priors to deep priors [lecture notes]," *IEEE Signal Process. Mag.*, vol. 34, no. 5, pp. 172–179, Sep. 2017.

[41] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser. (2016). "Deep convolutional neural network for inverse problems in imaging." [Online]. Available: https://arxiv.org/abs/1611.03679

[42] W.-S. Park and M. Kim, "CNN-based in-loop filtering for coding efficiency improvement," in *Proc. IEEE Image, Video, Multidimensional Signal Process. Workshop (IVMSP)*, Jul. 2016, pp. 1–5.

[43] R. Yang, M. Xu, and Z. Wang, "Decoder-side HEVC quality enhancement with scalable convolutional neural network," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2017, pp. 817–822.

[44] 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, pp. 5088–5103, Nov. 2016.

[45] C. Jia, X. Zhang, S. Wang, S. Wang, S. Pu, and S. Ma, "Light field image compression using generative adversarial network based view synthesis," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, to be published.

[46] Y. Zhang, T. Shen, X. Ji, Y. Zhang, R. Xiong, and Q. Dai, "Residual highway convolutional neural networks for in-loop filtering in HEVC," *IEEE Trans. Image Process.*, vol. 27, no. 8, pp. 3827–3841, Aug. 2018.

[47] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[48] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[49] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 807–814.

[50] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, May 2011.

[51] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556

[52] D. P. Kingma and J. Ba. (2014). "Adam: A method for stochastic optimization." [Online]. Available: https://arxiv.org/abs/1412.6980

[53] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22nd ACM Int. Conf. Multimedia*, 2014, pp. 675–678.

[54] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[55] G. Schaefer and M. Stich, "UCID: An uncompressed color image database," *Proc. SPIE*, vol. 5307, pp. 472–481, Dec. 2003.

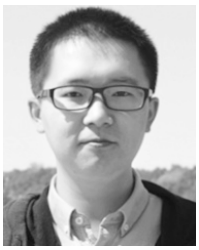
[56] G. Bjontegaard, *Calculation of Average PSNR Differences Between RD-Curves*, document ITU-T Q. 6/SG16 VCEG, 15th Meeting, Austin, TX, USA, Apr. 2001.

[57] F. Bossen, *Common Test Conditions and Software Reference Configurations*, document Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 5th Meeting, Jan. 2011.

[58] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 1646–1654.

[59] S. Han, H. Mao, and W. J. Dally. (2015). "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding." [Online]. Available: https://arxiv.org/abs/1510.00149

[60] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. (2016). "SqueezeNet: AlexNet-level accuracy with 50× fewer parameters and <0.5MB model size." [Online]. Available: https://arxiv.org/abs/1602.07360



Chuanmin Jia (S'18) received the B.E. degree in computer science from the Beijing University of Posts and Telecommunications, Beijing, China, in 2015. He is currently pursuing the Ph.D. degree with the Department of Computer Science, Peking University, Beijing.

In 2018, he joined the Video Lab, New York University, NY, USA, as a Visiting Student. His research interests include video compression, light field compression, and machine learning. He received the Best Paper Award at the Pacific-Rim Conference on Multimedia in 2017 at the *IEEE Multimedia Magazine* in 2018.

Shiqi Wang (M'15) received the B.S. degree in computer science from the Harbin Institute of Technology in 2008 and the Ph.D. degree in computer application technology from Peking University in 2014. From 2014 to 2016, he was a Post-Doctoral Fellow with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. From 2016 to 2017, he was a Research Fellow with the Rapid-Rich Object Search Laboratory, Nanyang Technological University, Singapore. He is currently an Assistant Professor with the Department of Computer Science, City University of Hong Kong. He has authored over 40 technical proposals in ISO/MPEG, ITU-T, and AVS standards. His research interests include image/video compression, analysis, and quality assessment.

Xinfeng Zhang (M'16) received the B.S. degree in computer science from the Hebei University of Technology, Tianjin, China, in 2007, and the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2014. From 2014 to 2017, he was a Research Fellow with the Rapid-Rich Object Search Lab, Nanyang Technological University, Singapore. He is currently a Post-Doctoral Fellow with the School of Electrical Engineering System, University of Southern California, Los Angeles, CA, USA. His research interests include image and video processing, and image and video compression.

Shanshe Wang received the B.S. degree from the Department of Mathematics, Heilongjiang University, Harbin, China, in 2004, the M.S. degree in computer software and theory from Northeast Petroleum University, Daqing, China, in 2010, and the Ph.D. degree in computer science from the Harbin Institute of Technology. He held a post-doctoral position with Peking University, Beijing, from 2016 to 2018. He joined the School of Electronics Engineering and Computer Science, Institute of Digital Media, Peking University, where he is currently a Research Assistant Professor. His current research interests include video compression, and image and video quality assessment.

Jiaying Liu (S'08–M'10–SM'17) received the B.E. degree in computer science from Northwestern Polytechnic University, Xi'an, China, in 2005, and the Ph.D. degree (Hons.) in computer science from Peking University, Beijing, China, in 2010.

She was a Visiting Scholar with the University of Southern California, Los Angeles, from 2007 to 2008. She was a Visiting Researcher with Microsoft Research Asia in 2015, supported by the Star Track for Young Faculties. She is currently an Associate Professor with the Institute of Computer Science and Technology, Peking University. She has authored over 100 technical articles in refereed journals and proceedings, and holds 28 granted patents. Her current research interests include image/video processing, compression, and computer vision. She has also served as a TC Member for the IEEE CAS-MSA/EOT and APSIPA IVM, and was an APSIPA Distinguished Lecturer from 2016 to 2017. She is a Senior Member of CCF.

Shiliang Pu received the Ph.D. degree from the University of Rouen in 2005. He is currently a Chief Research Scientist with Hikvision and the President of the Hikvision Research Institute. His research interests include AI, machine perception, and robotics.

Siwei Ma (M'03–SM'12) received the B.S. degree from Shandong Normal University, Jinan, China, in 1999, and the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2005. He held a post-doctoral position with the University of Southern California, Los Angeles, CA, USA, from 2005 to 2007. He joined the School of Electronics Engineering and Computer Science, Institute of Digital Media, Peking University, Beijing, where he is currently a Professor. He has authored over 200 technical articles in refereed journals and proceedings in image and video coding, video processing, video streaming, and transmission. He is an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY and the *Journal of Visual Communication and Image Representation*.