Efficient In-Loop Filtering Based on Enhanced Deep Convolutional Neural Networks for HEVC

Zhaoqing Pan, *Senior Member, IEEE*, Xiaokai Yi, Yun Zhang, *Senior Member, IEEE*, Byeungwoo Jeon, *Senior Member, IEEE*, and Sam Kwong, *Fellow, IEEE*

***Abstract*— The raw video data can be compressed much by the latest video coding standard, high efficiency video coding (HEVC). However, the block-based hybrid coding used in HEVC will incur lots of artifacts in compressed videos, the video quality will be severely influenced. To settle this problem, the in-loop filtering is used in HEVC to eliminate artifacts. Inspired by the success of deep learning, we propose an efficient in-loop filtering algorithm based on the enhanced deep convolutional neural networks (EDCNN) for significantly improving the performance of in-loop filtering in HEVC. Firstly, the problems of traditional convolutional neural networks models, including the normalization method, network learning ability, and loss function, are analyzed. Then, based on the statistical analyses, the EDCNN is proposed for efficiently eliminating the artifacts, which adopts three solutions, including a weighted normalization method, a feature information fusion block, and a precise loss function. Finally, the PSNR enhancement, PSNR smoothness, RD performance, subjective test, and computational complexity/GPU memory consumption are employed as the evaluation criteria, and experimental results show that when compared with the filter in HM16.9, the proposed in-loop filtering algorithm achieves an average of 6.45% BDBR reduction and 0.238 dB BDPSNR gains.**

***Index Terms*— Convolutional neural networks, high efficiency video coding, in-loop filtering.**

I. Introduction

THE appearance of the video coding standard, H.265/HEVC, has greatly improved the efficiency of video compression. Compared with the previous video coding standard, H.264/AVC [1], HEVC achieves approximately double compression ratio [2]. However, with the widespread applications of multimedia technology, the multimedia data has exploded. Even if HEVC can extremely compress the raw video data, the limited bandwidth and storage space have greatly restricted the spread of compressed video at the same time. Therefore, in order to improve the subjective video quality as much as possible with lower bitrate, the optimizations for HEVC are necessary.

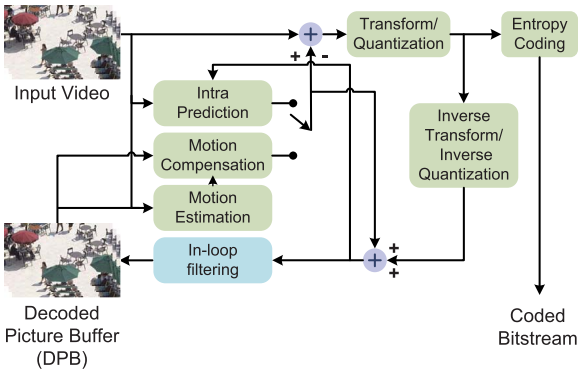


Fig. 1. The in-loop filtering in HEVC.

Since the block-based hybrid coding used in HEVC [3], the compression artifacts have come into the encoded video, such as blocking artifacts, ringing artifacts, color excursion, and so on. To address these artifact problems, the in-loop filtering is adopted in HEVC for significantly eliminating those artifacts, and enhancing the reconstructed video quality. Fig. 1 shows an example of the in-loop filtering in HEVC. It consists of two parts: deblocking filter (DF) [4] and sample adaptive offset (SAO) [5]. The purpose of DF is to reduce blocking artifacts by performing adaptive filters for different boundary types. And the DF is reported to achieve an average of 2.3% BD-rate reduction with the same video quality [6]. The main function of SAO is to attenuate ringing artifacts by adding an offset for each reconstructed sample of its category, and the SAO is shown to achieve an average of 3.5% BD-rate reduction [6]. To intuitively show the achievements of in-loop filtering, we perform comparisons on various in-loop filtering techniques, and the results are shown in Fig. 2. We can see that the compressed picture without filtering has the smallest PSNR for the objective evaluation, and from the subjective analysis, it has lots of artifacts, such as block artifacts in the horse, and ringing artifacts in the man. For the compressed picture

Manuscript received April 4, 2019; revised August 12, 2019, October 29, 2019, and January 18, 2020; accepted March 18, 2020. Date of publication March 27, 2020; date of current version April 2, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61971232, in part by the Six Talent Peaks Project of Jiangsu Province under Grant XYDXXJS-041, in part by the Project through the Priority Academic Program Development of Jiangsu Higher Education Institutions, and in part by the Collaborative Innovation Center of Atmospheric Environment and Equipment Technology. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Yun He. *(Corresponding author: Zhaoqing Pan.)*

Zhaoqing Pan and Xiaokai Yi are with the School of Computer and Software, Jiangsu Collaborative Innovation Center on Atmospheric Environment and Equipment Technology, Engineering Research Center of Digital Forensics, Ministry of Education, Nanjing University of Information Science and Technology, Nanjing 210044, China (e-mail: zhaoqingpan@nuist.edu.cn; xyi@nuist.edu.cn).

Yun Zhang is with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China (e-mail: yun.zhang@siat.ac.cn).

Byeungwoo Jeon is with the School of Electronic and Electrical Engineering, Sungkyunkwan University, Suwon 440746, South Korea (e-mail: bjeon@skku.edu).

Sam Kwong is with the Department of Computer Science, City University of Hong Kong, Hong Kong, China (e-mail: cssamk@cityu.edu.hk).

Digital Object Identifier 10.1109/TIP.2020.2982534

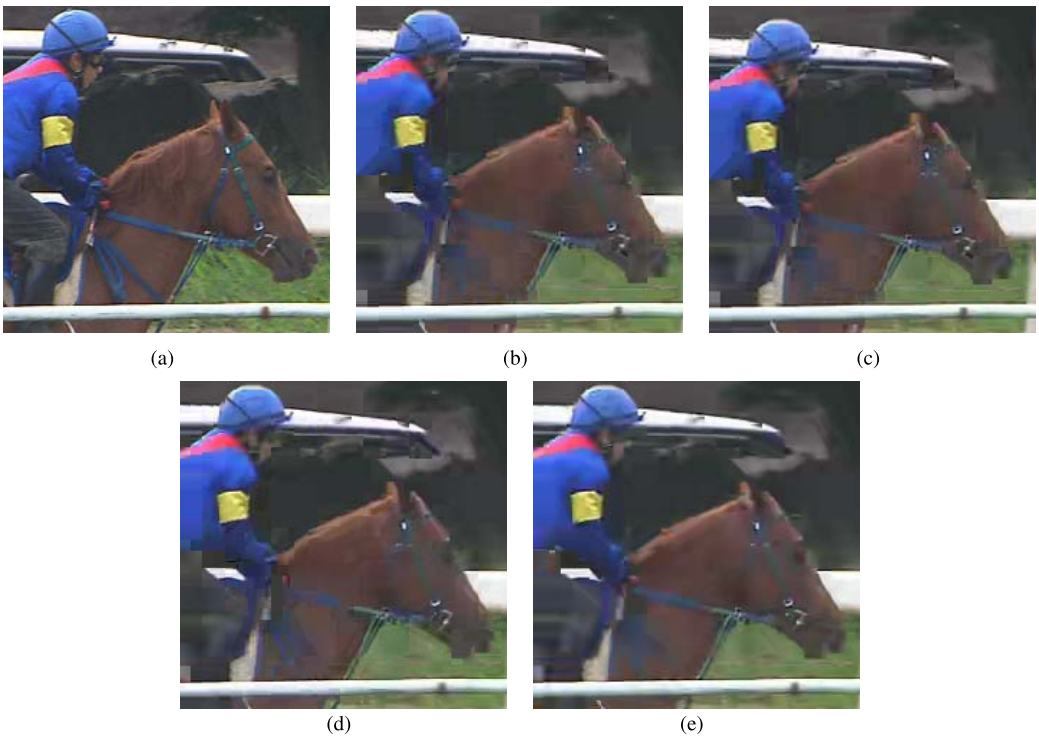


Fig. 2. The performance comparison among different loop filtering algorithms. (The 10th frame of Keiba encoded by the low-delay coding structure with QP37). (a) Original picture. (b) Compressed picture without filtering (PSNR: 30.61 dB). (c) Compressed picture with the DF (PSNR: 30.73 dB). (d) Compressed picture with the SAO (PSNR: 30.66 dB). (e) Compressed picture with DF+SAO (PSNR: 30.78 dB).

with the DF, the obvious blocking artifact in Fig. 2 (b) is eliminated. And as shown in Fig. 2 (d), the obvious ringing artifact in Fig. 2 (b) is eliminated by using the SAO. We can observe that the in-loop filtering can efficiently enhance the quality of compressed video.

Inspired by the success of deep learning, a large number of convolutional neural networks (CNN) [7] have emerged, and they have shown that CNN achieves excellent performance in image processing. A typical CNN model is composed of different layers, including input layer, hidden layer, and output layer. Among these layers, the hidden layer plays an important role, it is used to obtain the local information of images. Through the combination of different convolutions in the hidden layer, the mapping relationship between the input and output can be accurately modeled. There are several notable CNN based image restoration and image denoising works been proposed. Dong *et al.* [8] proposed an SRCNN for super-resolution by using the CNN to learn an end-to-end image mapping, and it turns out that it can generate higher resolution images than traditional methods. For further improving the reconstruction results, the FSRCNN was also proposed by Dong *et al.* [9] as an extension of the SRCNN. The FSRCNN uses the deconvolution layer to replace the bicubic interpolation, and it can compensate for the loss of the low resolution to high resolution, moreover, smaller filter sizes and more mapping layers are added for accelerating the training. Kim *et al.* [10] proposed a deep CNN model, named VDSR, whose model architecture is deepened by using 20 weight layers, and it achieves an outstanding performance. Zhang *et al.* [11] proposed a denoising convolutional neural networks (DnCNN), in which the residual learning is used to remove the latent clean image from the noise observation, meanwhile, the residual learning and batch normalization are used to speed up the training process, and boost the denoising performance. Zhang *et al.* [12] also proposed a fast and flexible denoising network (FFDNet) to handle various noise levels, in which the downsampled sub-images and tunable noise level maps are adopted as the input. The noisy image is downsampled to speed up the training process, and the tunable noise level map is added to reduce various noise levels as well as spatially variant noise. Since these CNN models have not fully considered the optimization problems of in-loop filtering, they hardly maximize the visual quality enhancement in HEVC.

In this paper, an efficient in-loop filtering algorithm based on optimized enhanced deep CNN (EDCNN) is proposed. The proposed EDCNN based in-loop filtering algorithm employs efficient network optimization methods to establish a precise mapping relationship between the reconstructed and uncompressed videos. Overall, the main contributions of this paper are summarized as follows.

- The advantages and disadvantages of existing CNN based in-loop filtering methods are analyzed. For obtaining high quality reconstruction results, some efficient CNN optimization methods are adopted, including a weighted normalization method, a feature information fusion block, and a precise loss function.
- Based on the above optimized methods, we propose an EDCNN based filter to efficiently eliminate artifacts

in compressed video, and improve the video subjective quality.

The rest of this paper is organized as follows. The related works are summarized in Section II. Section III describes the details of the proposed EDCNN based in-loop filtering algorithm. Experimental results are presented in Section IV. Finally, Section V concludes this paper.

II. Related Works

There are several in-loop filtering methods that have been proposed to eliminate artifacts. Yang *et al.* [13] proposed an SAO optimization method, in which the human visual characteristics are introduced into the SAO optimization process. For overcoming the limitations of these local image correlations based in-loop filtering methods, Ma *et al.* [14] utilized the nonlocal similarities to improve compression performance. In [15], Tsai *et al.* utilized a wiener-based adaptive filter to reduce artifacts. Zhang *et al.* [16] proposed a novel denoising method, in which the overlapped-block transform coefficients are estimated from the non-local blocks. Combined with the quantization noise model and block similarity prior model, the compression artifacts are well reduced. Zhang *et al.* [17] proposed a novel transform-domain adaptive in-loop filtering method based on the fusion of transform coefficients and non-local transform coefficients in similar blocks. Zhang *et al.* [18] utilized the low-rank constraint based image nonlocal prior knowledge to reduce artifacts. However, the entire structure of these in-loop filtering is not enormously changed by these methods, the achieved filtering performance is limited. Thus, new loop filtering methods should be considered to efficiently eliminate compression artifacts.

Moreover, many video quality enhancement works based on CNN for HEVC have been proposed. These works can efficiently reduce artifacts, and obtain a higher video quality performance. Park and Kim [19] proposed an in-loop filtering CNN (IFCNN), which is an extension of SRCNN. In IFCNN, the predicted residuals between the input and original images are used as the output. Dai *et al.* [20] proposed a VRCNN, in which the variable filter sizes are used for adapting to the variable block sizes in HEVC. Yang *et al.* [21] proposed a novel quality enhancement CNN (QECNN), which consists of a QECNN-I model for I pictures, and a QECNN-P model for P pictures. By considering the inter coding information, the QECNN-P model can efficiently improve the quality of P pictures. In [22], a temporal CNN architecture was proposed to reduce artifacts, in which the temporal correlation of consecutive pictures is used. Zhang *et al.* [23] proposed a residual highway CNN (RHCNN), in which the highway unit is used to protect the feature information. Moreover, the shortcut in RHCNN can be used to reconstruct the image with more detailed information, and the vanishing gradient problem [24] is well solved. However, these networks have simple architecture and small amount of network parameters, the reconstruction information cannot be well learned in the image mapping process. In addition, since the noise in the training process is not well eliminated by these networks, the reconstruction performance will be greatly influenced.

III. Proposed EDCNN Based In-Loop Filtering Algorithm

In this section, we will describe the network optimization methods in our proposed network in details. Firstly, the limitations of existing methods are analyzed. Then for addressing these problems, the proposed solutions are put forward, including a weighted normalization method, a feature information fusion block, and a precise loss function. Finally, the overall network architecture is given.

A. The Normalization Method

Generally, with the increased layers of network, the learning results will be better. However, a deep network having a large number of layers and complicated parameter updating will be difficult to train. The reason is that the new input distribution needs to be constantly adapt to the high-level layer, and this phenomenon is called internal covariate shift [25]. To address this problem, the batch normalization is proposed to flexibly reparameterize the input, which can be formulated as

$$y_i = BN(x_i), \tag{1}$$

where x_i, y_i are the input and output, respectively; BN represents the batch normalization, and it consists of four steps. First, the mean of mini-batch, μ, is calculated as

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i, \tag{2}$$

where N is the number of values over a mini-batch. Then, the variance of mini-batch, σ^2, is calculated, which is formulated as

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2. \tag{3}$$

Next, based on the above mean and variance, the input, x_i, is normalized by

$$\widehat{x_i} = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}}, \tag{4}$$

where ϵ is the value to prevent the divisor from being zero. Finally, the normalized value, y_i, is scaled and shifted to suit for output distribution, it is defined as

$$y_i = \gamma\,\widehat{x_i} + \beta, \tag{5}$$

where γ and β are parameters to be learned.

Moreover, the batch normalization will accelerate the training, and help to achieve a better performance. A great number of image enhancement networks have used the batch normalization, such as SRResNet [26], DnCNN [11], and so on. However, the batch normalization adds the noise which is not conducive to reconstructed image to the gradients. To address this problem, the weight normalization [27] is introduced to replace the batch normalization. While, the batch normalization and the weight normalization are both reparameterization methods, they are different from each other in that the object of batch normalization is the input, and that of the weight normalization is the weight vector. By using the

TABLE I
THE DATA SET OF NORMALIZATION METHODS

Sequence	Resolution	Sequence	Resolution
BQTerrace	1920×1080	BasketballDrill	832×480
BasketballDrive	1920×1080	BQMall	832×480
Cactus	1920×1080	BasketballPass	416×240
FourPeople	1280×720	BlowingBubbles	416×240
Johnny	1280×720	BQSquare	416×240

TABLE II
TEST CONDITIONS

Coding Structure	Low Delay-main (LD-main)
GOP Size	4
Intra Period	-1
QP Values	32
LoopFilterDisable	1
SAO	0
RateControl	0
Software Version	HM 16.9

weight normalization, the dependencies of mini-batch will not be introduced, and less noise will be introduced. The weight normalization is formulated as

$$\omega = \frac{g}{\|v\|}v, \tag{6}$$

where ω represents the weight vector; g represents a scalar parameter; v represents a parameter vector.

To analyze the performance of these two normalization methods, we implement them into the proposed network. The test conditions are that, (1) 10 standard video sequences in Table I are selected to be trained, and we will extract 50 frames in each sequence as the data set. After shuffling the data set, 490 frames are used for training, and the other frames are used for validation. It is noteworthy that the output images are uncompressed images, and the input images are compressed by HEVC. The encoding conditions are shown in Table II; (2) The input images are split into 64 × 64 patches, with the stride of 32. The experimental results are shown in Fig. 3. It can be seen that the network with weight normalization has a better performance than that with batch normalization, the weight normalization has smaller loss between the construction pictures and original pictures. Based on this observation, the weight normalization is adopted in the proposed network.

B. Proposed Feature Information Fusion Block

Each layer of neural network aims to extract the feature information, and the mapping relationships between the input pictures and output pictures can be established according to the learned information. Generally, a high-precise mapping relationship can generate a high-quality reconstruction result. Thus, it is essential that we should strengthen the extraction ability of feature information. Inspired by [28], we propose a feature information fusion block to enhance the learning ability of the neural network in this paper, and its structure is shown in Fig. 4.

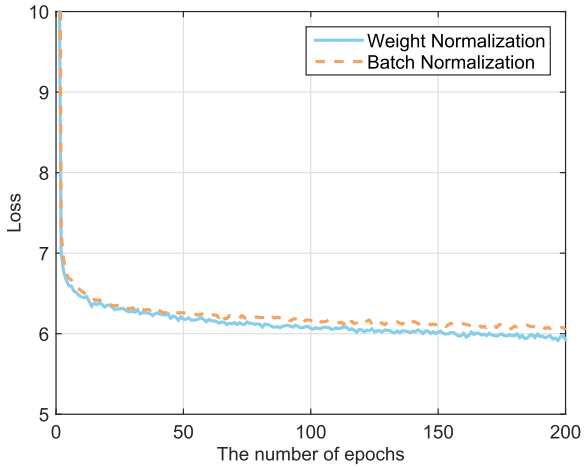


Fig. 3. The training loss comparison between the weight normalization and the batch normalization.

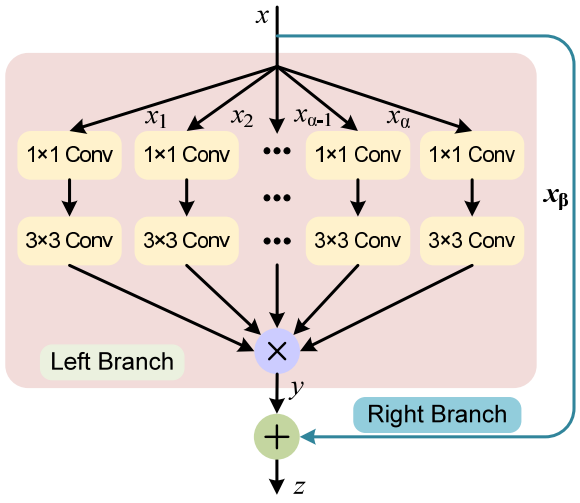


Fig. 4. The structure of proposed feature information fusion block.

As shown in Fig. 4, the proposed feature information fusion block consists of two branches, the right branch and the left branch, respectively. For the left branch, it has the size of α sub-branches, and each sub-branch consists of one 1×1 convolution layer and one 3×3 convolution layer. For the 1×1 convolution layer, it is used to change the dimensionality of input features. And, the 3×3 convolution layer is helpful to enhance the learning ability of the neural networks. After the convolution layer, the rectified linear unit (ReLU) activation layer is added to introduce more nonlinearity. Moreover, the output of the left branch, which is obtained by concatenating the outputs of its sub-branches, is added up with the right branch, and it is a typical shortcut connection [29]. Finally, the proposed feature information fusion block can be summarized as

$$\begin{cases} x_i = x, \quad 1 \le i \le \alpha \text{ or } i = \beta, \\ y = \sigma(x_1) \bigotimes \sigma(x_2) \bigotimes \cdots \bigotimes \sigma(x_\alpha), \\ z = x_\beta + y, \end{cases} \tag{7}$$

where x_i is the input of the left and right branches of the proposed feature information fusion block, which is obtained by

TABLE III
THE PERFORMANCE OF DIFFERENT α SETTINGS

Sequence	Resolution	α=2 (dB)	α=4 (dB)	α=8 (dB)	α=16 (dB)	α=32 (dB)
Flowervase	416×240	39.76	39.79	39.77	39.75	37.94
Keiba	832×480	36.24	36.25	36.23	36.18	36.16
RaceHorses	832×480	34.13	34.15	34.12	34.08	34.07
Tennis	1920×1080	37.73	37.75	37.67	37.64	37.64
Average		36.97	36.99	36.95	36.91	36.90

duplicating x, and x is the input of the proposed fusion block; y indicates the output of the left branch; $\sigma(\cdot)$ represents the convolution operation; $\bigotimes$ means the concatenation operation; z is the output of the proposed fusion block.

In the proposed feature information fusion block, the left branch is used to enhance the learning ability of the network, and it has the size of α sub-branches. If the size of sub-branches is small, which will degrade the learning ability of the network. On the contrary, the large size of sub-branches will boost the learning ability, but the training complexity will be increased significantly. In order to achieve the optimal filtering performance, a group of α values, including 2, 4, 8, 16, and 32, are tested. The test conditions are same with that using in Section III-A, and four video sequences with various resolutions are tested. The experimental results are shown in Table III. We can see from Table III, that when α equals to 4, the network achieves the optimal PSNR performance, and the PNSR value reaches up to 36.99 dB. Hence, the left branch of the proposed feature information fusion block adopts 4 sub-branches, and α is set to 4 in this paper.

To enhance the learning ability of the proposed feature information fusion block, we use the concatenation operation in the proposed fusion block to achieve the fusion function. The operation of concatenating fuses the features extracted by convolution layer, the dimensionality will be expanded. Since the input dimensionality and output dimensionality of add operation are identical, the input dimensionality of left branch should be reduced. There are two ways to reduce the input dimensionality, the one jointly uses one 1×1 convolution layer and one 3×3 convolution layer to reduce the dimensionality of features, its structure is shown in Fig. 4, and the other directly uses the 3×3 convolution layer to reduce the dimensionality of features, whose structure is shown in Fig. 5.

To prove the efficiency of the adopted dimensionality reduction method, the performance of these two dimensionality reduction methods is compared, and the results are tabulated in Table IV. We can see that the proposed fusion block with 1×1 and 3×3 convolution layers achieves better performance than that only with 3×3 convolution layer, and it obtains 0.03 dB PSNR increase. The reason is that the 1×1 convolution layer makes the network deeper, and the ReLU introduces more nonlinearity to the network. Both of these two are helpful to improve the learning ability of proposed fusion block. Hence, the 1×1 and 3×3 convolution layers are jointly used in the proposed feature information fusion block.

In order to demonstrate the efficiency of the proposed feature information fusion block, the performance of the proposed network is compared with that without using the proposed fusion block. The analyses results are shown in Table V. In the table, NF indicates the network contains the proposed fusion block, and NWF represents the network does not use the proposed fusion block. We can observe from Table V that the network using the proposed fusion block achieves better performance than that without using the proposed fusion block, and the network using the proposed fusion block obtains an average of 0.13 dB PSNR increase. These results turn out that the proposed fusion block works efficiently for the filtering task.

To sum up, by applying the proposed fusion block, more feature information can be kept in nonlinearity, and the feature information which comes from the low-level layer can efficiently transmit into the high-level layer. As a result, better reconstruction results can be achieved. Furthermore, due to the shortcut connection used in the network, the residual between the compressed image and original image is easier to be

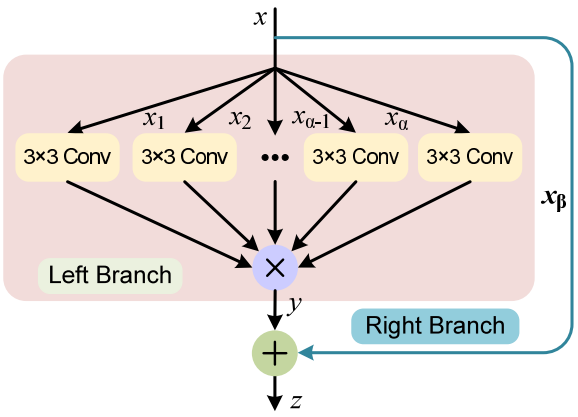


Fig. 5. The feature information fusion block with only 3×3 convolution layer.

TABLE IV
THE PSNR PERFORMANCE OF DIFFERENT DIMENSION REDUCTION METHODS

Sequence	Resolution	1×1&3×3 (dB)	3×3 (dB)
Flowervase	416×240	39.79	39.76
Keiba	832×480	36.25	36.24
RaceHorses	832×480	34.15	34.12
Tennis	1920×1080	37.75	37.71
Average		36.99	36.96

TABLE V
THE PERFORMANCE OF THE NETWORK WITH PROPOSED FUSION BLOCK AND THAT WITHOUT FUSION BLOCK

Sequence	Resolution	NF (dB)	NWF (dB)
Flowervase	416×240	39.79	39.58
Keiba	832×480	36.25	36.13
RaceHorses	832×480	34.15	34.11
Tennis	1920×1080	37.75	37.63
Average		36.99	36.86

learned, the vanishing gradient problem is well addressed, and the training is accelerated.

C. Proposed Adaptive Loss Function

For constantly training the mapping relationship between the reconstructed picture and its ground truth, the loss function is used to reflect the deviation between the input and output, and we can obtain a precise pixel prediction by minimizing the loss.

In recent CNN based works for HEVC [21], [23], [30], the mean square error (MSE) is most used, and it is calculated by

$$\mathcal{L}_{MSE}(\Theta) = \frac{1}{N}\sum_{n=1}^{N} \|\mathcal{F}(X_n;\Theta) - Y_n\|_2^2, \tag{8}$$

where Θ is network parameter; X_n is the compressed picture; Y_n is its ground truth. However, the MSE will over penalize the errors by the square, and it has been proved that the MSE cannot capture the intricate characteristics of the HVS [31], [32]. Unlike the MSE, the mean absolute error (MAE) will not over penalize larger errors, thus it is propitious to end-to-end learning. The MAE loss function is defined as

$$\mathcal{L}_{MAE}(\Theta) = \frac{1}{N}\sum_{n=1}^{N} \|\mathcal{F}(X_n;\Theta) - Y_n\|_1. \tag{9}$$

By using the MAE based loss function, the network is easier to obtain the precise results due to the MAE is not sensitive to the outlier. However, the MAE is hard to descend. On the contrary, the MSE loss function is sensitive to errors, and it can easily achieve a local minimum. Based on these characteristics, we propose a mixed loss function in this paper, and it is defined as

$$\mathcal{L}_{MIX} = \delta\mathcal{L}_{MSE} + (1-\delta)\mathcal{L}_{MAE}, \tag{10}$$

where δ is an adaptive parameter according to loss convergence, it is defined as

$$\delta = \begin{cases} 1 & if \quad \frac{1}{N}\sum_{i=1}^{N} |\mathcal{L}_{c-i+1} - \mathcal{L}_{c-i}| < \xi, \\ 0 & otherwise, \end{cases} \tag{11}$$

where N is the number of continuous epoch, and it equals to 3; c represents the number of current epoch; $\mathcal{L}$ is the loss value; ξ is the threshold, which is used to control the performance of the loss function.

To come up with the optimal ξ, a group of ξ values from 0.009 to 0.018 are tested with four video sequences (BasketballPass, BlowingBubbles, BQSquare, and RaceHorses), the average PSNR of each ξ is shown in Fig. 6. It can be seen that when ξ equals to 0.015, the proposed loss function achieves the best performance. Hence, the threshold ξ for the proposed loss function is set to 0.015 in this paper.

To prove the efficiency of the proposed loss function, the objective and subjective qualities are compared among these three loss functions (MSE, MAE, and proposed loss function), the experimental results are presented in Figs. 7 and 8. From Fig. 7, it can be obviously seen that the validation PSNR of the proposed loss function outperforms the loss functions MSE and MAE in general. Meanwhile, we can see from Fig. 8 that the zoomed regions of proposed loss function has the best performances with less artifacts. Based on these two observations, we can see that the proposed loss function efficiently improves the network's performance.

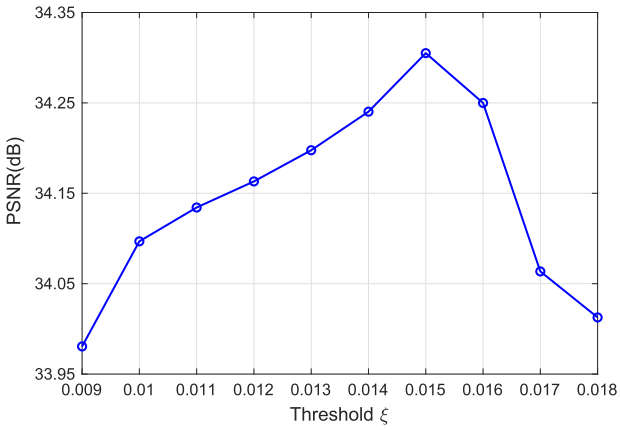


Fig. 6. The performance of proposed loss function with different threshold ξ.

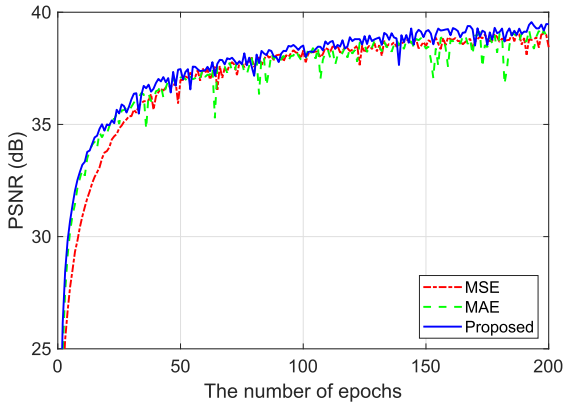


Fig. 7. Performance comparison among different loss functions.

D. Proposed EDCNN Architecture

Based on above analyses, the proposed in-loop filtering algorithm is modeled, and the network architecture is shown in Fig. 9. As shown in Fig. 9, the proposed EDCNN network consists of 7 blocks, each block contains 4 convolution and ReLU layers, in which each convolution layer has an operation of weight normalization. The blocks of convolution layers are used to keep the identical channel between the low-level layer and high-level layer. The overall proposed network has 16 layers, and the network parameters are listed in Table VI. Moreover, the input channel of the block is firstly duplicated in 4 equal parts, after the convolutional operating, the channels of 4 parts will be added in quantity, and then the addition of 4 parts will be added with the input feature maps in values. When the feature fusion is completed, it will be processed by a convolution layer for channel transformation. With the exception of shortcut connections in blocks, a long shortcut connection from the original input without processing by

Fig. 8. The subjective image quality comparison. (The 1st frame of PartyScene encoded by low-delay coding structure with QP32). (a) Ground truth. (b) MSE (PSNR: 34.92 dB). (c) MAE (PSNR: 35.41 dB). (d) Proposed (PSNR: 35.53 dB).

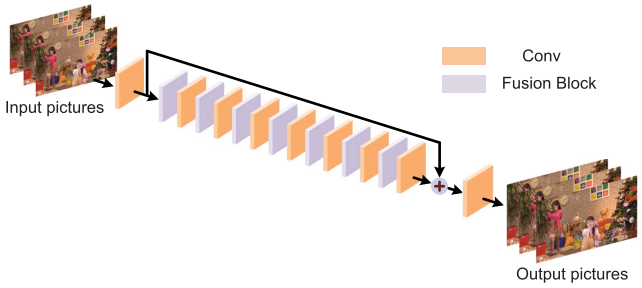


Fig. 9. The architecture of proposed EDCNN.

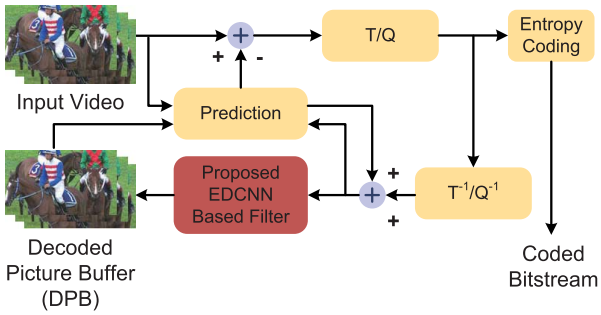


Fig. 10. The proposed EDCNN based in-loop filtering in HEVC.

convolution layer to the final output is established for obtaining the further precise mapping relationship.

Finally, in order to improve the encoding performance of HEVC, the proposed in-loop filtering algorithm is embedded into the HEVC reference software, as shown in Fig. 10. The filter in HEVC is replaced by the proposed EDCNN based in-loop filtering.

TABLE VI
THE DETAILED NETWORK PARAMETERS

Layer_name	Kernel_size	Input_channel	Output_channel
Conv1	3×3	3	64
Block1	3×3	64	64
Conv2	3×3	64	128
Block2	3×3	128	128
Conv3	3×3	128	256
Block3	3×3	256	256
Conv4	3×3	256	512
Block4	3×3	512	512
Conv5	3×3	512	256
Block5	3×3	256	256
Conv6	3×3	256	128
Block6	3×3	128	128
Conv7	3×3	128	64
Block7	3×3	64	64
Conv8	3×3	64	64
Conv9	3×3	64	3

IV. EXPERIMENTAL RESULTS

In this section, we will present the performance of the proposed EDCNN based in-loop filtering algorithm in detail, including experimental settings, comparisons on BDBR and BDPSNR, comparisons on objective visual performance, comparisons on rate distortion curves, comparisons on subjective visual quality, and comparisons on video quality smoothness.

A. Experimental Settings

1) Data Preparation: Twenty-four HEVC standard video sequences [33] in Table VII with various resolutions are adopted, and these twenty-four video sequences are divided into two groups: twenty video sequences are used for training, and four videos are used for testing. For obtaining the mapping relationships, the video sequences encoded by HEVC are used as the input of networks, and the uncompressed raw video sequences are used as the output of networks. The encoding platform is HEVC reference software HM16.9 [34]. Considering that different QPs in HEVC have different compression results with varying degrees of artifacts, four different QPs (22, 27, 32 and 37) are set to ensure the results of the experiments more representative. Besides, the other encoding conditions are set as follows, (1) the low-delay main (LD-main) and random-access main (RA-main) coding structure are used; (2) for the benchmark HM 16.9, it uses the default settings, and the DF/SAO is turned on. On the contrary, the proposed algorithm and the compared algorithms (Ledig *et al.* [26], Zhang *et al.* [23]) are implemented in the HM16.9 with DF/SAO off. Each sequence is encoded by HM16.9, and 1 to 50 frames are extracted into the data set, 3800 of which are used for training, and the other frames are used for validating. Generally, the training sequences and testing sequences should be kept different, however, since twenty video sequences for training are not enough, 100 to 150 frames of twenty training video sequences, and 1 to 100 frames of four testing video sequences, are used to verify the performance of networks,

TABLE VII
THE DATA SET

Data set	Sequence	Resolution	Sequence	Resolution
Training Set	BasketballPass	416×240	Vidyo1	1280×720
	BlowingBubbles	416×240	Vidyo3	1280×720
	BQSquare	416×240	Vidyo4	1280×720
	RaceHorses	416×240	BasketballDrive	1920×1080
	BasketballDrill	832×480	BQTerrace	1920×1080
	BQMall	832×480	Cactus	1920×1080
	PartyScene	832×480	Kimono1	1920×1080
	FourPeople	1280×720	ParkScene	1920×1080
	Johnny	1280×720	PeopleOnStreet	2560×1600
	KristenAndSara	1280×720	Traffic_crop	2560×1600
Testing Set	Flowervase	416×240	RaceHorses	832×480
	Keiba	832×480	Tennis	1920×1080

in which fifty frames of each training video are used to simulate sequences with similar contents.

2) Training Settings: For expanding dataset, each input picture and its corresponding ground truth are split into 64 × 64 patches, with a stride of 32. In our experiments, the learning rate is set to 0.0001, and the adaptive moment estimation (Adam) optimization method is used. Before starting training, the training data are shuffled. The training platform uses the ubuntu 16.04 operating system with Intel i7-6900K CPU, 64 GB RAM, and NVIDIA 1080Ti GPUs.

B. Comparisons on BDBR and BDPSNR

To demonstrate the efficiency of the proposed filter, the Bjontegaard delta bitrate (BDBR) and Bjontegaard delta peak signal-to-noise rate (BDPSNR) [35] are used to evaluate the performance of reconstructed results. The BDBR indicates the bitrate saving of two algorithms under the equivalent PSNR value, while the BDPSNR means the PSNR difference between two algorithms at the same bitrate. And the proposed EDCNN based filter is compared with the SRResNet in Ledig *et al.* [26], and the RHCNN in Zhang *et al.* [23]. The HEVC reference software HM 16.9 is used as the benchmark, which uses the default settings, and the DF/SAO is turned on. The proposed algorithm, Ledig's algorithm [26], and Zhang's algorithm [23] are also implemented in HM 16.9, but the DF/SAO is turned off. In order to make the SRResNet in Ledig's algorithm handle the filtering task, the stride of convolutional layer is set to 1, and the padding mode is set to the same mode. In Zhang *et al.* [23], the channels between the input of highway unit and the output of concatenation are not identical, this problem is addressed by doubling the channels of the input of each highway unit in our paper. In addition, the network of Zhang *et al.* [23] only processes the luminance components filtering task, in order to make the network handle the YUV components filtering task, a 1×1 convolution layer is added after the last convolution layer. The experimental results are compared and summarized in Tables VIII and IX.

It can be seen from Table VIII that when using the low-delay coding structure, the BDBR of Ledig's algorithm [26] is from −7.76% to7.78%, 0.92% on average, and its BDPSNR is from −0.266 dB to 0.342 dB, −0.003 dB on average. For Zhang *et al.* [23], the BDBR is from −8.81% to 7.87%, 0.02% on average, and its BDPSNR increases from −0.204 dB to 0.397 dB, 0.044 dB on average. For the proposed algorithm, the BDBR decreases from −12.06% to −1.77%, −6.27% on average, and its BDPSNR increases from 0.046 dB to 0.547 dB, 0.239 dB on average. From these values, we can observe that under the low-delay coding structure, the Ledig's and Zhang's algorithms only work efficiently for video sequences with similar content with the training data, which reflects that these two algorithms have bad generalization capacity. On the contrary, the proposed algorithm achieves the best RD performance, and efficiently improves the RD performance for videos with various resolutions and contents.

Table IX presents the encoding results of random-access coding structure, we can see that the BDBR of Ledig's algorithm [26] is from −1.94% to 5.86%, 1.33% on average, and its BDPSNR is from −0.221 dB to 0.146 dB, −0.022 dB on average. These values reflect that the Ledig's network can not efficiently address the video quality enhancement task. For Zhang's algorithm [23], its BDBR is from −10.68% to 1.07%, −3.64 on average, and its BDPSNR is from −0.013 dB to 0.317 dB, 0.146 dB on average. For our proposed algorithm, the BDBR decreases from −12.31% to −0.41%, −6.62% on average, and its BDPSNR increases from 0.032 dB to 0.427 dB, 0.237 dB on average. From these values, it can be seen that the proposed algorithm efficiently handles the filtering task, and achieves the best RD performance. In addition, we can observe that the performance of Zhang's algorithm [23] in our paper are different from that in its original paper, the main reason is that individual models are trained for different QPs in [23], while different QPs share the same model in our paper. Moreover, one significant contribution of our paper is that the proposed model uses one model to perform filtering for different QPs in encoding process, which means the proposed model could handle the filtering task for video with different distortions. Hence, compared with the other CNN-based filtering algorithms that use different models for different QPs, such as Zhang *et al.* [23], the proposed model has better practicality in video coding.

C. Comparisons on Objective Visual Quality

To intuitively show the achievements of the proposed algorithm, we also perform comparisons on average PSNR

TABLE VIII
RD PERFORMANCE COMPARISON FOR LOW-DELAY CODING STRUCTURE

Sequence	Resolution	Ledig [26] vs. HM16.9		Zhang [23] vs. HM 16.9		Proposed vs. HM 16.9	
		BDBR (%)	BDPSNR (dB)	BDBR (%)	BDPSNR (dB)	BDBR (%)	BDPSNR (dB)
PeopleOnStreet	2560×1600	-0.76	0.045	-0.91	0.070	-6.82	0.334
Traffic		-1.65	0.067	1.89	-0.024	-5.02	0.166
BasketballDrive	1920×1080	1.30	-0.023	-0.28	0.015	-4.31	0.112
BQTerrace		-7.76	0.342	-8.81	0.397	-13.08	0.547
Cactus		2.07	-0.034	3.89	-0.053	-3.38	0.089
Kimono1		3.47	-0.111	2.21	-0.059	-1.77	0.064
ParkScene		-1.56	0.053	0.85	-0.011	-2.77	0.090
BasketballDrill	832×480	-2.97	0.122	-7.64	0.315	-12.06	0.525
BQMall		-0.17	0.017	-1.78	0.100	-6.03	0.246
PartyScene		-1.86	0.096	-3.86	0.207	-9.10	0.417
BasketballPass	416×240	-0.25	0.014	-3.27	0.164	-6.05	0.294
BlowingBubbles		-4.30	0.167	-3.19	0.138	-8.10	0.315
BQSquare		-7.76	0.342	-8.81	0.397	-13.08	0.547
RaceHorses		-1.15	0.056	-1.29	0.066	-3.27	0.150
FourPeople	1280×720	2.80	-0.070	-1.96	0.101	-8.85	0.320
Johnny		3.94	-0.062	5.90	-0.183	-9.83	0.249
KristenAndSara		4.38	-0.091	5.06	-0.140	-8.35	0.257
vidyo1		5.35	-0.131	-0.06	0.023	-7.56	0.241
vidyo3		4.11	-0.092	-1.14	0.076	-8.56	0.293
vidyo4		1.89	-0.035	6.98	-0.104	-4.86	0.151
Flowervase	416×240	7.78	-0.266	5.83	-0.163	-2.62	0.139
Keiba	832×480	4.77	-0.160	7.87	-0.204	-1.02	0.046
RaceHorses	832×480	3.48	-0.130	2.01	-0.055	-2.07	0.088
Tennis	1920×1080	7.01	-0.201	0.87	-0.015	-1.82	0.058
Average		0.92	-0.003	0.02	0.044	-6.27	0.239

TABLE IX
RD PERFORMANCE COMPARISON FOR RANDOM-ACCESS CODING STRUCTURE

Sequence	Resolution	Ledig [26] vs. HM 16.9		Zhang [23] vs. HM16.9		Proposed vs. HM 16.9	
		BDBR (%)	BDPSNR (dB)	BDBR (%)	BDPSNR (dB)	BDBR (%)	BDPSNR (dB)
PeopleOnStreet	2560×1600	-3.00	0.146	-6.82	0.317	-9.17	0.427
Traffic		-1.13	0.058	-2.92	0.106	-5.97	0.197
BasketballDrive	1920×1080	-0.98	0.024	-2.86	0.076	-5.99	0.145
BQTerrace		-0.30	0.024	-0.17	0.013	-5.08	0.074
Cactus		1.35	-0.016	-1.68	0.047	-5.29	0.119
Kimono1		0.79	-0.021	-1.89	0.064	-3.68	0.120
ParkScene		3.86	-0.088	-1.80	0.063	-3.85	0.122
BasketballDrill	832×480	1.04	-0.027	-6.91	0.294	-10.65	0.470
BQMall		-0.12	0.024	-4.46	0.182	-7.37	0.288
PartyScene		-1.36	0.075	-6.92	0.310	-9.28	0.419
BasketballPass	416×240	2.40	-0.082	-4.57	0.216	-6.34	0.300
BlowingBubbles		-0.45	0.041	-5.01	0.208	-7.54	0.305
BQSquare		-1.94	0.141	-8.32	0.298	-10.99	0.428
RaceHorses		2.02	-0.057	-2.16	0.096	-3.52	0.152
FourPeople	1280×720	-1.07	0.036	-6.87	0.257	-10.53	0.379
Johnny		4.17	-0.069	-2.20	0.082	-11.22	0.285
KristenAndSara		1.32	-0.016	-2.08	0.090	-9.00	0.265
vidyo1		1.22	-0.009	-4.79	0.176	-9.22	0.298
vidyo3		-0.07	0.018	-10.68	0.335	-12.31	0.415
vidyo4		3.81	-0.074	-1.58	0.069	-5.13	0.157
Flowervase	416×240	5.86	-0.221	-0.44	0.081	-0.41	0.070
Keiba	832×480	5.69	-0.170	1.07	-0.013	-0.53	0.032
RaceHorses	832×480	3.23	-0.104	-1.11	0.053	-2.59	0.105
Tennis	1920×1080	5.63	-0.161	-2.27	0.075	-3.28	0.105
Average		1.33	-0.022	-3.64	0.146	-6.62	0.237

TABLE X
THE PSNR STANDARD DEVIATIONS OF LOW-DELAY CODING STRUCTURE (UNIT: dB)

Sequence	Resolution	HM 16.9				Ledig [26]				Zhang [23]				Proposed			
		QP22	QP27	QP32	QP37	QP22	QP27	QP32	QP37	QP22	QP27	QP32	QP37	QP22	QP27	QP32	QP37
PeopleOnStreet	2560×1600	0.94	0.96	1.09	1.11	0.74	0.91	1.12	1.17	0.55	0.78	1.05	1.17	0.79	0.90	1.10	1.19
Traffic		0.57	0.51	0.44	0.36	0.47	0.49	0.45	0.38	0.43	0.47	0.45	0.39	0.52	0.51	0.46	0.39
BasketballDrive	1920×1080	0.57	0.53	0.71	0.77	0.50	0.50	0.70	0.78	0.44	0.46	0.68	0.79	0.50	0.48	0.70	0.80
BQTerrace		1.45	0.58	0.45	0.40	1.13	0.54	0.43	0.39	0.94	0.48	0.41	0.38	1.26	0.54	0.44	0.39
Cactus		0.63	0.42	0.50	0.49	0.55	0.41	0.51	0.51	0.44	0.38	0.50	0.51	0.53	0.40	0.51	0.52
Kimono1		0.63	0.89	1.12	1.15	0.62	0.84	1.07	1.10	0.68	0.90	1.13	1.18	0.63	0.89	1.14	1.20
ParkScene		0.57	0.56	0.50	0.40	0.52	0.55	0.52	0.41	0.47	0.53	0.52	0.42	0.54	0.57	0.53	0.42
BasketballDrill	832×480	0.39	0.49	0.59	0.61	0.33	0.47	0.61	0.63	0.30	0.44	0.61	0.66	0.35	0.48	0.63	0.68
BQMall		0.66	0.68	0.68	0.58	0.55	0.62	0.67	0.60	0.47	0.58	0.67	0.61	0.56	0.63	0.68	0.62
PartyScene		0.94	0.84	0.72	0.53	0.66	0.74	0.71	0.54	0.63	0.71	0.71	0.57	0.79	0.79	0.74	0.57
BasketballPass	416×240	0.73	0.90	0.97	0.89	0.57	0.83	0.96	0.90	0.53	0.80	0.97	0.92	0.61	0.86	0.99	0.93
BlowingBubbles		0.87	0.56	0.37	0.26	0.71	0.52	0.38	0.27	0.68	0.51	0.38	0.27	0.78	0.55	0.39	0.28
BQSquare		0.95	0.91	0.69	0.47	0.58	0.74	0.65	0.47	0.57	0.69	0.63	0.51	0.78	0.81	0.66	0.52
RaceHorses		1.22	1.25	1.15	0.90	1.11	1.21	1.15	0.92	1.05	1.19	1.14	0.91	1.17	1.24	1.15	0.92
FourPeople	1280×720	0.42	0.35	0.37	0.32	0.37	0.35	0.37	0.34	0.28	0.33	0.38	0.36	0.37	0.37	0.40	0.37
Johnny		0.35	0.21	0.17	0.16	0.29	0.19	0.18	0.16	0.26	0.20	0.19	0.17	0.30	0.20	0.19	0.17
KristenAndSara		0.38	0.35	0.36	0.35	0.31	0.32	0.36	0.35	0.29	0.32	0.38	0.38	0.32	0.34	0.39	0.39
vidyo1		0.35	0.33	0.31	0.26	0.27	0.29	0.29	0.26	0.25	0.29	0.31	0.28	0.30	0.33	0.33	0.29
vidyo3		0.42	0.38	0.32	0.26	0.34	0.35	0.31	0.27	0.29	0.35	0.33	0.28	0.37	0.39	0.35	0.29
vidyo4		0.57	0.40	0.29	0.23	0.49	0.37	0.28	0.23	0.42	0.35	0.28	0.24	0.51	0.39	0.29	0.24
Flowervase	416×240	1.19	1.22	1.20	1.25	0.81	1.01	1.11	1.20	1.01	1.12	1.17	1.24	1.06	1.16	1.19	1.26
Keiba	832×480	1.03	1.25	1.34	1.28	0.94	1.18	1.30	1.25	0.94	1.18	1.30	1.27	0.96	1.21	1.33	1.28
RaceHorses	832×480	1.17	1.38	1.20	0.88	0.97	1.30	1.20	0.90	0.87	1.26	1.19	0.90	0.99	1.34	1.22	0.91
Tennis	1920×1080	0.43	0.62	0.79	0.83	0.40	0.61	0.81	0.86	0.35	0.57	0.78	0.85	0.38	0.59	0.79	0.86
Average		0.73	0.69	0.68	0.61	0.59	0.64	0.67	0.62	0.55	0.62	0.67	0.64	0.64	0.67	0.69	0.65

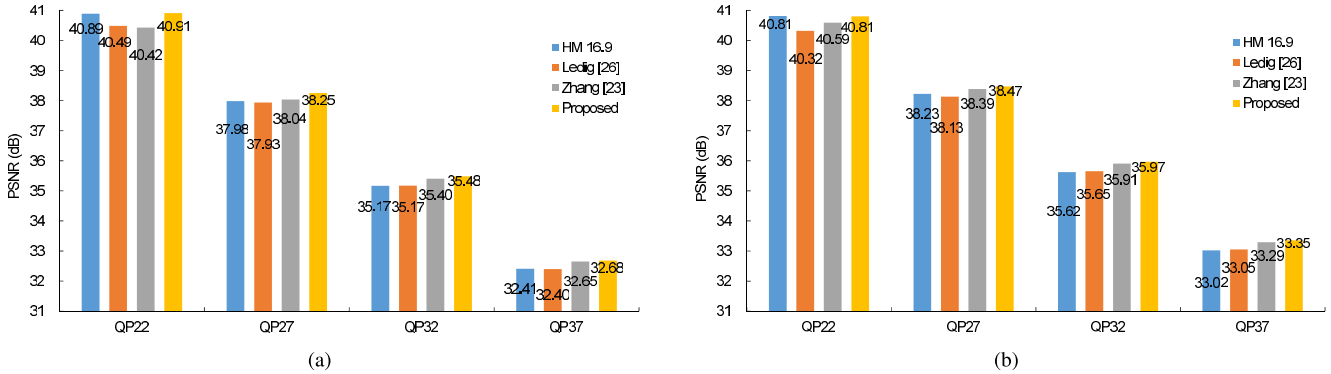
TABLE XI
THE PSNR STANDARD DEVIATIONS OF RANDOM-ACCESS CODING STRUCTURE (UNIT: dB)

Sequence	Resolution	HM 16.9				Ledig [26]				Zhang [23]				Proposed			
		QP22	QP27	QP32	QP37	QP22	QP27	QP32	QP37	QP22	QP27	QP32	QP37	QP22	QP27	QP32	QP37
PeopleOnStreet	2560×1600	1.46	1.20	1.14	1.10	1.12	1.13	1.16	1.13	1.03	1.06	1.13	1.16	1.15	1.09	1.14	1.18
Traffic		0.70	0.47	0.39	0.32	0.59	0.46	0.40	0.34	0.57	0.45	0.40	0.36	0.62	0.46	0.41	0.36
BasketballDrive	1920×1080	0.81	0.66	0.77	0.81	0.68	0.63	0.77	0.81	0.65	0.59	0.75	0.82	0.68	0.59	0.76	0.83
BQTerrace		1.83	0.63	0.37	0.34	1.20	0.56	0.36	0.34	1.33	0.51	0.32	0.31	1.50	0.53	0.33	0.32
Cactus		0.85	0.43	0.41	0.39	0.69	0.42	0.41	0.39	0.65	0.39	0.41	0.40	0.69	0.39	0.41	0.41
Kimono1		0.72	0.92	1.03	0.98	0.75	0.92	1.00	0.95	0.69	0.92	1.04	1.00	0.69	0.93	1.05	1.02
ParkScene		0.63	0.48	0.42	0.35	0.54	0.47	0.43	0.36	0.54	0.47	0.43	0.37	0.58	0.48	0.44	0.38
BasketballDrill	832×480	0.68	0.67	0.65	0.60	0.53	0.62	0.65	0.61	0.53	0.63	0.66	0.63	0.58	0.66	0.68	0.66
BQMall		0.76	0.63	0.57	0.49	0.60	0.57	0.55	0.48	0.57	0.55	0.55	0.50	0.62	0.57	0.57	0.51
PartyScene		1.14	0.97	0.81	0.57	0.80	0.84	0.77	0.58	0.79	0.81	0.78	0.61	0.94	0.86	0.80	0.62
BasketballPass	416×240	1.06	1.03	0.97	0.84	0.78	0.93	0.94	0.84	0.80	0.94	0.96	0.88	0.87	0.96	0.96	0.88
BlowingBubbles		0.82	0.59	0.48	0.39	0.65	0.53	0.46	0.39	0.63	0.51	0.46	0.41	0.72	0.54	0.47	0.41
BQSquare		1.43	0.92	0.59	0.43	0.86	0.69	0.50	0.42	0.90	0.62	0.46	0.43	1.03	0.64	0.46	0.43
RaceHorses		1.46	1.20	1.04	0.84	1.24	1.13	1.01	0.86	1.28	1.12	1.01	0.87	1.34	1.13	1.01	0.87
FourPeople	1280×720	0.49	0.36	0.34	0.33	0.44	0.36	0.36	0.35	0.35	0.33	0.36	0.36	0.40	0.33	0.36	0.36
Johnny		0.42	0.27	0.26	0.25	0.35	0.25	0.25	0.25	0.30	0.24	0.26	0.25	0.33	0.25	0.27	0.27
KristenAndSara		0.48	0.35	0.33	0.30	0.40	0.33	0.32	0.30	0.35	0.31	0.32	0.31	0.38	0.32	0.32	0.31
vidyo1		0.44	0.36	0.34	0.32	0.37	0.32	0.32	0.32	0.32	0.31	0.33	0.34	0.36	0.32	0.34	0.34
vidyo3		0.53	0.33	0.29	0.27	0.43	0.30	0.27	0.27	0.39	0.28	0.28	0.29	0.44	0.30	0.27	0.28
vidyo4		0.68	0.45	0.35	0.31	0.57	0.42	0.34	0.32	0.55	0.41	0.35	0.33	0.59	0.42	0.35	0.33
Flowervase	416×240	1.03	0.94	0.87	0.85	0.95	0.90	0.87	0.85	0.95	0.93	0.89	0.85	0.92	0.89	0.87	0.85
Keiba	832×480	1.41	1.43	1.44	1.35	1.19	1.35	1.41	1.33	1.14	1.31	1.39	1.33	1.14	1.29	1.38	1.34
RaceHorses	832×480	2.01	1.55	1.18	0.84	1.61	1.44	1.17	0.86	1.60	1.44	1.17	0.87	1.65	1.44	1.17	0.87
Tennis	1920×1080	0.67	0.79	0.94	0.97	0.59	0.78	0.95	1.00	0.55	0.74	0.93	1.00	0.56	0.74	0.93	1.01
Average		0.94	0.74	0.67	0.59	0.75	0.68	0.65	0.60	0.73	0.66	0.65	0.61	0.78	0.67	0.66	0.62

for each algorithm with different QPs, and the results are presented in Fig. 11. We can see that for the videos with less distortions (i.e., QP equals to 22), the Ledig's and Zhang's networks can not well improve the video quality. While the proposed algorithm can efficiently improve the video quality in different-level distortions (i.e., QP equals to 22, 27, 32, and 37.). In addition, our proposed EDCNN based filtering algorithm obtains significant PSNR increase than the Ledig's and Zhang's algorithms. Overall, we can make a conclusion that our proposed filtering algorithm efficiently improves the video objective quality for HEVC.

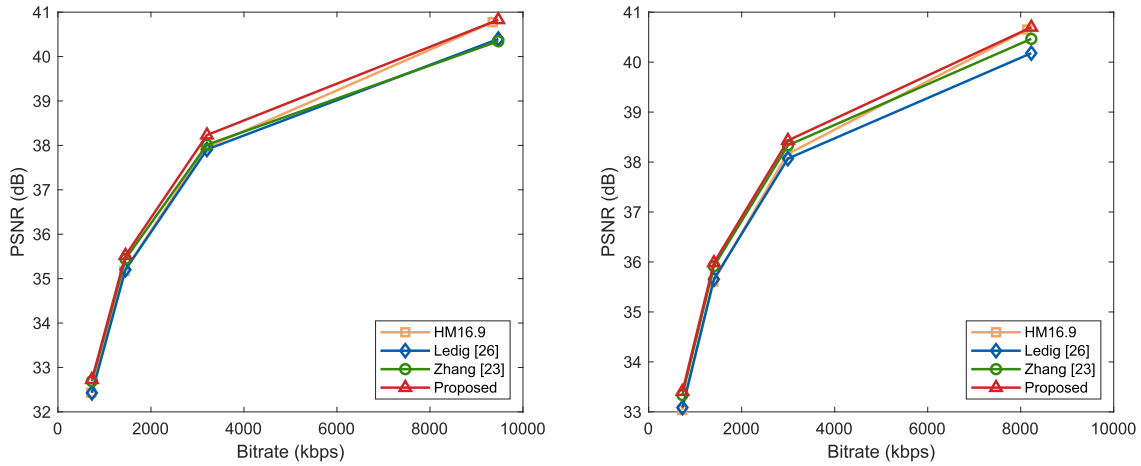
D. Comparisons on Rate Distortion Curves

To show the RD performance of each algorithm, the RD curves of each algorithm are given in Fig. 12. The values of bitrate and PSNR in Fig. 12 denote the total average encoding results of these 24 test video sequences. We can see that the proposed filtering algorithm achieves the best RD performance



(a) (b)

Fig. 11. The PSNR performance comparison for different QPs. (a) Low-delay coding structure. (b) Random-access coding structure.



(a) (b)

Fig. 12. RD curvs comparison. (a) Low-delay coding structure. (b) Random-access coding structure.

in both low-delay and random-access coding structures. These results prove that the proposed algorithm efficiently optimizes the RD performance of HEVC.

E. Comparisons on Subjective Visual Quality

In order to show the video subjective quality of different filtering algorithms, we select five video sequences to perform comparisons, including "BasketballDrive", "Cactus", "Johnny", "BQMall", and "Keiba". The results of these five videos are presented in Fig. 13. The left side is the ground truth of these five video sequences, and the image crops are zoomed from these five videos. From left to right are the subjective results for these four filtering algorithms, HM16.9, Ledig *et al.* [26], Zhang *et al.* [23] and the proposed algorithm, respectively. In Fig. 13, we can see that the compared areas in HM16.9 have obvious artifacts, including ringing artifacts and color excursion. The other three algorithms can reduce most of artifacts, however, some obvious artifacts are still in Ledig *et al.* [26], especially for the blocking artifacts. The artifacts are much reduced by Zhang *et al.* [23], but, the compared areas become more blurring, and lots of details in image crops are eliminated. Compared with these algorithms, the proposed algorithm achieves a remarkable subjective quality with few artifacts and more details. In the end, it can be concluded that the visual performance of the proposed algorithm is superior to the other three algorithms.

F. Comparisons on Video Quality Smoothness

To further verify the performance of the proposed filtering algorithm, the video quality smoothness is used to test the stabilities of video frames. The video quality smoothness significantly affects the global subjective video quality, thus the objective PSNR standard deviation is used as the criterion. The experimental results of the PSNR standard deviation for each algorithm are shown in Tables X and XI. From Table X, we can see that the average PSNR quality fluctuation ranges of low-delay coding structure are (0.61, 0.73), (0.59, 0.62), (0.55, 0.64), and (0.64, 0.69) for the filter in HM16.9, Ledig *et al.* [26], Zhang *et al.* [23], and proposed algorithm, respectively. The average PSNR deviations for these four algorithms are 0.68 dB, 0.63 dB, 0.62 dB, and 0.66 dB, respectively. For the random-access coding structure in Table XI, we can see that the average PSNR quality fluctuation ranges are (0.59, 0.94), (0.60, 0.75), (0.61, 0.73), and (0.62, 0.78) for the filter in HM16.9, Ledig *et al.* [26], Zhang *et al.* [23], and proposed filter, respectively. The average PSNR deviations for these four models are 0.74 dB, 0.67 dB, 0.76 dB, and 0.68 dB, respectively. From these results, we can see that the

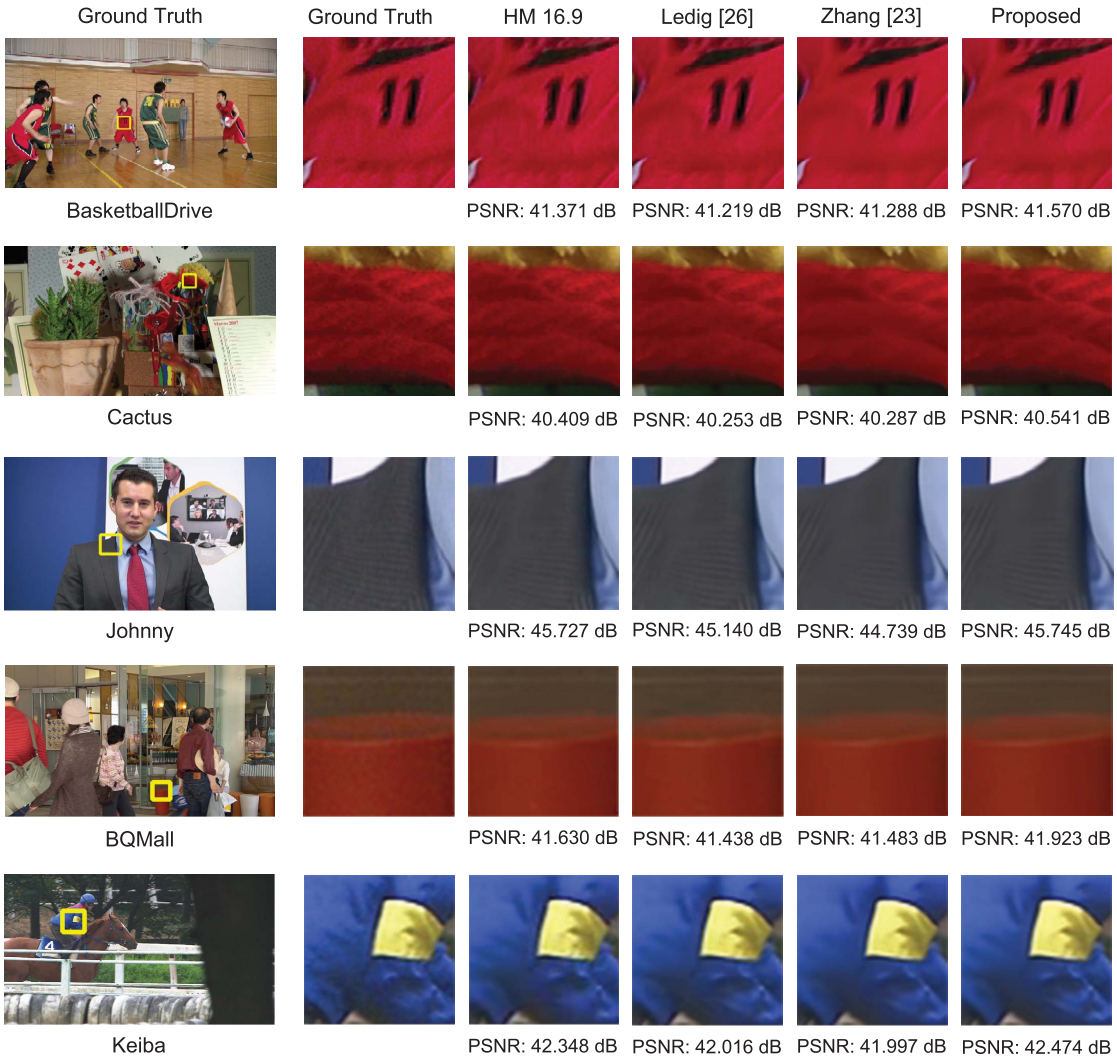


Fig. 13. Video subjective quality comparison ("BasketballDrive": the 150th frame with QP of 22; "Cactus": the 150th frame with QP of 22; "Johnny": the 150th frame with QP of 22; "BQMall": the 150th frame with QP of 22; "Keiba": the 100th frame with QP of 22.).

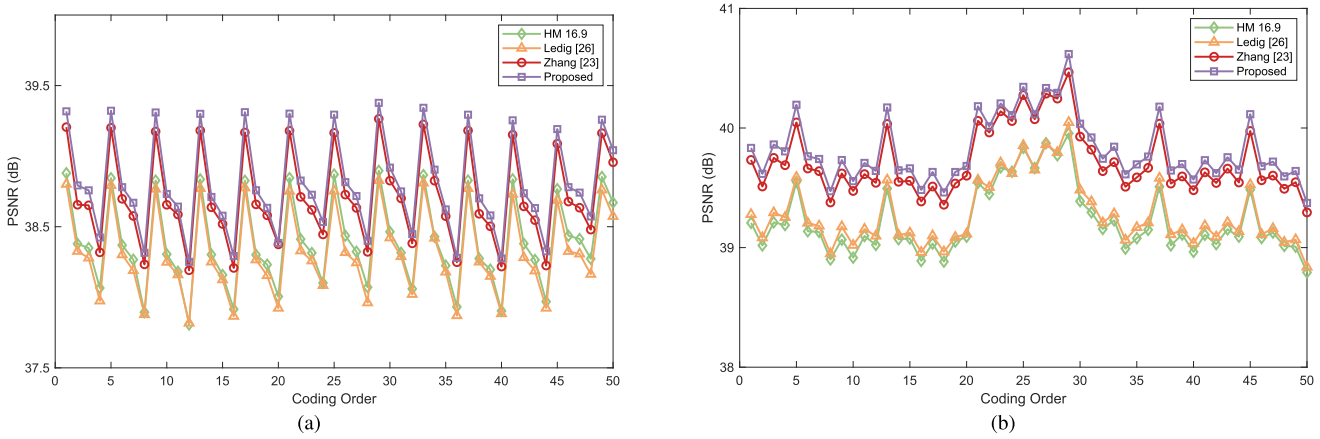


Fig. 14. PSNR fluctuation curves for different algorithms. Note that it shows the frame range (100∼150) of vidyo3 at QP32. (a) Low-delay coding structure. (b) Random-access coding structure.

PSNR fluctuation of the proposed algorithm is better than the original filter in HM 16.9, and achieves similar performance with Ledig's and Zhang's algorithms.

To intuitively show the video quality smoothness, the PSNR fluctuation curves of each filtering algorithm are presented in Figs. 14 and 15. Two video sequences are selected, one is

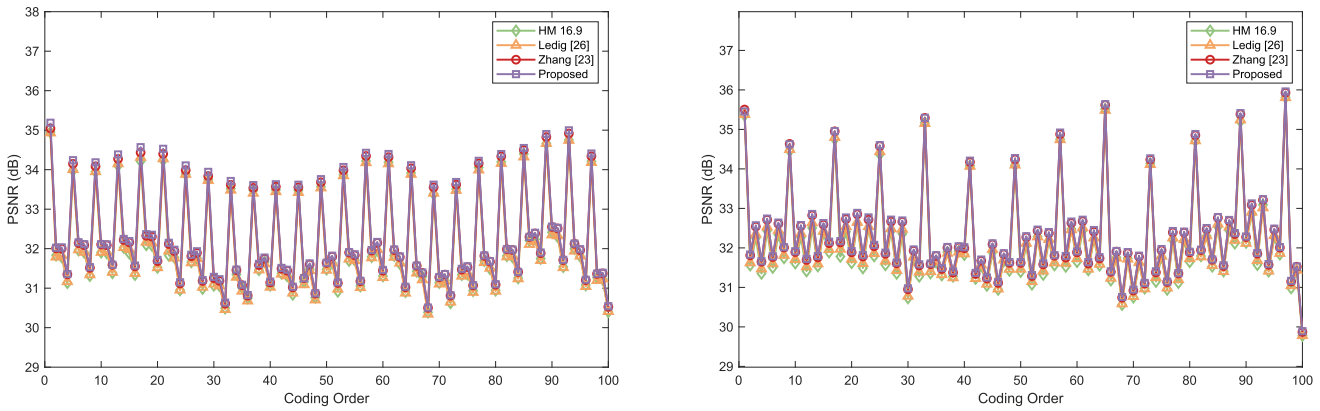


Fig. 15. PSNR fluctuation curves for different algorithms. Note that it shows the frame range (1∼100) of RaceHorses at QP32. (a) Low-delay coding structure. (b) Random-access coding structure.

TABLE XII
COMPUTATIONAL COMPLEXITY ANALYSES RESULTS (UNIT: %)

Coding Structure	Algorithm	2560×1600	1920×1080	1280×720	832×480	416×240	Average
Low-delay	Legig [26]	87	72	104	57	61	76
	Zhang [23]	208	172	251	143	138	182
	Proposed	195	160	236	130	138	172
Random-access	Legig [26]	124	107	149	87	99	113
	Zhang [23]	299	254	350	212	223	267
	Proposed	277	237	330	196	197	247

the "Vidyo3", which has large background content and moves slow; the other is the "RaceHorse", the content in this video has fast motion activity. It can be observed from these two figures that these four algorithms have quite similar PSNR fluctuation, and the proposed filtering algorithm achieves the best PSNR values among these four algorithms.

G. Comparisons on Computational Complexity and GPU Memory

To evaluate the computational complexity of each CNN-based filtering algorithm, the encoding time is used to calculate the complexity, and it is defined as

$$c = \left(\frac{T_\theta - T_h}{T_h}\right) \times 100\% \quad (\%), \tag{12}$$

where T_θ indicates the total encoding time consumed by CNN-based filtering algorithm, which includes CPU running time for the original HM16.9, and GPU testing time for CNN-based filtering. T_h represents the encoding time of the HM 16.9 with using original filtering algorithm. The analyses results of each CNN-based filtering algorithm are tabulated in Table XII.

From Table XII, we observe that when using the low-delay coding structure, the encoding time of Ledig *et al.* [26], Zhang *et al.* [23], and proposed algorithm increases an average of 76%, 182%, and 172%, respectively, as compared with the original HEVC encoder. When using the random-access coding structure, the growth rate of encoding time of Ledig *et al.* [26], Zhang *et al.* [23], and proposed algorithm is 113%, 267%, and 247%, respectively, as compared with the original HEVC filtering algorithm. We can see that the computational complexity of Zhang's algorithm is with the largest value, the reason is that its quantity of convolutional kernel is larger than the other two algorithms. In addition, we can see that the proposed fusion block has four sub-branches, which is larger than two branches in Zhang *et al.* [23], while the computational complexity of the proposed network is smaller than Zhang *et al.* [23], the reason is that the quantity of convolutional kernel in Zhang's highway unit is larger than that in our proposed fusion block. Overall, the proposed network achieves better computational complexity performance than Zhang *et al.* [23], and a similar performance with the Ledig's algorithm.

TABLE XIII
ANALYSES RESULTS OF MODEL SIZE AND GPU MEMORY

Algorithm	Model Size	GPU Memory
Ledig [26]	5.1 MB	3859 MB
Zhang [23]	27.2 MB	4825 MB
Proposed	18.2 MB	5193 MB

Furthermore, the model size and GPU memory of each CNN are analyzed. The model size reflects the number of network parameters, and the cost of GPU memory depends on the model size and the size of feature maps in hidden layers. The analyses results are listed in Table XIII. It can be seen that the model size of Ledig *et al.* [26], Zhang *et al.* [23], and proposed algorithm is 5.1 MB, 27.2 MB, and 18.2 MB, respectively. For the GPU memory consumption, the Ledig's algorithm consumes 3859 MB GPU memory, and 4825 MB GPU memory is needed for the Zhang's algorithm. The proposed algorithm needs 5193 MB GPU memory. From

these values, we can see that the model size of the proposed network is smaller than that of the Zhang's network, however, the proposed network consumes larger GPU memory than the cost of Zhang's network, the main reason is that the sub-branch of the proposed fusion block contains one 3×3 and one 1×1 convolution operations, while each branch of the highway unit in Zhang's network only contains one 3×3 convolution operation.

V. Conclusions

To reduce the artifacts for HEVC encoded videos, we proposed an in-loop filtering algorithm based on the EDCNN. Firstly, the current works for in-loop filtering are analyzed. Then, based on the analyses, the EDCNN with efficient network tools is proposed. The proposed EDCNN consists of a weighted normalization method, an efficient feature information fusion block, and a precise loss function. By utilizing the weight normalization, the internal covariate shift problem is solved with less noise. Moreover, with the utilization of feature information fusion block and precise loss function, more feature information can be retained to reconstruct a high quality video. The experimental results show that when compared with the original filter in HM16.9, the proposed EDCNN based in-loop filtering algorithm achieves an average of 6.45% BDBR reduction and 0.238 dB BDPSNR gains. The ability to eliminate artifacts of the proposed algorithm is much better than the traditional in-loop filtering algorithms.

References

[1] 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.

[2] M. Pourazad, C. Doutre, M. Azimi, and P. Nasiopoulos, "HEVC: The new gold standard for video compression: How does HEVC compare with H.264/AVC?" *IEEE Consum. Electron. Mag.*, vol. 1, no. 3, pp. 36–46, Jul. 2012.

[3] Z. Pan, J. Lei, Y. Zhang, and F. L. Wang, "Adaptive fractional-pixel motion estimation skipped algorithm for efficient HEVC motion estimation," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 14, no. 1, pp. 1–19, Jan. 2018.

[4] A. Norkin *et al.*, "HEVC deblocking filter," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1746–1754, Dec. 2012.

[5] 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.

[6] V. Sze, M. Budagavi, and G. J. Sullivan, *High Efficiency Video Coding (HEVC)* (Integrated Circuits and Systems). London, U.K.: Springer, 2014, pp. 200–203.

[7] Y. LeCun *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989.

[8] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. London, U.K.: Springer, 2014, pp. 184–199.

[9] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. London, U.K.: Springer, 2016, pp. 391–407.

[10] 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. (CVPR)*, Jun. 2016, pp. 1646–1654.

[11] 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.

[12] K. Zhang, W. Zuo, and L. Zhang, "FFDNet: Toward a fast and flexible solution for CNN-based image denoising," *IEEE Trans. Image Process.*, vol. 27, no. 9, pp. 4608–4622, Sep. 2018.

[13] K. Yang, S. Wan, Y. Gong, H. R. Wu, and Y. Feng, "Perceptual based SAO rate-distortion optimization method with a simplified JND model for H.265/HEVC," *Signal Process., Image Commun.*, vol. 31, pp. 10–24, Feb. 2015.

[14] 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 Mag.*, vol. 23, no. 2, pp. 16–26, Apr. 2016.

[15] 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.

[16] 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.

[17] X. Zhang, W. Lin, K. Gu, Q. Li, S. Wang, and S. Ma, "Transform-domain in-loop filter with block similarity for HEVC," in *Proc. Vis. Commun. Image Process. (VCIP)*, Nov. 2016, pp. 1–4.

[18] 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.

[19] W.-S. Park and M. Kim, "CNN-based in-loop filtering for coding efficiency improvement," in *Proc. IEEE 12th Image, Video, Multidimensional Signal Process. Workshop (IVMSP)*, Jul. 2016, pp. 1–5.

[20] Y. Dai, D. Liu, and F. Wu, "A convolutional neural network approach for post-processing in HEVC intra coding," in *Proc. MultiMedia Modeling (MMM)*, 2017, pp. 28–39.

[21] R. Yang, M. Xu, T. Liu, Z. Wang, and Z. Guan, "Enhancing quality for HEVC compressed videos," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 7, pp. 2039–2054, Jul. 2019.

[22] J. W. Soh *et al.*, "Reduction of video compression artifacts based on deep temporal networks," *IEEE Access*, vol. 6, pp. 63094–63106, 2018.

[23] 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.

[24] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[25] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*. [Online]. Available: http://arxiv.org/abs/1502.03167

[26] C. Ledig *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 105–114.

[27] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 901–909.

[28] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5987–5995.

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

[30] Y. Li *et al.*, "Convolutional neural network-based block up-sampling for intra frame coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 9, pp. 2316–2330, Sep. 2018.

[31] L. Zhang, L. Zhang, X. Mou, and D. Zhang, "A comprehensive evaluation of full reference image quality assessment algorithms," in *Proc. 19th IEEE Int. Conf. Image Process.*, Sep. 2012, pp. 1477–1480.

[32] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Loss functions for image restoration with neural networks," *IEEE Trans. Comput. Imag.*, vol. 3, no. 1, pp. 47–57, Mar. 2017.

[33] F. Bossen, *Common Test Conditions and Software Reference Configurations*, document JCTVC-B300, JCTVC-J1100, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 10th Meeting, Stockholm, Sweden, Jul. 2012.

[34] *HEVC Reference Model HM 16.9*. Accessed: Dec. 20, 2017. [Online]. Available: https://hevc.hhi.fraunhofer.de/trac/hevc/browser/tags

[35] G. Bjontegaard, *Calculation of Average PSNR Differences Between RD-Curves*, document TR VCEG-M33, ITU-T Q. 6/SG16 VCEG, 15th Meeting, Austin, Texas, USA, Apr. 2001.

Zhaoqing Pan (Senior Member, IEEE) received the Ph.D. degree in computer science from the City University of Hong Kong, Hong Kong, in 2014. In 2013, he was a Visiting Scholar with the Department of Electrical Engineering, University of Washington, Seattle, WA, USA, for six months. He is currently a Professor with the School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, China. His research interests focus on video coding and image quality assessment.

Xiaokai Yi received the B.S. degree in computer science and technology from the Nanjing University of Information Science and Technology, Nanjing, China, in 2016, where he is currently pursuing the M.S. degree in computer science and technology. His research interests focus on video coding and machine learning.

Yun Zhang (Senior Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from Ningbo University, Ningbo, China, in 2004 and 2007, respectively, and the Ph.D. degree in computer science from the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing, China, in 2010. From 2009 to 2014, he was a Postdoctoral Researcher with the Department of Computer Science, City University of Hong Kong, Hong Kong. From 2010 to 2017, he was an Assistant Professor and an Associate Professor with the Shenzhen Institutes of Advanced Technology (SIAT), CAS, Shenzhen, China, where he is currently a Professor. His research interests include video compression, 3D video processing, and visual perception.

Byeungwoo Jeon (Senior Member, IEEE) received the B.S. *(magna cum laude)* and M.S. degrees in electronics engineering from Seoul National University, Seoul, South Korea, in 1985 and 1987, respectively, and the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1992. From 1993 to 1997, he was with Signal Processing Laboratory, Samsung Electronics, South Korea, where he conducted research and development in video compression algorithms, the design of digital broadcasting satellite receivers, and other MPEG-related research for multimedia applications. Since September 1997, he has been with the faculty of the School of Electronic and Electrical Engineering, Sungkyunkwan University, South Korea, where he is currently a Full Professor. He has served as the Project Manager of digital TV and broadcasting in the Korean Ministry of Information and Communications from March 2004 to February 2006, where he supervised all digital TV-related R&D in Korea. He has authored many articles in the areas of video compression, pre/post processing, and pattern recognition. His research interests include multimedia signal processing, video compression, statistical pattern recognition, and remote sensing. He is a member of Tau Beta Pi and Eta Kappa Nu. He is also a member of SPIE, IEEK, KICS, and KSOBE. He was a recipient of the 2005 IEEK Haedong Paper Award in Signal Processing Society, South Korea.

Sam Kwong (Fellow, IEEE) received the B.S. degree in electrical engineering from the State University of New York at Buffalo in 1983, the M.S. degree in electrical engineering from the University of Waterloo, Waterloo, ON, Canada, in 1985, and the Ph.D. degree from the University of Hagen, Germany, in 1996. From 1985 to 1987, he was a Diagnostic Engineer with Control Data Canada. He joined Bell Northern Research Canada as a member of Scientific Staff. In 1990, he became a Lecturer with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong, where he is currently a Professor with the Department of Computer Science. His research interests include video and image coding and evolutionary algorithms.