PARTITION TREE GUIDED PROGRESSIVE RETHINKING NETWORK FOR IN-LOOP FILTERING OF HEVC

*Dezhao Wang, Sifeng Xia, Wenhan Yang, Yueyu Hu, Jiaying Liu**

Institute of Computer Science and Technology, Peking University, Beijing, China

ABSTRACT

In-Loop filter is a key part in High Efficiency Video Coding (HEVC) which effectively removes the compression artifacts. Recently, many newly proposed methods combine residual learning and dense connection to construct a deeper network for better in-loop filtering performance. However, the long-term dependency between blocks is neglected, and information usually passes between blocks only after dimension compression. To address these issues, we propose the Progressive Rethinking Block (PRB) to deliver long-term memory between the neighboring blocks and allow information to flow without compression, which is similar to human decision mechanism – usually reviewing the complete past memorized experiences to decide in the present, not just based on simple principles summarized before. PRBs further establish the Progressive Rethinking Network (PRN). In addition, we calculate the Multi-scale Mean value of Coding Units (MM-CU) to generate the side information maps which guide the training of the network by novelly telling the network architecture of the entire coding partition tree. Experimental results show that our proposed partition tree guided PRN provides 10.1% BD-rate reduction on average compared to the HEVC baseline.

***Index Terms*—** In-Loop Filter, High Efficiency Video Coding (HEVC), Video Compression

1. INTRODUCTION

The artifacts of lossy compressed video frames are mainly derived from two aspects. First, quantization erases the high frequency components of the frame which introduces the ringing artifacts. Second, frames are coded in units known as blocks. It inevitably results in blocking effects. To alleviate the compression artifacts, in-loop filter tools, *i.e.* the Deblocking Filter (DF) [1] and the Sample Adaptive Offset (SAO) [2], are introduced in video codecs like High Efficiency Video Coding (HEVC) [3] which effectively improves the coding performance.

Recently, deep learning methods [4, 5, 6] have obtained large performance gain over traditional methods [7, 8, 9] on multiple image processing tasks. Specifically, many methods introduce deep Convolutional Neural Networks (CNN) to remove the compression artifacts and successfully achieved considerable enhancement results. Dong *et al.* [10] proposed an Artifacts Reduction CNN (ARCNN) to eliminate JPEG artifacts. It gives inspiration to the video artifacts reduction. In [11], Dai *et al.* proposed the Variable-Filter-Size Residual-Learning CNN (VRCNN) to replace DF and SAO in HEVC as the post-processing component. In [12], Jia *et al.* first exploited temporal information as another input and built the Spatial-Temporal Residue Network (STResNet) as an additional in-loop filter. However, these networks are commonly shallow, which limits the performance of the filtering.

With the popularity of residual learning [13] and dense connection [14] techniques, some works explore to utilize them to build deeper networks for low-level image processing tasks. In [15], Zhang *et al.* used Residual Dense Blocks for image super-resolution which integrates multi-level features within a block by a 1×1 convolutional layer and appends a residual connection between the input and output of a block which facilitates better gradient back-propogation in training. Furthermore, the techniques are also adopted for in-loop filtering and further boost the performance. He *et al.* [16] used the residual learning technique to build a deeper network for in-loop filtering. In [17], Wang *et al.* built a Dense Residual Convolutional Neural Network (DRN) by stacking Dense Residual Units (DRU). By building deeper networks with the help of residual learning and dense connection, better compression artifacts removal results can be achieved. The existing networks benefit from stacking basic blocks. However, long-term memory is neglected and information at different depths of the network exchanges in limited ways, which significantly affects the learning performance.

In this paper, we propose a simple but effective approach to boost the network capacity to handle information flow. A Progressive Rethinking Network (PRN) is built with skip connections which are progressively added between neighbouring blocks before dimension compression. With the added skip connections, long-term memory can be additionally delivered so that the learning capacity of the network can be further improved. Moreover, different from the existing method

*Corresponding author. This work was supported in part by National Natural Science Foundation of China under contract No. 61772043, and in part by Beijing Natural Science Foundation under contract No. L182002 and No. 4192025.

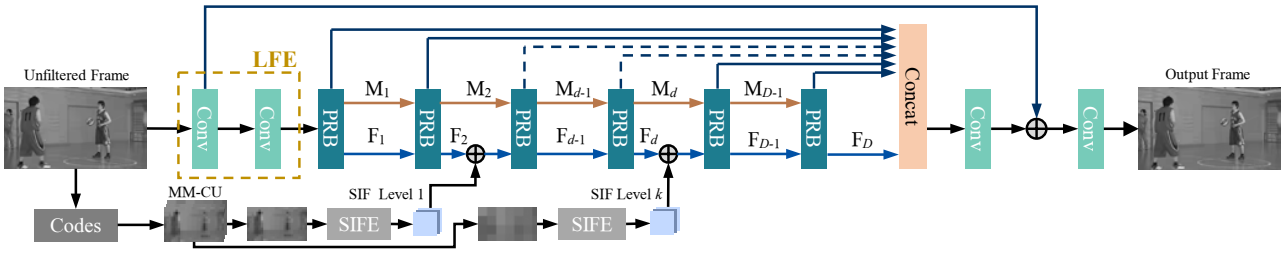


Fig. 1. The architecture of the Partition Tree Guided Progressive Rethinking Network. The network takes the unfiltered frame as the input and it generates the filtered output frame. The feature maps extracted from the side information by SIFE are added to data flow during the processing.

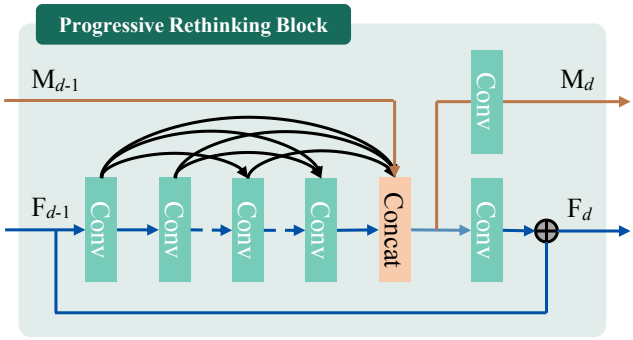


Fig. 2. The detailed structure of the Progressive Rethinking Block.

[16] which simply introduces final coding unit (CU) partition side information to facilitate the network training, we novelly tell the network more detailed coding tree architecture by generating Multi-scale Mean value of CU (MM-CU) as the side information so that both local and global coding structure can be fed into the network. Experimental results have also identified the effectiveness of our progressive inter-block connection and MM-CU side information.

The rest of the paper is organized as follows. Sec. 2 introduces the proposed method. The detail of MM-CU side information generation and the proposed network with progressive inter-block connection is presented. Experimental results are shown in Sec. 3 and concluding remarks are given in Sec. 4.

2. PARTITION TREE GUIDED PROGRESSIVE RETHINKING NETWORK

We first illustrate our main contributions to construct our partition tree guided progressive rethinking network. Then, the details of progressive connection in the Progressive Rethinking Blocks (PRB) and the generation and fusion process of the coarse-to-fine MM-CU feature will be explained later in detail. We integrate our network as an additional filter between DF and SAO. Namely, each frame should firstly pass DF before fed into our network. The output frame of our network is processed by SAO afterwards.

2.1. Motivation

Our motivation to construct the proposed network is two-fold:

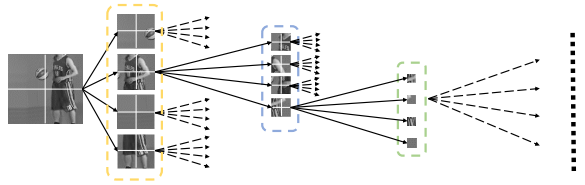
- **Progressive Representative Feature Review**. Previous restoration networks [15] perform progressive feature refinement. However, when the generated new information is fused with the past information, the feature dimension is usually compressed. Some high dimensional representative features are lost. In our work, we construct an additional connection between blocks to maintain this information, which is denoted by the orange path shown in Figs. 1 and 2.

- **Hierarchical Side Information**. The coding process is performed block by block with the coding tree unfolding. Thus, the guidance side information should reflect this property, for a better context modeling, and enabling the network to perform restoration at different levels. In this work, we extract MM-CU as side information to boost the proposed network for better in-loop filter, as shown in Fig. 3.

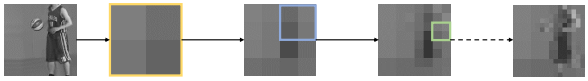
2.2. Progressive Rethinking Network

Overall. Our overall network framework is shown in Fig. 1. The main branch of the framework is the PRN which first extracts low-level features by a Low-level Feature Extractor (LFE) and then stacks sequential PRBs with progressive connections between neighbouring blocks. Apart from feeding the coded frame into the network, we generate coarse-to-fine MM-CU as guidance. The feature maps of MM-CU are respectively extracted through Side Information Feature Extractors (SIFE) and then fused with the main branch after different PRBs.

Low-level Feature Extractor (LFE). The unfiltered frame x first is fed into an LFE consisting of two convolutional layers



(a) Partition Tree



(b) Multi-Scale Mean value of CU

Fig. 3. (a) shows a partition tree generated by the codec. We calculate the mean value of each CU at different levels to form the side information maps. (b) shows the results of side information maps of different scales.

to extract shallow feature F_0 which is to be fed into sequential PRBs. The process is denotes as $\mathrm{P}_{\mathrm{LFE}}(\cdot)$:

$$F_0 = \mathrm{P}_{\mathrm{LFE}}(x). \tag{1}$$

Progressive Rethinking Block (PRB). Beyond previous work RDB in [15], we further connect the information flow between different blocks before dimension compression as Fig. 2 shows. For the k-*th* PRB, it receives feature maps F_k as a common input to generate multi-level feature maps G_k. We name the nonlinear procedure as H_k and then G_k is obtained by:

$$G_k = H_k(F_k). \tag{2}$$

M_k works as a long-term memory which is also generated by the previous PRB. We concatenate M_k with G_k. After that, we use two 1×1 convolutional layers to respectively extract two new feature maps M_{k+1} and F_{k+1}, denoted by $\mathrm{P}_{\mathrm{M}}(\cdot)$ and $\mathrm{P}_{\mathrm{F}}(\cdot)$ as follows,

$$M_{k+1} = \mathrm{P}_{\mathrm{M}}([G_k, M_k]), \tag{3}$$

$$F_{k+1} = \mathrm{P}_{\mathrm{F}}([G_k, M_k]), \tag{4}$$

where $[\cdot]$ dentoes the concatenation operation. P_{M} and F_{M} usually take the form of dense network.

Progressive Rethinking Network (PRN). We stack multiple PRBs to construct our PRN. It should be noted that all PRBs indeed need two inputs: M_k and F_k. We simply set $M_0 = F_0$ as the initial. So, the formula can be written as:

$$F_k, M_k = \mathrm{P}_{\mathrm{PRB}}(F_{k-1}, M_{k-1}). \tag{5}$$

where $\mathrm{P}_{\mathrm{PRB}}(\cdot)$ denotes the working process of PRB. After generating F_D (D is the number of PRBs), we concatenate all feature maps $F_0, F_1, ..., F_D$ together and use a 1×1 convolutional layer, denoted by $\mathrm{P}_{\mathrm{Compress}}(\cdot)$, to compress them as follows:

$$F_{D+1} = \mathrm{P}_{\mathrm{Compress}}\left([F_1, F_2, F_3..., F_D]\right). \tag{6}$$

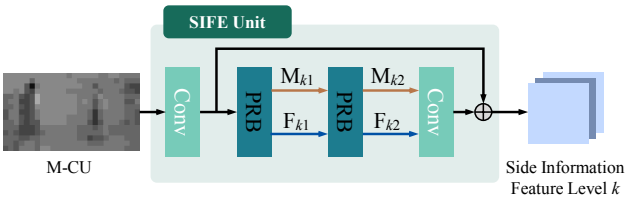


Fig. 4. The detailed structure of the SIFE unit.

We append a global residual, which faciliates better detail modeling and network training. After two convolutional layers, the output frame is finally reconstructed.

2.3. MM-CU Generation and Fusion

Apart from the main branch of the network, we additionally generate the MM-CU side information and add it to the main branch to guide the network learning. In this paper, we extract the MM-CU based on the coding partition tree from coarse to fine.

As HEVC standard proposes, each CU can be recurrently divided into four smaller CUs. Fig. 3 (a) shows an example partition tree. Since HEVC codecs encode a frame at CU levels independently with different coding parameters, the partition information contains a lot of extra important side information which can facilitate removing the coding artifacts. Different from generating mean value of CU (M-CU) at the bottom layer of the quadtree [16], we extract M-CU in each layer. Namely, we calculate the mean value of a CU everytime a partition happens. Consequently, the side information can guide the network to remove the coding artifacts at different scales according to the entire coding partition architecture.

The way we generate M-CU is shown in Fig. 3. As Fig. 3 (a) shows, HEVC codec parts a frame into multiple coding tree units (CTU) first. Then the CTUs are divided progressively in a quadtree manner. We calculate the mean value of each CU at different levels from coarse to fine to derive the corresponding side information maps. Blocks framed by the yellow dotted box are four CTUs and their corresponding M-CU side information map which is the coarsest ones is shown in Fig. 3 (b) with a yellow border. Then, everytime the CUs are parted into four smaller CUs, we recurrently calculate the mean value of each partitioned CU. If the CU is not parted, we will keep its side information value the same with that of the upper level, which means that the side information value of that CU will be kept unchanged afterwards. We do the procedure recurrently until the last level. Finally, the multi-scale M-CU side information maps can be obtained.

With the derived multi-scale side information maps, we use a shallow CNN as a feature extractor to extract feature maps from them. As Fig. 4 shows, the architecture of the extractor is similar to PRN but much shallower. We name the output feature maps of the k-*th* M-CU as SF_k.

It is intuitive that finer M-CU maps contain more local details of the coding architecture while coarser ones contain some global coding structure information. Thus, we fuse coarser M-CU maps with the main branch in deeper layers so that global information can further guide the learning of the network after larger area is perceived in deeper layers.

When it comes to the implementation of fusion, we simply choose to use element-wise add. We fuse SF_k after the n_k *th* PRB of the PRN by:

$$F_{n_k} = F_{n_k} + SF_k. \tag{7}$$

It not only brings about considerable performance gains but also uses relatively fewer additional network parameters.

In this paper, we choose to stack 10 PRBs to establish our baseline considering the balance of performance and computation complexity. In each PRB, the multi-level feature map is generated by 6 convolutional layers with dense connection.

We choose HM 16.15 as implementation whose default settings provide at most four-layer partition quadtree. So we can get MM-CU of four levels. We respectively add the feature maps of MM-CU after the 2-*nd*, 4-*th*, 6-*th*, 8-*th* PRB of the PRN baseline.

3. EXPERIMENTAL RESULTS

3.1. Implementation

Training Data. We choose DIV2K [18] to generate our training data. The dataset provides us 800 2K images. We generate the coded frames by HM 16.15 and for the consistency, we truncate the frames before SAO. Every image is randomly cropped into 64×64 patches and randomly filped both horizontally and vertically for augmentation. We compress the training data using different QPs so as to train different models for each corresponding QP.

Training Protocol. Mean Square Error (MSE) is adopted as the loss function. Let X_i denote the unfiltered frame and Y_i denote the ground truth. Besides the highlighting 1×1 convolutional kernels, the sizes of other convolutional kernels are all 3×3. We use Θ to represent the network parameters so that the frame filtered by our proposed method can be denoted as $F(X_i; \Theta)$. Therefore, our training is to find the optimal Θ to minimize the following loss:

$$L(\Theta) = \frac{1}{n}\sum_{i=1}^{n} ||F(X_i; \Theta) - Y_i||_2, \tag{8}$$

where n is the number of training samples. We use ReLU as our activation function.

The network is implemented on Pytorch and Adam is used as the optimizer with $\beta_1 = 0.9, \beta_2 = 0.999$. The learning rate is firstly set to 10^{-4} and turned down adaptively until convergence. We terminate training at 75 epochs.

Table 1. BD-rate reduction of MM-CU guided PRN over HM-16.15 baseline

Class	Sequence	BD-rate(%)	
C	RaceHorses	-6.0	-9.4
	BQMall	-9.9	
	PartyScene	-6.2	
	BasketballDrill	-15.5	
D	RaceHorses	-10.7	-9.7
	BQSquare	-9.2	
	BlowingBubbles	-7.8	
	BasketballPass	-11.0	
E	FourPeople	-12.7	-11.5
	Johnny	-11.0	
	KristenAndSara	-10.7	
Average		-10.1	

We train our model on Y channel of the frame. Therefore, only the performance on Y channel is taken into consideration. We test our model under all-intra (AI) configuration at four levels (QP=22, 27, 32, 37).

3.2. Comparison with HEVC baseline

Due to time constraints, we test our model on the first second of the HEVC benchmark sequences [19] from Class C to Class E. The rate-distortion performance is measured by Bjontegaard Distortion-rates (BD-rate) [20]. The experimental results are shown in Table 1. According to the table, our proposed method can obtain on average 10.1% and up to 15.5% gain over HEVC baseline. More ablation experimental results can be found on our website[1].

4. CONCLUSION

In this paper, a Progressive Rethinking Network based in-loop filter with partition tree guided side information is proposed. Besides utilizing residual learning and dense connection inside the block, we further propose a novel Progressive Rethinking Block which delivers long-term memory between neighbouring blocks. The added skip connections fuse the high-dimensional features before dimension compression in each basic block, preserving richer representative information for in-loop filter. In addition, we develop coarse-to-fine MM-CU deriving from the partition tree which not only contains boundary details but also tells the structure of the frame. By fusing the feature maps of MM-CU element-wisely to PRN, the new in-loop filter provides 10.1% BD-rate reduction on average and 15.5% at most compared to the HEVC baseline.

[1] https://huzi96.github.io/PRN.html

5. REFERENCES

[1] A. Norkin, G. Bjontegaard, A. Fuldseth, M. Narroschke, M. Ikeda, K. Andersson, M. Zhou, and G. Van der Auwera, "HEVC Deblocking Filter," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1746–1754, 2012.

[2] C. M. Fu, Elena Alshina, Alexander Alshin, Y. W. Huang, C. Y. Chen, C. Y. Tsai, C. W. Hsu, S. M. Lei, J. H. Park, and W. J. Han, "Sample Adaptive Offset in the HEVC standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1755–1764, 2012.

[3] G. J. Sullivan, J. Ohm, W. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[4] C. Dong, C. C. Loy, K. M. He, and X. O. Tang, "Image super-resolution using deep convolutional networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, 2016.

[5] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2016.

[6] K. Zhang, W. M. Zuo, Y. J. Chen, D. Y. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, 2017.

[7] M. D. Li, J. Y. Liu, W. H. Yang, X. Y. Sun, and Z. M. Guo, "Structure-revealing low-light image enhancement via robust retinex model," *IEEE Transactions on Image Processing*, vol. 27, no. 6, pp. 2828–2841, 2018.

[8] S. Yang, J. Y. Liu, Y. M. Fang, and Z. M. Guo, "Joint-feature guided depth map super-resolution with face priors," *IEEE Transactions on Cybernetics*, vol. 48, no. 1, pp. 399–411, 2018.

[9] J. Ren, J. Y. Liu, W. Bai, and Z. M. Guo, "Similarity modulated block estimation for image interpolation," in *Proc. IEEE Int'l Conf. Image Processing*, 2011.

[10] D. Chao, Y. B. Deng, C. C. Loy, and X. O. Tang, "Compression Artifacts Reduction by a Deep Convolutional Network," in *IEEE Int'l Conf. on Computer Vision*, 2015.

[11] Y. Y. Dai, D. Liu, and F. Wu, "A convolutional neural network approach for post-processing in HEVC intra coding," in *Proc. International MultiMedia Modeling Conf.*, 2017.

[12] C. M. Jia, S. Q. Wang, X. F. Zhang, S. S. Wang, and S. W. Ma, "Spatial-temporal residue network based in-loop filter for video coding," in *Proc. IEEE Visual Communications and Image Processing*, 2017.

[13] K. M. He, X. Y. Zhang, S. Q. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2016.

[14] G. Huang, Z. Liu, L. Van Der Maaten, and K. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2017.

[15] Y. L. Zhang, Y. P. Tian, Y. Kong, B. N. Zhong, and Y. Fu, "Residual dense network for image super-resolution," in *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2018.

[16] X. Y. He, Q. Hu, X. Y. Zhang, C. Y. Zhang, W. Y. Lin, and X. T. Han, "Enhancing HEVC compressed videos with a partition-masked convolutional neural network," in *Proc. IEEE Int'l Conf. Image Processing*, 2018.

[17] Y. B. Wang, Z. Han, Y. M. Li, Z. Z. Chen, and S. Liu, "Dense Residual Convolutional Neural Network based In-Loop Filter for HEVC," in *Proc. IEEE Int'l Conf. Image Processing*, 2018.

[18] E. Agustsson and R. Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study," in *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition Workshop*, 2017.

[19] K. Sharman and K. Suehring, "Common test conditions," in *document JCTVC-Z1100, Joint Collaborative Team on Video Coding*, 2017.

[20] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," in *document VCEG-M33, ITU-T Video Coding Experts Group Meeting*, 2001.