JVET-AB0090

EE1-1.3 related: Lightweight and Efficient CNN In-loop Filter

Hao Zhang[1], Cheolkon Jung[1], Yang Liu[2], and Ming Li[2]

[1]Xidian University, China

[2]OPPO, China

Introduction

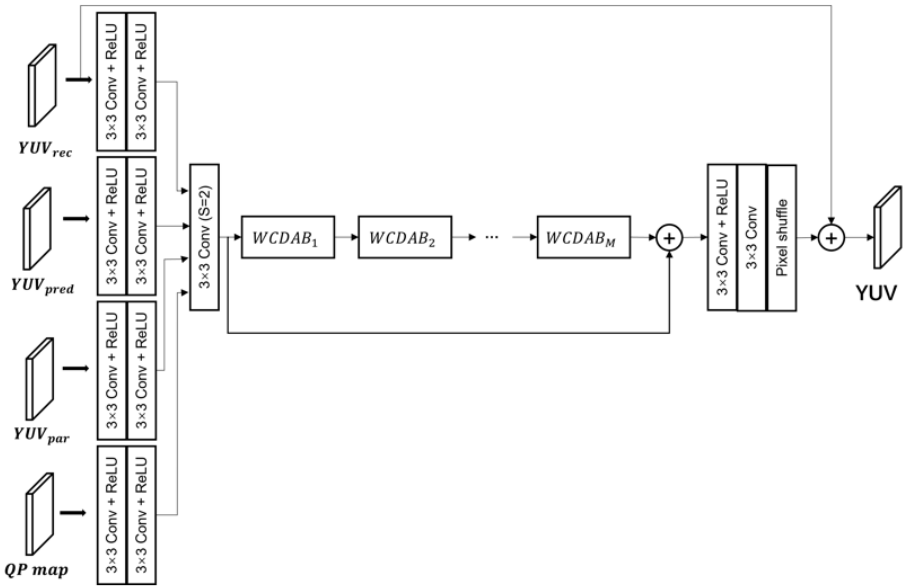
In this contribution, the CNN-based in-loop filter proposed in JVET-AA0074 (AA0074 filter) is made lightweight and efficient.

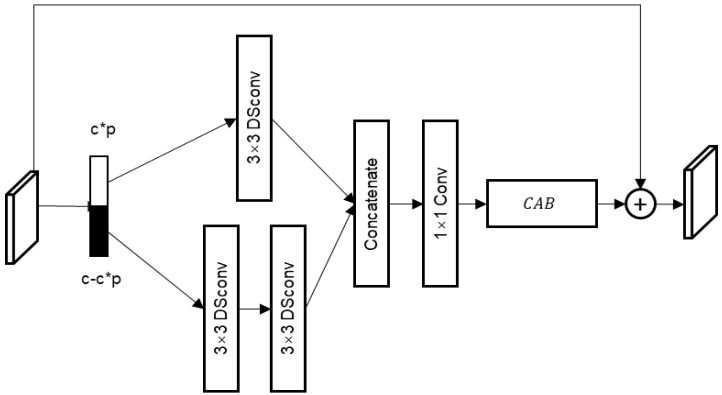
- The number of skip connections, convolution layers and channels are reduced in each residual block, and correspondingly increased the number of residual blocks to ensure good learning ability

- A multi-stage training strategy is proposed to train the proposed network taking advantage of progressive learning(In AI configuration)

- Verify the performance of the improved network using AA0074 training strategy (In RA configuration)

Proposed Network in EE1-1.3





Proposed network

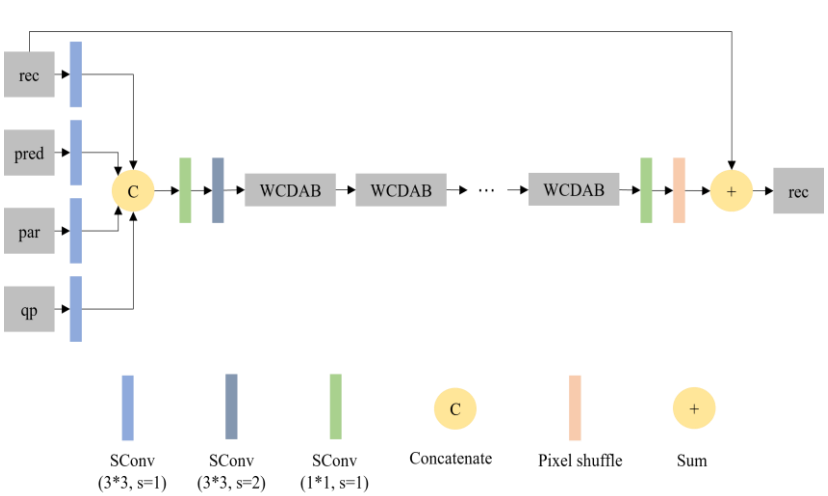
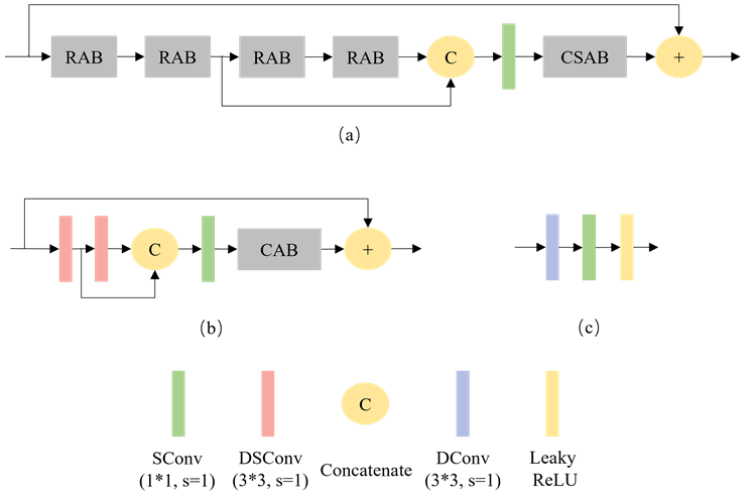


Illustration of the proposed NN-based in-loop filter. (SConv is the standard convolution)



(a) The architecture of WCDAB. (b) The architecture of RAB. (c) The architecture of Depthwise separable convolution. (SConv is standard convolution, DSConv is depthwise separable convolution, and DConv is depthwise convolution)

Training Detail

Dataset: DIV2K and BVI-DVC

Loss function: weighted L1 and L2 loss

$$L_{total} = 8 * L(Y_{\text{out}}, Y_{label}) + L(U_{\text{out}}, U_{label}) + L(V_{\text{out}}, V_{label})$$

Training data acquisition：Obtain reconstruction map, prediction map and partition map after LMCS, and obtain labels after ALF

Training strategy

In this contribution, a multi-stage training strategy is proposed to maximize the performance of the model during training.

- A parameter qp_dis is used, which represents the QP difference between the network input and the label. First, a smaller qp_dis is used to train the network, and then gradually increase qp_dis after the network converges while continuing to train the network.

- Since our loss function is multi-stage loss, we combine the loss function with the training strategy, i.e. a multi-stage training strategy.

Training strategy

Network performance at each training stage in terms of BD-rate (Class D in AI configuration).

Stage	Y-PSNR	U-PSNR	V-PSNR
qp_dis=5 & L1 loss	-4.61%	-14.09%	-14.59%
qp_dis=5 & L2 loss	-5.60%	-14.84%	-17.12%
qp_dis=10 & L1 loss	-6.12%	-14.39%	-17.23%
qp_dis=10 & L2 loss	-6.63%	-14.65%	-16.90%
qp_dis=15 & L2 loss	-6.91%	-13.44%	-15.53%

For specific qp_ dis, the difference of image quality corresponding to Y channel is far greater than that of UV channel. This results in that the proposed training strategy cannot train both luminance and chrominance components at the same time.

Therefore, in actual training, qp_dis is only used for the Y channel. For the UV channel, its label uses the uncompressed image.

This problem can be solved by training luminance and chrominance filters respectively.

Network Information in Training Stage

Network Information in Training Stage		
Mandatory	GPU Type	GPU: GeForce RTX 3090
	Framework:	PyTorch v1.9.0
	Number of GPUs per Task	1
	Epoch:	800
	Batch size:	32
	Training time:	~200h/model
	Training data information:	BVI-DVC, DIV2K
	Training configurations for generating compressed training data (if different to VTM CTC):	VTM-11.0, QP {12,17,22, 27, 32, 37, 42}
	Loss function:	Weighted L1 and L2
Optional		
	Number of iterations	3100
	Patch size	144x144
	Learning rate:	1e-4
	Optimizer:	ADAM
	Preprocessing:	random cropping
	Other information:	

Network Information in Inference Stage

Network Information in Inference Stage		
Mandatory	HW environment:	
	GPU Type	CPU only
	Framework:	Libtorch v1.9.0
	Number of GPUs per Task	0
	Number of Parameters (Each Model)	0.78M
	Total Number of Parameters (All Models)	0.78M
	Parameter Precision (Bits)	32
	Memory Parameter (MB)	3.12M
	Multiply Accumulate (MAC)/pixel	200K
Optional		
	Total Conv. Layers	64 Depthwise separable convolution 56 Common convolution
	Total FC Layers	72
	Total Memory (MB)	
	Batch size:	1
	Patch size	144x144
	Changes to network configuration or weights required to generate rate points	
	Peak Memory Usage (Total)	
	Peak Memory Usage (per Model)	
	Border handling	
	Other information:	

Experimental Results

Performance of the proposed method in AI configuration

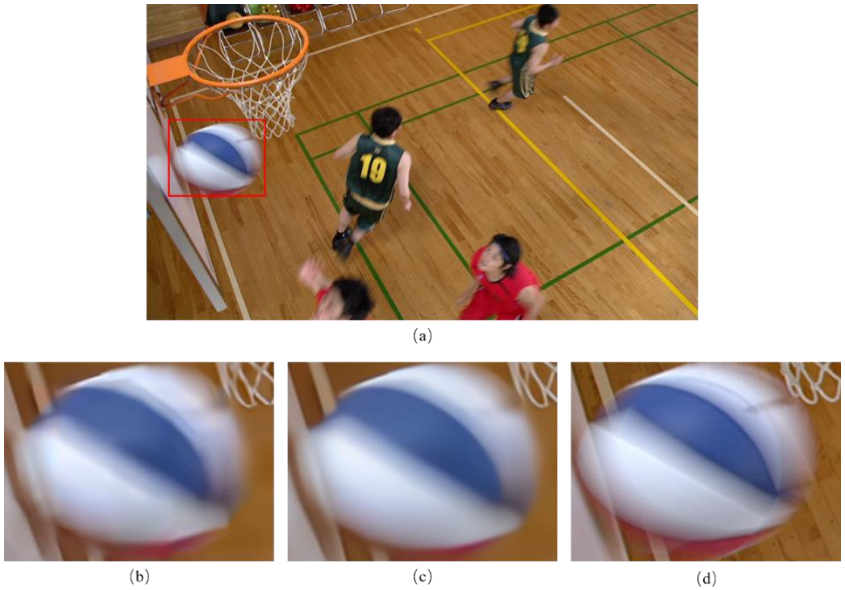
	All Intra Main10							
	BD-rate Over VTM-11.0_nnvc-2.0							
	Y-PSNR	U-PSNR	V-PSNR	Y-MSIM	U-MSIM	V-MSIM	EncT	DecT CPU
Class A1	-6.76%	-6.17%	-8.38%	-8.17%	-11.39%	-15.39%	143%	22105%
Class A2	-6.57%	-13.20%	-16.81%	-7.31%	-18.98%	-19.55%	121%	17413%
Class B	-6.41%	-12.49%	-9.84%	-6.51%	-18.25%	-16.60%	119%	11670%
Class C	-6.82%	-14.77%	-14.83%	-6.88%	-18.70%	-19.75%	110%	11838%
Class E	-9.36%	-14.86%	-15.12%	-9.71%	-18.62%	-20.69%	123%	22867%
Overall	-7.08%	-12.46%	-12.75%	-7.53%	-17.39%	-18.27%	122%	15572%
Class D	-6.91%	-13.44%	-15.53%	-6.30%	-18.48%	-20.83%	109%	10682%
Class F	-4.46%	-11.65%	-10.84%	-4.72%	-14.08%	-13.59%	108%	17178%
Class H	#VALUE!	#VALUE!	#VALUE!	#VALUE!	#VALUE!	#VALUE!	#DIV/0!	#DIV/0!

Experimental Results

Performance of the proposed method in RA configuration

	Random access Main10							
	BD-rate Over VTM-11.0_nnvc-2.0							
	Y-PSNR	U-PSNR	V-PSNR	Y-MSIM	U-MSIM	V-MSIM	EncT	DecT CPU
Class A1	-6.02%	-10.70%	-12.77%	-7.12%	-15.12%	-15.88%	118%	#REF!
Class A2	-6.23%	-15.45%	-20.00%	-6.07%	-18.65%	-21.15%	101%	#REF!
Class B	-5.33%	-18.48%	-14.11%	-5.22%	-19.95%	-16.81%	113%	#REF!
Class C	-4.50%	-14.80%	-13.24%	-4.29%	-14.49%	-12.77%	191%	#REF!
Class E								
Overall	-5.43%	-15.34%	-14.79%	-5.52%	-17.27%	-16.41%	128%	#REF!
Class D	-5.78%	-14.63%	-13.80%	-3.95%	-15.74%	-12.86%	132%	#REF!
Class F	-2.27%	-9.67%	-8.04%	-2.15%	-10.03%	-8.82%	143%	#REF!
Class H	#VALUE!	#VALUE!	#VALUE!	#VALUE!	#VALUE!	#VALUE!	#DIV/0!	#REF!

Visual comparison



Visual quality comparison on the test datasets. (a) Video frame of Class A2-CatRobot. (b) Result of VTM-11.0_NNVC-2.0. (c) Result of the proposed filter. (d) Ground truth.

Conclusions

- Fine-tune the structure of AA0074 filter, make it more lighter.
- Propose a multi-stage training strategy to take advantage of progressive learning in the network training.
- The proposed CNN filter achieve average {7.08%, 12.46%, 12.75%} and {5.43%, 15.34%, 14.79%} BD-rate reductions over VTM-11.0-NNVC-2.0 for {Y, Cb, Cr} channels in AI and RA configurations.
- We recommend further study on the proposed CNN filter by including it in EE1.

THANK YOU!