Tencent 腾讯 interdigital

JVET-Z0091

EE1-1.2: neural network based in-loop filter with a single model

Liqiang Wang, Xiaozhong Xu, Shan Liu (Tencent)

Franck Galpin (InterDigital)

Outline

- Introduction
- Proposed method
- Results
- Conclusions

Introduction

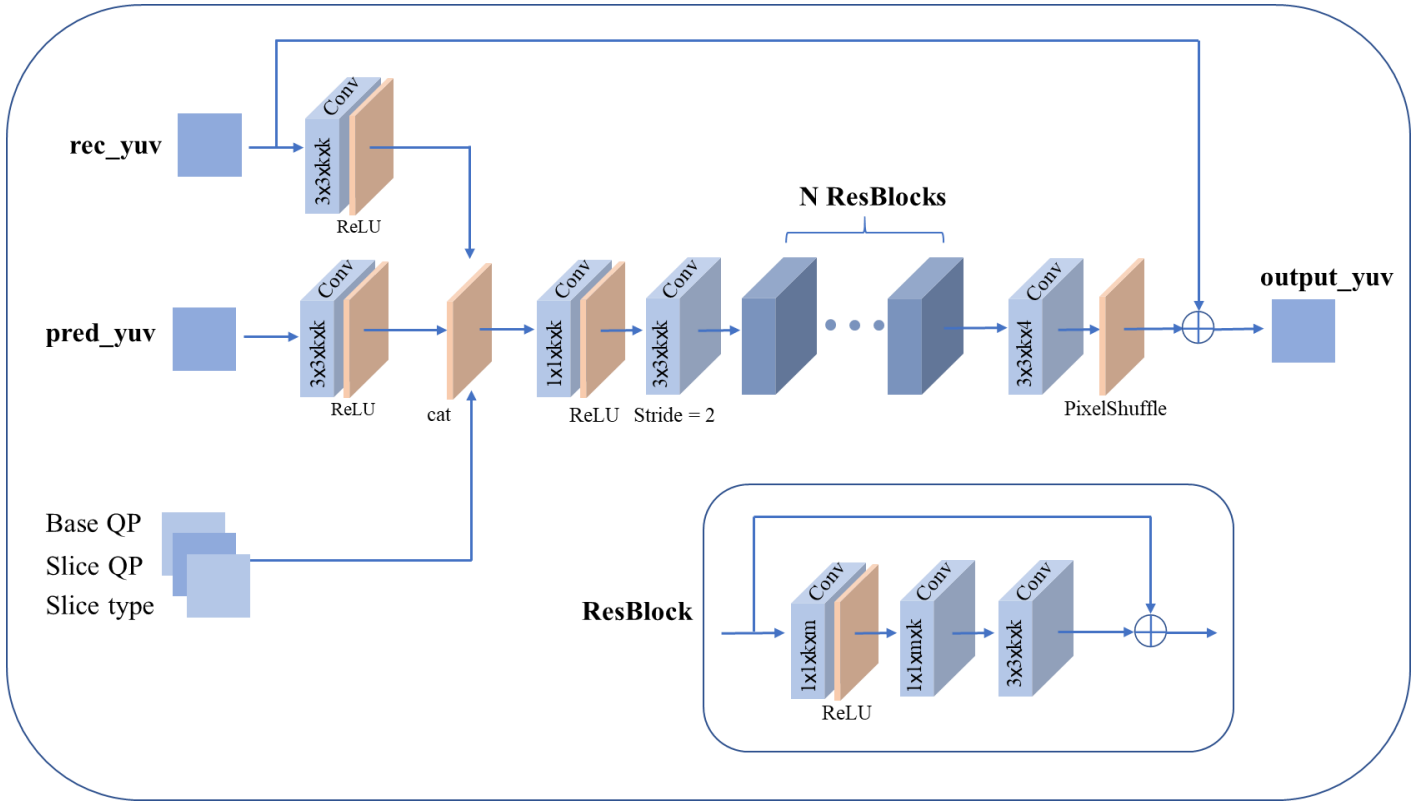
Optimize the trade off based on JVET-Y0091.

- Only one model is used in the proposed filter design
- For both I and B slices, the optimal filtered result is decided between the two outputs from the proposed filter and SAO.

- More side information is utilized, such as the prediction image, the slice QP, the based QP and the slice type.
- The main difference with JVET-Y0078 is shown as follows:
  - The convolutional block attention module is removed for the simplicity. It seems to have a tiny influence on the performance.
  - A simpler Nearest method is used to replace Lanczos method in the resampling process for chroma components.
  - Multi-scaling refinement method, like the one in JVET-Y0098, is used to further improve the performance.
  - SADL deployment is studied in EE1-1.2.2 and EE1-1.2.3.

Proposed Method

The network architecture of the proposed method.



The backbone of this network is cascaded by serval Resblocks. As for the structure of Resblock:

- It consists of two 1x1 and one 3x3 convolutional layers.
- The number of channels firstly goes up before the activation layer, and then goes down after the activation layer.

Proposed Method

Network information in the training and inference stages.

Network Information in Training Stage		
Mandatory	GPU Type	Tesla A100 40GB
	Framework:	Torch v1.9.0
	Number of GPUs per Task	1
	Epoch:	~1000
	Batch size:	64
	Loss function:	L1
	Training time:	~150h
	Training data information:	DIV2K, TVD, BVI-DVC
	Training configurations for generating compressed training data (if different to VTM CTC):	
Optional		
	Number of iterations	1200
	Patch size	144x144
	Learning rate:	1e-4
	Optimizer:	ADAM
	Preprocessing:	flipped, rotated
	Mini-batch selection process:	random cropped
	Other information:	

Network Information in Inference Stage			
	Test items	Test 1.2.1 / Test 1.2.2	Test 1.2.3
Mandatory	HW environment:		
	GPU Type	CPU only	
	Framework:	Libtorch v1.9.0 or SADL	
	Number of GPUs per Task	0	
	Number of Parameters (Each Model)	1.9M	
	Total Number of Parameters (All Models)	1.9M	
	Parameter Precision (Bits)	32	16
	Memory Parameter (MB)	7.6MB/model, 1 model in all	3.8MB/model, 1 model in all
	Multiplay Accumulate (MAC)/pixel, worst-case	485K	
Optional			
	Total Conv. Layers	101	
	Total FC Layers	0	
	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:		

Proposed Method

Implementation

- For I slices and B slices, Deblock and SAO are both enabled. The optimal filtered result is decided between the two outputs from the proposed filter and the SAO.
- A scaling operation is carried out to refine the result of NN filter. The scaling factors are signaled in the slice header for each component. The specific process is the same as the scaling process used in JVET-W0130. Additionally, there are 3 fixed weights to blend the results of the proposed filter and the conventional filter. These fixed weights are 1, 0.75 and 0.5, which are similar as those in JVET-Y0098.
- The proposed filter can be turned on/off at the CTU level and slice level.

Results

EE1-1.2.1
Libtorch, flt32

	All Intra Main10 BD-rate Over VTM-11.0_nnvc-1.0					
	YUV-PSNR	Y-PSNR	U-PSNR	V-PSNR	EncT	DecT CPU
Class A1	-8.33%	-6.18%	-13.48%	-16.09%	148%	22594%
Class A2	-7.70%	-5.80%	-14.77%	-12.04%	125%	18936%
Class B	-8.25%	-5.86%	-14.79%	-16.04%	121%	18100%
Class C	-9.06%	-6.44%	-15.94%	-17.95%	112%	13759%
Class E	-10.53%	-8.69%	-15.12%	-17.00%	124%	20774%
Overall	**-8.73%**	**-6.50%**	**-14.88%**	**-15.96%**	124%	18218%
Class D	-8.95%	-6.47%	-15.00%	-17.77%	110%	15079%
Class F	-5.87%	-4.06%	-11.21%	-11.44%	110%	15172%

	Random access Main10 BD-rate Over VTM-11.0_nnvc-1.0					
	YUV-PSNR	Y-PSNR	U-PSNR	V-PSNR	EncT	DecT CPU
Class A1	-11.35%	-9.34%	-15.99%	-18.74%	124%	34810%
Class A2	-11.06%	-9.30%	-17.82%	-14.84%	123%	33875%
Class B	-11.06%	-8.29%	-18.99%	-19.70%	124%	35045%
Class C	-11.41%	-8.20%	-20.74%	-21.33%	115%	31965%
Class E						
Overall	**-11.21%**	**-8.68%**	**-18.62%**	**-18.97%**	121%	33918%
Class D	-11.82%	-9.02%	-19.10%	-21.32%	115%	32507%
Class F	-5.79%	-3.77%	-12.08%	-11.64%	139%	15108%

EE1-1.2.2
SADL, flt32

	All Intra Main10 BD-rate Over VTM-11.0_nnvc-1.0					
	YUV-PSNR	Y-PSNR	U-PSNR	V-PSNR	EncT	DecT CPU
Class A1	-8.33%	-6.18%	-13.48%	-16.09%	351%	140807%
Class A2	-7.70%	-5.80%	-14.78%	-12.06%	237%	113229%
Class B	-8.25%	-5.86%	-14.79%	-16.03%	218%	111593%
Class C	-9.06%	-6.44%	-15.94%	-17.95%	171%	84772%
Class E	-10.53%	-8.69%	-15.12%	-17.00%	247%	131778%
Overall	**-8.73%**	**-6.50%**	**-14.88%**	**-15.97%**	232%	112467%
Class D	-8.95%	-6.47%	-15.00%	-17.77%	164%	92188%
Class F	-5.87%	-4.06%	-11.21%	-11.44%	158%	94921%

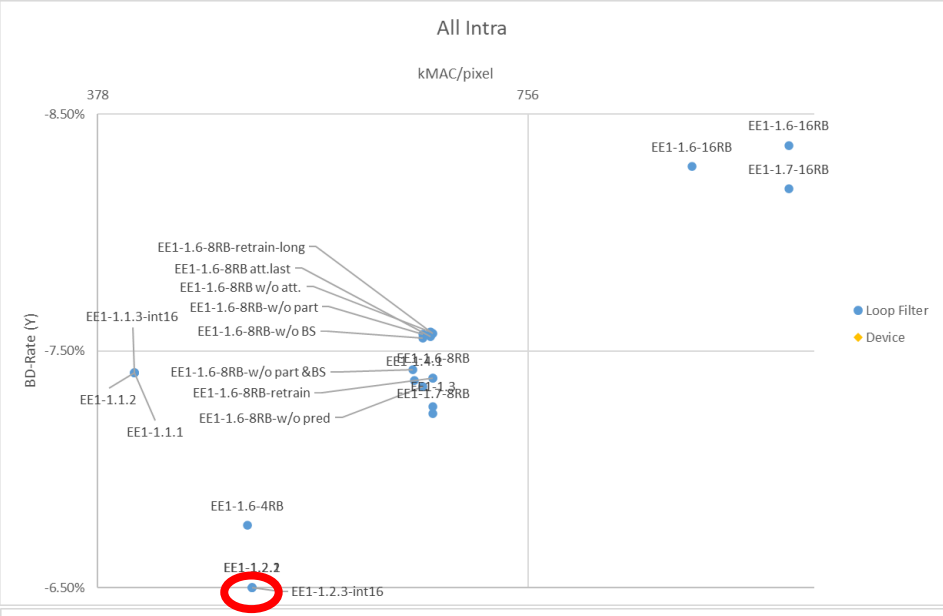
	Random access Main10 BD-rate Over VTM-11.0_nnvc-1.0					
	YUV-PSNR	Y-PSNR	U-PSNR	V-PSNR	EncT	DecT CPU
Class A1	-11.32%	-9.31%	-15.90%	-18.79%	234%	211290%
Class A2	-11.06%	-9.30%	-17.80%	-14.87%	223%	200064%
Class B	-11.06%	-8.30%	-19.03%	-19.66%	237%	215668%
Class C	-11.42%	-8.20%	-20.81%	-21.32%	199%	196621%
Class E						
Overall	**-11.21%**	**-8.67%**	**-18.63%**	**-18.97%**	223%	206430%
Class D	-11.82%	-9.02%	-19.11%	-21.36%	207%	202803%
Class F	-5.78%	-3.76%	-12.04%	-11.62%	326%	94190%

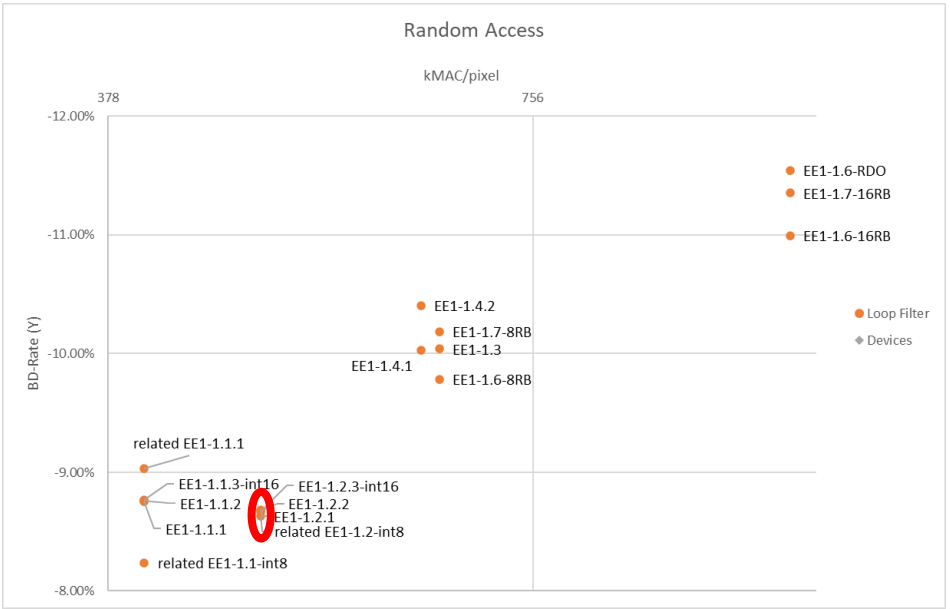
EE1-1.2.3
SADL, int16

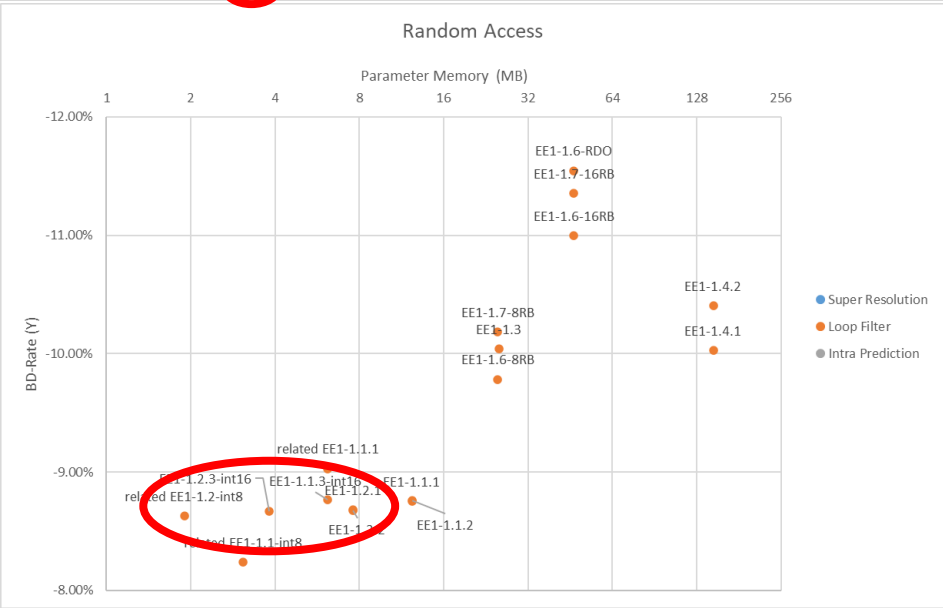
	All Intra Main10 BD-rate Over VTM-11.0_nnvc-1.0					
	YUV-PSNR	Y-PSNR	U-PSNR	V-PSNR	EncT	DecT CPU
Class A1	-8.34%	-6.18%	-13.50%	-16.15%	350%	142206%
Class A2	-7.71%	-5.80%	-14.80%	-12.08%	236%	113635%
Class B	-8.24%	-5.86%	-14.77%	-16.03%	218%	112040%
Class C	-9.06%	-6.43%	-15.96%	-17.95%	170%	83904%
Class E	-10.53%	-8.69%	-15.12%	-17.01%	244%	131151%
Overall	**-8.73%**	**-6.50%**	**-14.89%**	**-15.98%**	231%	112498%
Class D	-8.95%	-6.46%	-15.03%	-17.77%	163%	91392%
Class F	-5.88%	-4.06%	-11.21%	-11.44%	157%	95131%

	Random access Main10 BD-rate Over VTM-11.0_nnvc-1.0					
	YUV-PSNR	Y-PSNR	U-PSNR	V-PSNR	EncT	DecT CPU
Class A1	-11.35%	-9.34%	-16.02%	-18.72%	235%	212333%
Class A2	-11.04%	-9.28%	-17.85%	-14.81%	224%	199960%
Class B	-11.05%	-8.29%	-19.00%	-19.69%	236%	215216%
Class C	-11.42%	-8.19%	-20.79%	-21.40%	198%	195140%
Class E						
Overall	**-11.21%**	**-8.67%**	**-18.65%**	**-18.97%**	222%	206052%
Class D	-11.84%	-9.02%	-19.27%	-21.31%	205%	200519%
Class F	-5.80%	-3.77%	-12.20%	-11.61%	323%	93010%

The trade-off







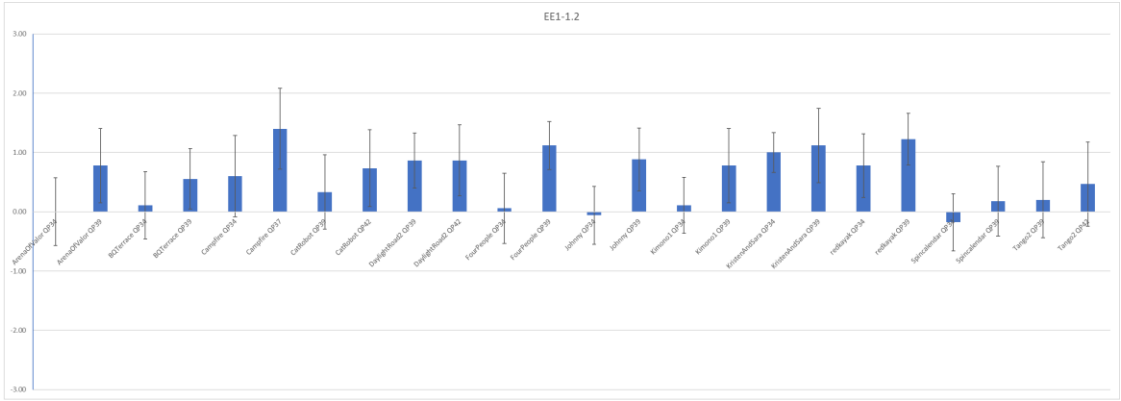
Under RA, similar trade-off between MAC and BD-rate is achieved, but with even less memory size of parameters than EE1-1.1, which is the filter with 2 models.

Compared with other filters except for EE1-1.1, the memory size is serval times less.

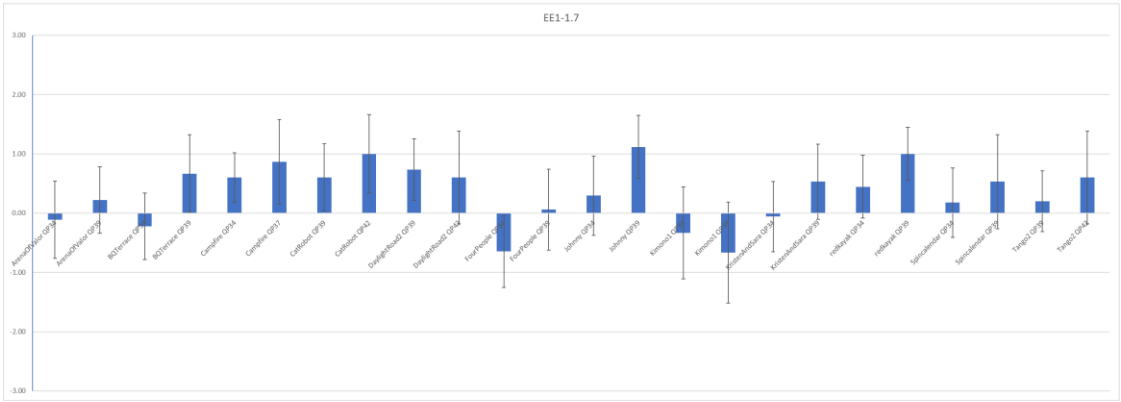
All the data are obtained from the EE1 summary in JVET-Z0023.

The result of subjective test

EE1-1.2 with 1 model



EE1-1.7 with 4 models



For the EE1-2 experiment, a significant visual benefit is reported for 14 out of 24 test cases. For the remaining cases, MOS=0 is included in the confidence interval such that comparable quality is indicated.

In the EE1-1.7 experiment, 9 test cases out of indicate a significant visual benefit while one case showed a significant loss. The remaining 14 cases indicate comparable quality.

It seems that EE-1.2 shows better subjective performance.

All the data are obtained from the REV Result in JVET-Z0053.

Conclusions

- With a constrained memory size and lower complexity, a neural network based in-loop filter is presented in this contribution.
- Up to 11.2% weighted BD-rated savings can be saved with relatively lower MACs and fewer models, where only 1 model is used in all.
- Good trade-off between performance and complexity can be achieved by applying the proposed method.
- The filter with the fewer models and the lower MAC is easier to crosscheck training and do some study, because it costs fewer GPU resource or time to train the model.
- Recommend to adopt the proposed filter as the base software to further explore the NN-based tools.

Thank **OPPO** for crosschecking.

Thanks