

[AHG11] Neural Network-based Adaptive Model Selection for CNN In-Loop Filtering

JVET-X0126

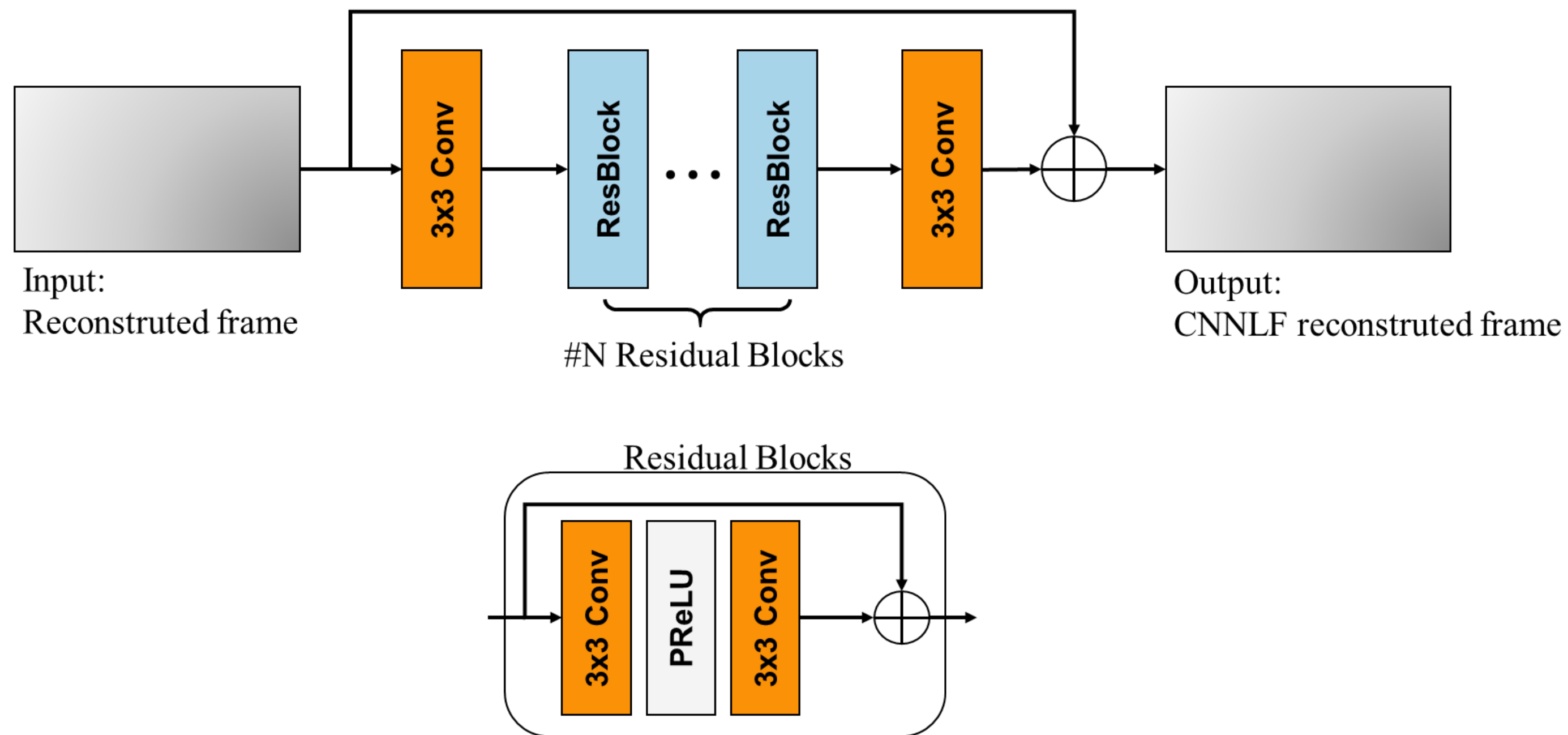
Zhenyu Dai, Yue Yu, Haoping Yu, Kazushi Sato, Luhang Xu, Zhihuang Xie, Dong Wang

Introduction

- A convolutional neural network-based in-loop filter (CNNLF), also comprising multiple models, is introduced.
- The CNNLF is placed between SAO and ALF in the in-loop filter process.
- A flag-free method of neural network-based, adaptive model selection (NNAMS) is developed that can adaptively select at the CTU level one of the CNNLF models for the best coding performance without sending CNNLF model information in the bitstream.

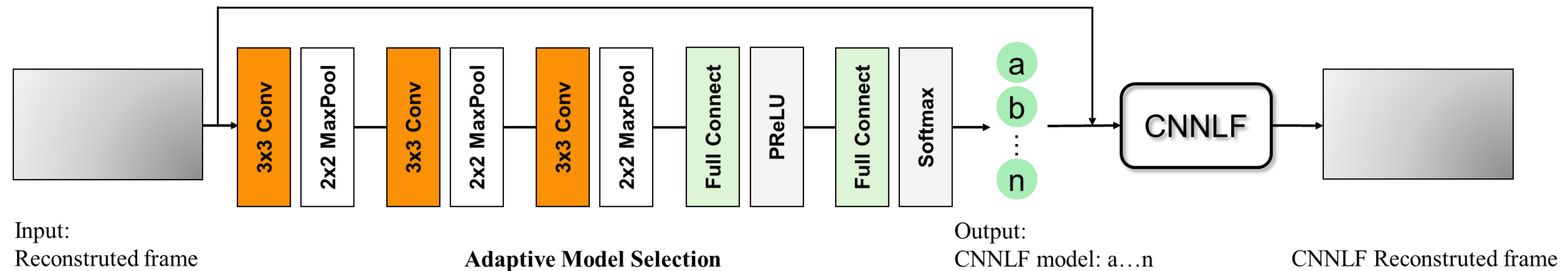
Architecture of CNNLF

- The network structure of the proposed CNNLF.
- The number of ResBlock is set to 10.
- For all the convolutional layers, the number of feature maps is set as 64.
- Different groups of CNNLF models are trained for Y, U, V, respectively.



Architecture of NNAMS

- The network structure of the proposed NNAMS.
- For all the convolutional layers, the number of feature maps is set as 64.
- Different groups of NNAMS models are trained for Y, U, V, respectively.



Neural network-based adaptive model selection (NNAMS)

- In the proposed method, separate CNNLF models are trained with individual CTC QPs{22, 27, 32, 37, 42} for Y, U, V, respectively. Therefore, a candidate list of 5 CNNLF models {modelQP22, modelQP27, modelQP32, modelQP37, modelQP42} can be set up for each color component.
- A neural network-based adaptive model selection method (NNAMS) is proposed to predict the most suitable CNNLF model for the current CTU. The NNAMS method is used in both the encoder and the decoder, there's no need to signal the selected model index in the bitstream.
- After selecting the CNNLF model to be used for the current CTU, the encoder can further decide whether to apply the CNNLF to the current CTU through dedicated flags.
- For the all-intra configuration, the NNAMS method is disabled.

Training and inference information

Network Information in Training Stage		
Mandatory	HW environment:	CPU: Intel(R) Xeon(R) Gold 6142 CPU @ 2.60GHz GPU: Tesla V100 (32GB)
	SW environment:	OS: Ubuntu 16.04.4
	Framework:	Pytorch v1.6.0
	Epoch:	20
	Batch size:	32
	Training time:	48h
	Training data information:	DIV2K
	Configurations for generating compressed training data (if different to VTM CTC):	
Optional		
	Patch size	128X128x1 for Y, 64x64x1 for U or V
	Learning rate:	1e-4
	Optimizer:	ADAM
	Loss function:	L1
	Preprocessing:	
	Other information:	

Network Information in Inference Stage		
Mandatory	HW environment:	
	GPU Type	N/A
	Framework:	Pytorch v1.6.0
	Number of GPUs per Task	0
	Total Conv. Layers	22 for CNNLF, 3 for NNAMS
	Total FC Layers	2 for NNAMS
	Total Parameter Number	CNNLF: 739k NNAMS: 41k(Y), 39k(U or V)
	Parameter Precision	32 (F)
	Memory Parameter (MB)	CNNLF: 3.30MB/model, 15models NNAMS: 273KB/model, 15models
	Memory Temp (MB)	
	Multiplay Accumulate (MAC)/pixel	CNNLF: 740k NNAMS: 9.76k
Optional		
	Patch size	128X128x1 for Y, 64x64x1 for U or V
	Other information:	

Simulation results over VTM-11.0+newMCTF

Test 1: CNNLF only

For the CTC RA configuration, the following model selection method is applied:
if the current coding qp <=22: CNNLF model is modelQP22;
else if the current coding qp <=27: CNNLF model is modelQP27;
else if the current coding qp <=32: CNNLF model is modelQP32;
else if the current coding qp <=37: CNNLF model is modelQP37;
else: CNNLF model is modelQP42;

	Random access Main10										
	BD-rate Over VTM-11.0_nnvc-1.0										
	Y-PSNR	U-PSNR	V-PSNR	Y-MSIM	U-MSIM	V-MSIM	EncT	DecT CPU	DecT GPU	bit DIFF	
Class A1	-1.19%	-4.72%	-5.99%	-0.69%	-3.75%	-4.93%	145%	55989%	#DIV/0!	1%	
Class A2	-1.13%	-4.23%	-3.37%	-0.48%	-3.66%	-2.32%	142%	51493%	#DIV/0!	1%	
Class B	-1.33%	-4.73%	-5.69%	-0.62%	-5.13%	-5.14%	144%	56814%	#DIV/0!	1%	
Class C	-1.50%	-3.97%	-4.95%	-0.63%	-4.21%	-4.71%	133%	50323%	#DIV/0!	1%	
Class E											
Overall	-1.31%	-4.43%	-5.09%	-0.61%	-4.32%	-4.42%	141%	53777%	#DIV/0!	1%	
Class D	-2.29%	-3.56%	-4.47%	-1.07%	-3.35%	-2.65%	138%	31287%	#DIV/0!	0%	
Class F	-0.36%	-3.78%	-4.12%	0.15%	-4.14%	-4.24%	163%	41848%	#DIV/0!	0%	
Class H	#VALUE!	#VALUE!	#VALUE!	#VALUE!	#VALUE!	#VALUE!	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	

For the CTC AI configuration, a fixed CNNLF model, selected based on the encoding QP, is used in the coding of the entire sequence.

	All Intra Main10										
	BD-rate Over VTM-11.0_nnvc-1.0										
	Y-PSNR	U-PSNR	V-PSNR	Y-MSIM	U-MSIM	V-MSIM	EncT	DecT CPU	DecT GPU	bit DIFF	
Class A1	-2.24%	-4.14%	-6.15%	-1.95%	-3.70%	-4.86%	169%	34652%	#DIV/0!	0%	
Class A2	-2.57%	-3.87%	-4.23%	-2.36%	-3.33%	-3.01%	138%	25805%	#DIV/0!	0%	
Class B	-2.62%	-3.91%	-5.57%	-2.12%	-4.03%	-4.75%	135%	28627%	#DIV/0!	0%	
Class C	-3.82%	-5.40%	-7.13%	-2.74%	-5.33%	-6.82%	120%	21496%	#DIV/0!	0%	
Class E	-4.91%	-6.15%	-6.60%	-3.82%	-5.15%	-6.65%	141%	29431%	#DIV/0!	0%	
Overall	-3.20%	-4.65%	-5.96%	-2.55%	-4.33%	-5.26%	138%	27381%	#DIV/0!	0%	
Class D	-4.39%	-4.24%	-6.46%	-2.83%	-4.24%	-6.00%	116%	20846%	#DIV/0!	0%	
Class F	-1.82%	-4.88%	-5.75%	-1.19%	-4.98%	-5.58%	117%	24211%	#DIV/0!	0%	
Class H	#VALUE!	#VALUE!	#VALUE!	#VALUE!	#VALUE!	#VALUE!	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	

Test 2: CNNLF + NNAMS with CTC RA configuration

the NNAMS method is applied to every frame, and both the encoder and the decoder use the same CNNLF model selected by NNAMS at the CTU level whenever the CNNLF flag is set to true in the bitstream.

	Random access Main10										
	BD-rate Over VTM-11.0_nnvc-1.0										
	Y-PSNR	U-PSNR	V-PSNR	Y-MSIM	U-MSIM	V-MSIM	EncT	DecT CPU	DecT GPU	bit DIFF	
Class A1	-1.39%	-5.13%	-6.58%	-0.93%	-4.13%	-5.43%	141%	60634%	#DIV/0!	1%	
Class A2	-1.49%	-4.81%	-3.89%	-0.76%	-4.17%	-2.68%	137%	55431%	#DIV/0!	1%	
Class B	-1.78%	-5.50%	-6.56%	-0.94%	-5.79%	-5.81%	145%	61700%	#DIV/0!	1%	
Class C	-2.23%	-5.36%	-6.24%	-1.34%	-5.24%	-5.64%	132%	57148%	#DIV/0!	1%	
Class E											
Overall	-1.77%	-5.25%	-5.94%	-1.01%	-4.99%	-5.06%	139%	58964%	#DIV/0!	1%	
Class D	-3.19%	-4.85%	-5.92%	-1.67%	-4.57%	-3.82%	137%	40594%	#DIV/0!	1%	
Class F	-0.72%	-4.73%	-4.88%	-0.18%	-4.97%	-4.76%	171%	50340%	#DIV/0!	0%	
Class H	#VALUE!	#VALUE!	#VALUE!	#VALUE!	#VALUE!	#VALUE!	#DIV/0!	#DIV/0!	#DIV/0!	#DIV/0!	

Conclusion

- This contribution presents a convolutional neural network-based in-loop filter (CNNLF) wherein a neural network-based, adaptive model selection (NNAMS) method is introduced.
- When working alone, the proposed CNNLF shows interesting coding gains compared to the VTM-11.0 + new MCTF. It is observed that the coding gains can be further increased when using the proposed CNNLF together with the proposed NNAMS method.
- It is proposed to further study neural network-based adaptive model selection for NN-based in-loop filters in an Ad-Hoc group.

Thank you

oppo