

MEDIATEK

CE13-1.1: Convolutional neural network loop filter

Yu-Ling Hsiao, Olena Chubach, Ching-Yeh Chen, Tzu-Der Chuang,
Chih-Wei Hsu, Yu-Wen Huang, Shaw-Min Lei

Presented by Olena Chubach
14th Meeting: Geneva, CH,
19–27 March 2019

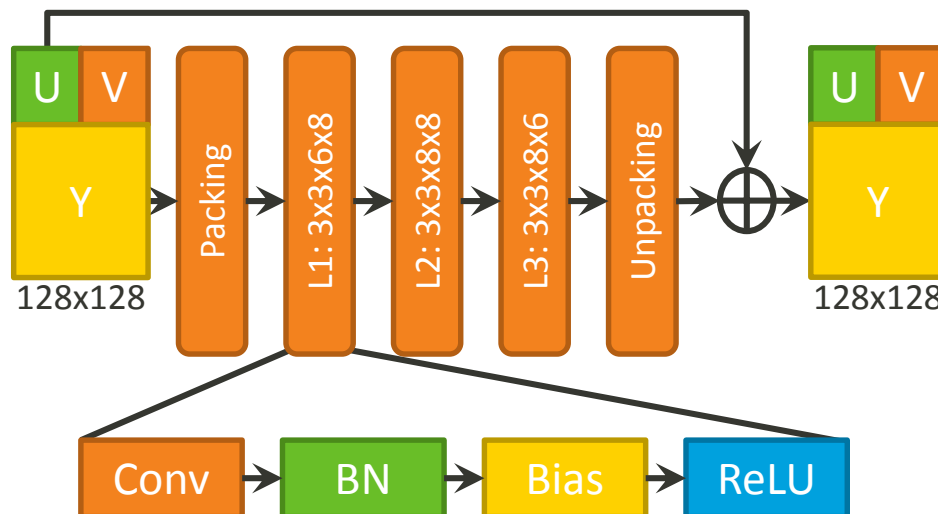
Overall Summary

- Convolutional neural network loop filter (CNNLF) used as an additional in-loop filtering stage
 - One 3-layer CNN shared by luma and chroma
 - The CNN inputs are ALF output samples
 - The outputs of CNN are the differences between uncompressed and reconstructed samples
 - Sum of CNN output and ALF output samples is CNNLF output
- Multi-level on/off control (picture, CTB, and 32x32 block levels) for Y, Cb, and Cr components
 - Y picture/CTB off enforce Cb&Cr picture/CTB off
- CNN parameters signaled at the I-picture of each random access segment (RAS, roughly 1 sec in RA CTC)
 - Trained from pictures of temporal levels 0 and 1 in this RAS
 - Do not signal CNN parameters when the CNNLF is off at the I-picture and assign all remaining pictures in this RAS as CNNLF-off

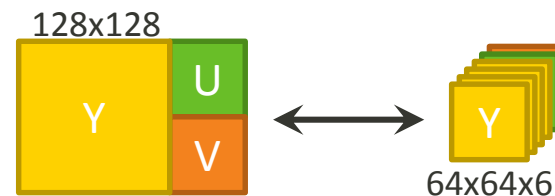
		Y	U	V	Enc. T	Dec. T
Proposed Method	RA	-1.36%	-14.96%	-14.91%	100%	142%

CNNLF Structure

- **LK: M x N x P x Q**
 - **K** is layer id
 - **M x N** is spatial filter shape
 - **P** is number of input channels
 - **Q** is number of output channels
- Operations in each layer:
 - convolution
 - batch normalization
 - adding bias
 - rectified linear unit (ReLU)
 - No ReLU is applied for the last layer



- Packing
 - 2x2 sub-sampling to form 4 sub-pictures (channels)
 - 128x128 luma samples and 64x64x2 chroma samples converted to 64x64x6
- Unpacking
 - First four channels are combined for the luma CTB
 - Last two channels correspond to the two chroma CTBs



Inference Stage Information

Total conv. layers	3
Total FC layers	0
Framework	NA
Param. num	1506
Weight param. num	1440 (6-bit)
Scale, shift, bias param. num	66 (32-bit)
Mem.P (MB)	0.001282 (1344 Bytes)
Mem.T (MB)	0.344 (128x128, CTU based)
Test GPU	NA
Test CPU	Xeon E5-2643 v3
OS	Red Hat 6.7
Fix-point operation	Yes

Simulation Results

- No GPU and no dependency on any other frameworks (e.g., TensorFlow, PyTorch) for encoding and decoding
- Run time of training by GPU and the first encoding pass are not included in EncT, since they are encoder-only issues
- Anchor: VTM4.0

		Y	U	V	Enc. T	Dec. T
Proposed Method	RA	-1.36%	-14.96%	-14.91%	100%	142%

	Random Access Main 10				
	Over VTM-4.0				
	Y	U	V	EncT	DecT
Class A1	-2.95%	-8.57%	-13.33%	100%	189%
Class A2	-1.47%	-18.33%	-15.72%	100%	130%
Class B	-1.52%	-23.99%	-21.70%	100%	148%
Class C	0.12%	-5.94%	-6.99%	99%	116%
Class E					
Overall	-1.36%	-14.96%	-14.91%	100%	142%

Conclusions

- For UHD&FHD sequences, the reported luma BD-rate saving is about 3% for VTM-4.0-RA if coding gains are balanced with around 55% decoding time increase
- For sequences with resolution smaller than HD coding of CNN side information should be improved to achieve better coding gains

		Y	U	V	Enc. T	Dec. T
Proposed Method	RA	-1.36%	-14.96%	-14.91%	100%	142%



everyday genius

Training Stage Settings

Learning rate	0.01
Optimizer	ADAM
Batch size	All patches
Epoch	10000
Loss function	L2 (Y:U:V=8:1:1)
Training GPU	GTX 1080ti
Training time	
For Class A	2.716 hours per network (one RAS)
For Class B	1.299 hours per network (one RAS)
For Class C	0.243 hours per network (one RAS)
Framework	TensorFlow 1.8.0
Patch size	128x128, grid cropped
Dataset	CTC sequences
QP set for generating training dataset	22, 27, 32, 37

