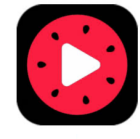
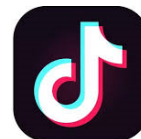


JVET-U0068

AHG11: CONVOLUTIONAL NEURAL NETWORKS-BASED IN-LOOP FILTER WITH ADAPTIVE MODEL SELECTION

Yue Li, Li Zhang, Kai Zhang



Overview

- This contribution presents a convolutional neural network-based in-loop filtering method.
- The proposed technique enables model selection from a candidate model list including three models for each slice and each CTU
- To evaluate the performance, the proposed CNN-based in loop filter is tested on top of VTM-9.0
 - *Under AI, on average 8.33%, 23.11%, and 23.55% BD-rate reductions for Y, Cb, and Cr*
 - *Under RA, on average 10.28%, 28.22%, and 27.97% BD-rate reductions for Y, Cb, and Cr*
 - *Under LDB, on average 9.46%, 23.74%, and 23.09% BD-rate reductions for Y, Cb, and Cr*

Network Architecture

■ Features

- *Stride = 2 for the first layer*
- *Kernel size 3x3 for all layers*
- *#feature maps is 128 for internal convolutional layers*
- *PReLU is used as the activation function.*
- *Different groups of models are trained for I slice and B slice, respectively.*
 - For I slices, prediction and partition information are fed into the network.
 - For B slices, prediction is fed into the network

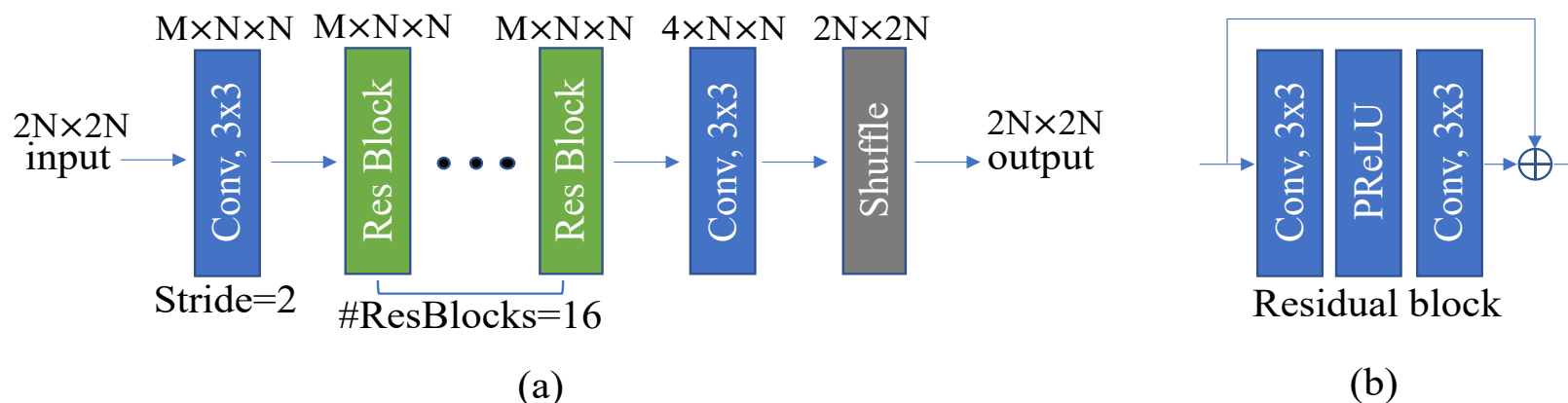
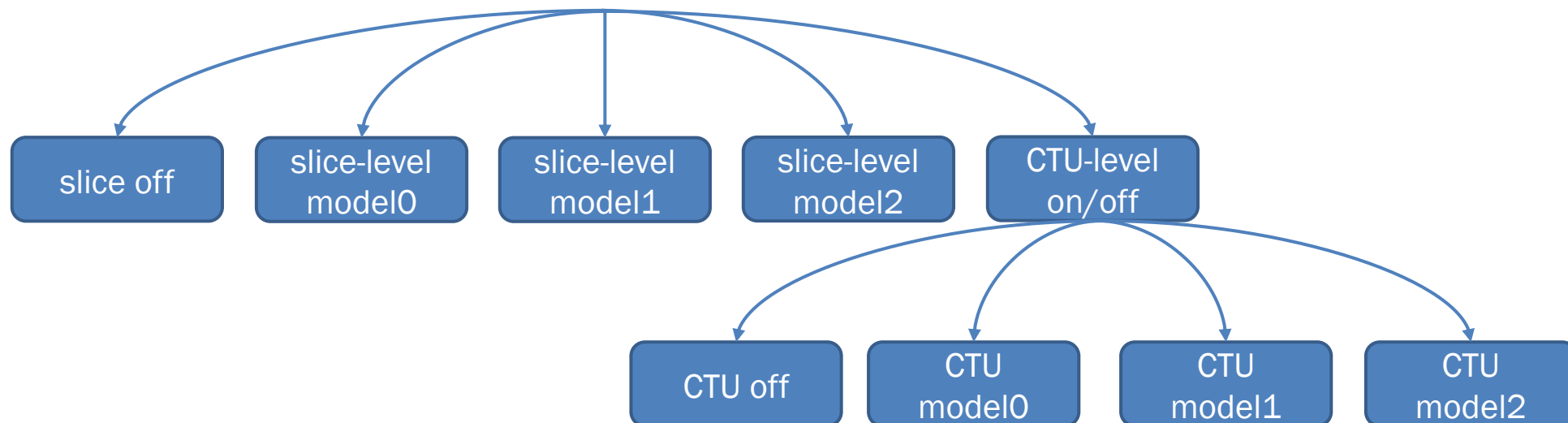


Figure 1. (a) Architecture of the proposed CNN filter (b) Construction of Residual Block in (a)

Model Selection

- Candidate model list includes models trained with QPs equal to $\{q, q-5, q-10\}$



Inference

- LibTorch is used for performing the inference of the proposed CNN filters in VTM
- Deblocking filter and SAO are disable while ALF (and CC-ALF) is placed after the CNN-based filter.
- Different CNN models are trained to adapt to different QP points and different slices

Network Information in Inference Stage		
Mandatory	HW environment:	
	GPU Type	N/A
	Framework:	PyTorch v1.6
	Number of GPUs per Task	0
	Total Parameter Number	4.91M/model
	Parameter Precision (Bits)	32 (F)
	Memory Parameter (MB)	~20M/model, 24 models in total
	MAC (Giga)	10404 (Input: 3840×2160)
Optional		
	Total Conv. Layers	36
	Total FC Layers	0
	Total Memory (MB)	
	Batch size:	1
	Patch size	128×128
	Changes to network configuration or weights required to generate rate points	
	Peak Memory Usage	
	Other information:	

Table 1. Network information in inference stage

Training

- PyTorch is used as the training platform
- Training dataset
 - *DIV2K and BVI-DVC are adopted to train the CNN filters of I slices and B slices, respectively*

Network Information in Training Stage		
Mandatory	GPU Type	GPU: Tesla-V100-SXM2-32GB
	Framework:	PyTorch v1.6
	Number of GPUs per Task	2
	Epoch:	90
	Batch size:	64
	Training time:	40h/model
	Training data information:	DIV2K, BVI-DVC
Optional	Training configurations for generating compressed training data (if different to VTM CTC):	VTM-9.0, QP {17, 22, 27, 32, 37, 42}
	Number of iterations	
	Patch size	128×128
	Learning rate:	1e-4
	Optimizer:	ADAM
	Loss function:	
	Preprocessing:	
	Other information:	

Table 2. Network information in training stage

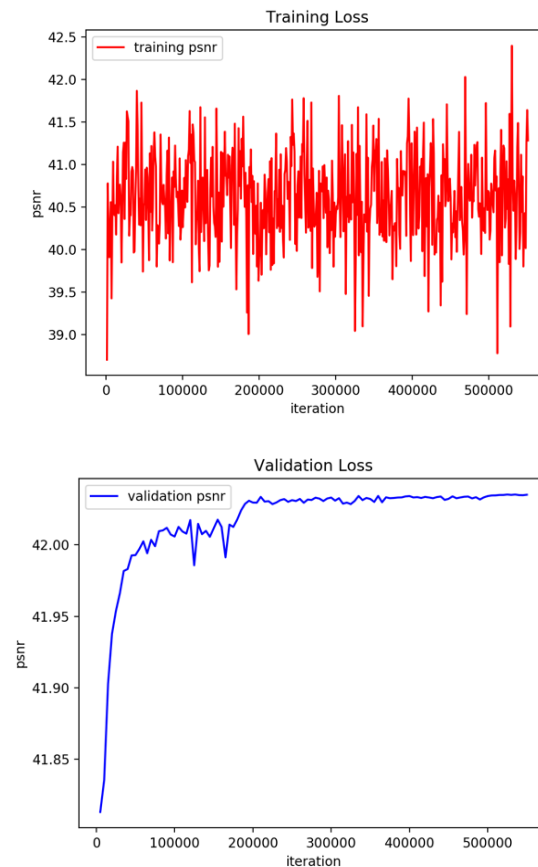


Figure 2. Training and validation loss curved for the qp27 model of inter slices

Experimental Results

■ Test results on top of VTM-9.0

- 8.33%, 23.11%, and 23.55% BD-rate reductions on average for Y, Cb, and Cr, under AI
- 10.28%, 28.22%, 27.97% BD-rate reductions on average for Y, Cb, and Cr, under RA
- 9.46%, 23.74%, 23.09% BD-rate reductions on average for Y, Cb, and Cr, under LDB

	AI					RA					LDB				
	Y	U	V	EncT	DecT	Y	U	V	EncT	DecT	Y	U	V	EncT	DecT
Class A1	-7.04%	-19.72%	-20.94%	423%	14896%	-9.25%	-22.08%	-22.88%	248%	21338%					
Class A2	-7.32%	-24.01%	-22.73%	268%	13380%	-11.20%	-29.17%	-28.76%	238%	19116%					
Class B	-7.48%	-24.24%	-24.06%	240%	13606%	-9.79%	-31.05%	-29.57%	248%	23968%	-8.69%	-24.51%	-25.26%	235%	10572%
Class C	-8.90%	-22.61%	-25.08%	176%	10061%	-10.97%	-28.59%	-29.18%	196%	21502%	-10.21%	-24.50%	-24.60%	189%	11145%
Class E	-11.30%	-24.37%	-24.11%	274%	14814%						-9.75%	-21.46%	-17.47%	455%	8730%
Overall	-8.33%	-23.11%	-23.55%	257%	13065%	-10.28%	-28.22%	-27.97%	231%	21743%	-9.46%	-23.74%	-23.09%	258%	10257%
Class D	-8.75%	-22.68%	-24.96%	158%	10571%	-12.17%	-28.99%	-30.27%	187%	20512%	-11.56%	-26.59%	-27.98%	182%	12071%
Class F	-5.03%	-15.94%	-15.38%	170%	9364%	-5.27%	-16.93%	-16.58%	337%	9883%	-5.05%	-15.83%	-15.41%	332%	5723%

	AI					RA				
	Y	U	V	EncT	DecT	Y	U	V	EncT	DecT
Class A1	-7.28%	-12.35%	-3.96%	140%	19772%	-2.78%	-7.23%	-6.78%	118%	1132%
Class A2	-7.34%	-17.24%	-15.76%	122%	14960%	-5.05%	-14.46%	-14.69%	113%	956%
Class B	-7.07%	-16.77%	-12.85%	119%	15317%	-3.91%	-16.20%	-12.94%	114%	1029%
Class C	-9.02%	-18.98%	-21.04%	107%	12166%	-3.73%	-13.81%	-12.65%	110%	1055%
Class E	-11.04%	-11.48%	-16.78%	122%	19438%					
Overall	-8.24%	-15.72%	-14.33%	121%	15738%	-3.86%	-13.42%	-11.98%	113%	1040%
Class D	-8.84%	-21.17%	-23.89%	103%	10718%	-3.80%	-15.72%	-14.30%	111%	1106%

Table 3 (above). Performance of the proposed technique

Table 4 (left) Performance of our prior contribution JVET-T0088

Conclusions

- This contribution presents a CNN-based model for the in-loop filtering.
- The proposed CNN-based filtering with adaptive model selection method shows very promising coding gains.
- It is proposed to further study CNN-based in-loop filters in the Ad-Hoc group.

Thanks!