Zergriss : 상품 구별을 위한 효율적인 지능형 모바일 시스템

LINC+ 4차 산업혁명
혁신선도대학 사업 산업박람회

Zergriss : An energy efficient intelligent mobile system for recognize merchandise

Researcher 최근석, 전자공학과 (ieggiegg@ajou.ac.kr)
홍석민, 전자공학과 (smhong1123@ajou.ac.kr)

Professor 선우명훈, 전자공학과 (sunwoo@ajou.ac.kr)

ABSTRACT

Merchandise Recognition(상품 식별)은 가사/산업용 로봇, 자율주행 자동차 등 다양한 분야에서 필요한 기술이다. 현재 있는 DNN(Deep Neural Network)은 모바일 시스템에서 상품 식별이 어렵다. 기존 DNN system은 높은 accuracy를 갖으나 runtime, area, power가 크다. 따라서 모바일에서 효율적인 DNN accelerator와 algorithm 개발하였다.

Bottleneck layer, Additional layer, ImageDataGenerator 등으로 이루어진 DNN algorithm을 개발하였다. 이를 통해 run-time, data set/size 감소시켰고 accuracy를 높였다.

Energy efficient DNN Accelerator인 zergriss를 개발하였다. ZAC(Zero Aware compression), ZSPE(Zero Skip Processing Element), ZOD(Zero Optimized Dataflow)를 통해 zero를 skip하여 MAC operation을 줄이고 data reuse rate를 높여 DRAM access를 줄였다.

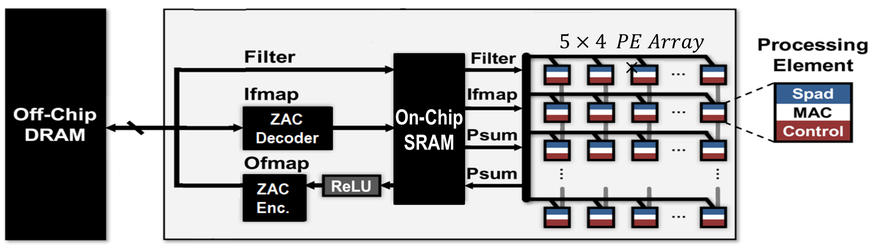
OBJECTIVES

1. Software 면에서 상품 구별을 위해 모바일에서 효율적인 DNN algorithm을 개발한다.
   - Data를 limit하고 accuracy를 유지하면서 data set/size, run-time이 감소시킨다.
2. Hardware 면에서 상품 구별을 위해 모바일에서 효율적인 DNN accelerator를 개발한다.
   - DRAM access와 MAC operation을 줄인다.

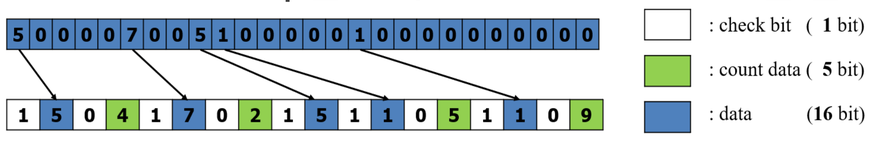
METHODOLOGY

1. Bottleneck layer, Additional layer, ImageDataGenerator 등으로 이루어진 DNN algorithm을 개발하였다.
   - python을 이용하여 VGG16, ResNet, MobileNet, InceptionV3 등의 model로 효율을 검증한다.
   - Snack picture10개 class, 1200 samples로 batch size 16/48, epoch 500, 10GB GPU로 학습/평가한다.

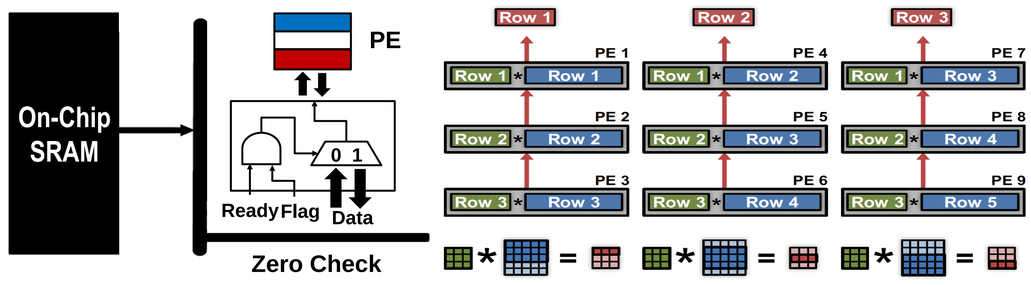
2. ZAC, ZSPE, ZOD 등으로 이루어진 DNN accelerator로 Zergriss를 개발하였다.
   - 구조는 DRAM, SRAM, PE, Encoder, Decoder이다.
   - 1차 : python으로 DNN accelerator를 Lenet-5에서 검증한다.
   - 2차 : verilog으로 작성 후 modelsim으로 검증한다.
   - 3차 : vivado로 FPGA에서 검증한다.



- ZAC : OFMap을 압축하여 DRAM access를 줄인다.

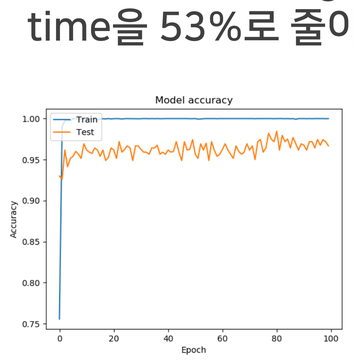


- ZSPE : IFMap(Image Feature Map)와 Filter data 중 Zero를 aware/skip하여 MAC operation을 줄인다.
- ZOD : RS dataflow로 IFMap과 filter를 reuse하여 DRAM access를 줄인다.

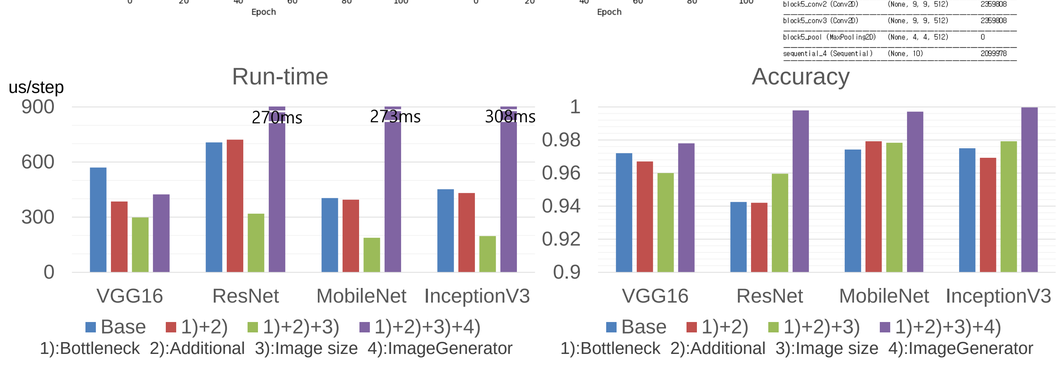


RESULTS

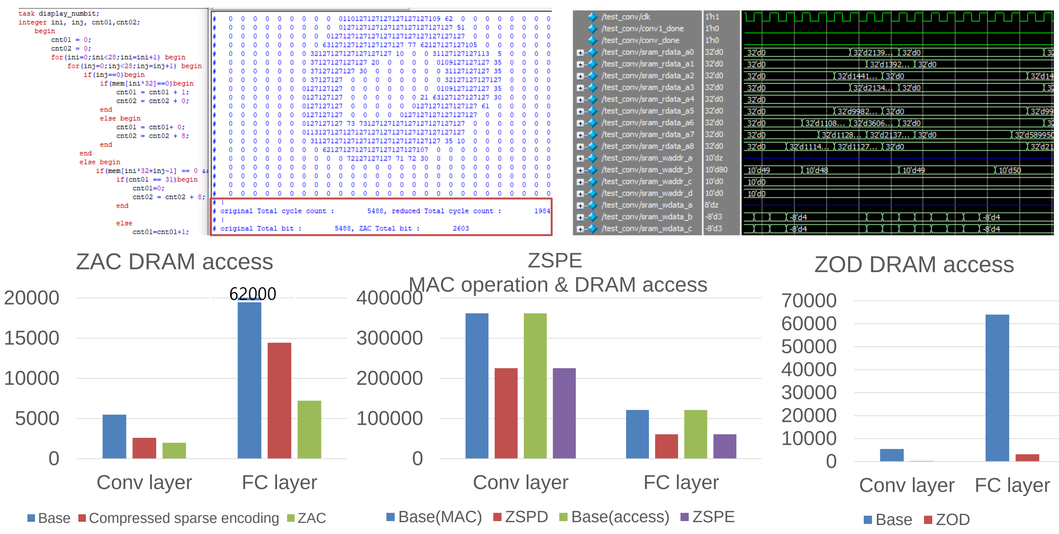
1. 한 상품의 training data를 100개로 limit하고 run time을 53%로 줄이고 accuracy를 2.8% 향상시켰다.



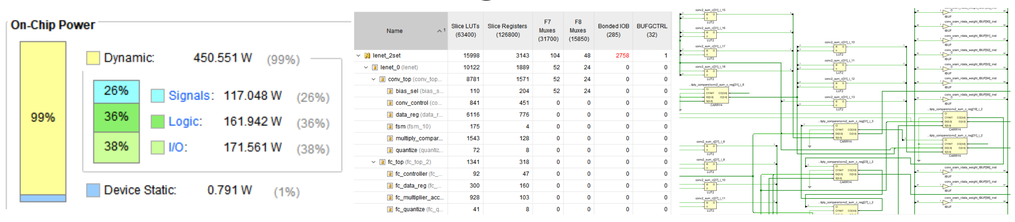
Layer (type)	Output Shape	Param #
input_4 (InputLayer)	(None, 150, 150, 3)	0
block1_conv1 (Conv2D)	(None, 150, 150, 64)	1792
block1_conv2 (Conv2D)	(None, 150, 150, 64)	36928
block1_pool (MaxPooling2D)	(None, 75, 75, 64)	0
block2_conv1 (Conv2D)	(None, 75, 75, 128)	73856
block2_conv2 (Conv2D)	(None, 75, 75, 128)	147584
block2_pool (MaxPooling2D)	(None, 37, 37, 128)	0
block3_conv1 (Conv2D)	(None, 37, 37, 256)	295168
block3_conv2 (Conv2D)	(None, 37, 37, 256)	590080
block3_conv3 (Conv2D)	(None, 37, 37, 256)	590080
block3_pool (MaxPooling2D)	(None, 18, 18, 256)	0
block4_conv1 (Conv2D)	(None, 18, 18, 512)	1180160
block4_conv2 (Conv2D)	(None, 18, 18, 512)	2359808
block4_conv3 (Conv2D)	(None, 18, 18, 512)	2359808
block4_pool (MaxPooling2D)	(None, 9, 9, 512)	0
block5_conv1 (Conv2D)	(None, 9, 9, 512)	2359808
block5_conv2 (Conv2D)	(None, 9, 9, 512)	2359808
block5_conv3 (Conv2D)	(None, 9, 9, 512)	2359808
block5_pool (MaxPooling2D)	(None, 4, 4, 512)	0
sequential_4 (Sequential)	(None, 10)	2099978



2. Modelsim으로 Lenet5에서 bit, cycle을 비교하였다. -
   - ZAC로 DRAM access를 기존보다 76% 줄였다.
   - ZSPE : MAC operation을 40.8%, DRAM access를 40.7% 줄였다.
   - ZOD : DRAM access를 Lenet5 대비 95% 줄 것이다.
   (testbench에서 ZOD 사용 시 reuse 횟수를 통해 예측한 값 / 미 구현)



3. Vivado로 board i/o pins의 한계로 synthesis까지 진행하여 Lenet5에서 zergriss 효율을 검증하였다.



Name	Slice LUTs (63400)	Slice Registers (126800)	F7 Muxes (31700)	F8 Muxes (15850)	Bonded IOB (285)	BUFGCTRL (32)
lenet_2set	15998	3143	104	48	2758	1
lenet_0 (lenet)	10122	1889	52	24	0	0
conv_top (conv_top...	8781	1571	52	24	0	0
bias_sel (bias_s...	110	204	52	24	0	0
conv_control (co...	841	451	0	0	0	0
data_reg (data_r...	6116	776	0	0	0	0
fsm (fsm_10)	175	4	0	0	0	0
multiply_compar...	1543	128	0	0	0	0
quantize (quantiz...	72	8	0	0	0	0
fc_top (fc_top_2)	1341	318	0	0	0	0
fc_controller (fc_...	92	47	0	0	0	0
fc_data_reg (fc_...	300	160	0	0	0	0
fc_multiplier_acc...	928	103	0	0	0	0
fc_quantize (fc_q...	41	8	0	0	0	0

CONCLUSIONS

개발된 DNN algorithm은 기존보다 적은 data에서 run-time은 53% 줄였고 accuracy는 2.8% 향상시켰다.

개발된 DNN accelerator은 기존보다 DRAM access를 85.8%, MAC operation을 40.8% 줄였다.

추후 algorithm으로는 pruning, skipping 관련 기술과 ZOD 구현하여 다른 모델을 FPGA에서 추가 검증한다.