J. Internet Comput. Serv.
ISSN 1598-0170 (Print) / ISSN 2287-1136 (Online)
http://www.jics.or.kr

내장형 인공지능 프로세서를 위한 성능 분석기☆

Performance Analyzer for Embedded AI Processor

황 동 현[1] 윤 영 현[1] 한 창 엽[1] 이 승 은[1*]
Dong Hyun Hwang Young Hyun Yoon Chang Yeop Han Seung Eun Lee

요 약

최근 인공지능에 대한 관심이 높아짐에 따라 인공지능 프로세서를 하드웨어로 구현하는 연구가 활발히 진행되고 있다. 하지만 인공지능 프로세서는 기존에 기능 검증을 위한 프로세서 시뮬레이션 외에 애플리케이션 단계에서 인공지능 프로세서가 해당 애플리케이션에 적합한지에 대한 성능 검증이 추가로 필요하다. 본 논문에서는 인공지능 프로세서를 활용한 애플리케이션 성능 검증과 프로세서의 한계점을 탐색할 수 있는 내장형 인공지능 프로세서를 위한 성능 분석기를 제안한다. 본 논문은 내장형 인공지능 프로세서를 위한 성능 분석기를 구현하기 위하여 기존에 구현된 인공지능 프로세서의 구조를 분석하고 이를 기반으로 인공지능 프로세서를 모사하는 내장형 인공지능 프로세서를 위한 성능 분석기를 구현한다. 내장형 인공지능 프로세서를 위한 성능 분석기를 활용해 이미지 인식, 음성 인식 애플리케이션에서 인공지능 프로세서의 성능 분석 및 한계점을 탐색하고, 제한된 메모리 크기 안에서 인공지능 프로세서의 구조를 최적화한다.

☞ 주제어 : 인공지능 프로세서, 시뮬레이터, 인공지능, 임베디드 시스템, SoC

ABSTRACT

Recently, as interest in artificial intelligence has increased, many studies have been conducted to implement AI processors. However, the AI processor requires functional verification as well as performance verification on whether the AI processor is suitable for the application. In this paper, We propose an AI processor performance analyzer that can verify the application performance and explore the limitations of the processor. By Using the performance analyzer, we explore the limitations of the AI processor and optimize the AI model to fit an AI processor in image recognition and speech recognition applications.

☞ keyword : AI processor, Simulator, Artificial intelligence, Embedded system, SoC

1. 서 론

인공지능 알고리즘은 기존의 영상, 음성, 자연어 처리 분야에서 풀기 어려웠던 객체 인식, 음성인식, 언어 모델링 등의 문제들을 해결할 수 있음을 실험적으로 증명하였다. 대표적으로 음성 및 이미지를 분류하는 분류모델, 영상에서 움직이는 객체들을 추적하며 사람의 수를 세는 객체 탐지 및 추적 모델, 이미지에서 그림자를 제거해주는 생성적 적대 모델(generative adversarial network) 등이 있다[1-3]. 최근 실험적으로 증명된 인공지능 알고리즘들을 임베디드, 모바일 디바이스 등에 적용하려는 경량화 연구가 제안되었다[4-13]. 인공지능 경량화 알고리즘은 크게 1). 지식 증류(knowledge distillation), 2). 모델 아키텍처 개선, 3). 비트 양자화(bit quantization), 4). 프루닝(pruning)으로 총 네 가지로 나뉜다.

스튜던트-티처(student-teacher) 모델로 불리기도 하는 지식 증류 기법은 기존에 잘 학습되어, 성능이 좋은 인공지능 모델을 티처, 학습이 안 되어있는 인공지능 모델을 스튜던트로 정의한다. 지식 증류 기법은 학습 시, 입력 데이터에 대한 티처와 스튜던트 모델의 softmax 출력 값의 그래디언트를 합하여 스튜던트 모델에게 전달한다. 티처와 스튜던트 모델의 그래디언트 합은 뛰어난 티처 모델에서 중요한 지식만 잘 정제하여 스튜던트 모델로 전달하는 방식이다. 이를 통해 정제된 지식을 전달받은 스튜던트 모델의 성능은 티처 모델의 성능에 근사하게 된다. 지식 증류 기법은 스튜던트 모델이 적은 파라미터를 갖더라도 티처 모델의 정제된 지식을 전달받기 때문에 티처 모델에 근사하는 높은 성능을 유지할 수 있다는 장점이 있다[4].

[1] Dept. of Electronic Engineering, Seoul National University of Science and Technology, Seoul, 01811, Korea
* Corresponding author: (seung.lee@seoultech.ac.kr)
[Received 1 July 2020, Reviewed 12 July 2020(R2 31 August 2020), Accepted 25 September 2020]
☆ This work is supported by Seoul National University of Science and Technology.

모델 아키텍처 개선은 채널 방향 연산과 공간 방향 연산의 합으로 구성된 기존의 convolution 연산을 채널 방향의 연산인 pointwise convolution과 공간 방향의 연산인 depthwise convolution으로 나누어 기존의 convolution 연산보다 연산 복잡도를 줄여 추론 시간을 단축한다[5-8].

비트 양자화는 16, 32, 64bit 부동 소수점으로 표현되는 인공지능 모델 파라미터를 2, 3, 4, 6, 8bit 정수로 양자화 하며, 하드웨어 친화적인 연산을 사용하는 방법론이다. 비트 양자화는 양자화 방법에 따라 성능과 추론 시간에 대한 trade-off가 있다. 6, 8bit 양자화는 모델의 성능과 추론 시간에 대한 이득을 균형 있게 선택하는 방법론이며, 2, 3, 4bit 양자화는 성능을 희생하고, 추론 시간을 가속하는 방법론이다[9-12].

프루닝은 학습된 인공지능 모델에서 중요도가 낮은 뉴런을 제외하는 방식으로 작동하며, 약 90% 이상의 뉴런을 모델에서 제외하는 방식으로 파라미터 수를 감소시켜 전체 연산 수를 줄인다[13].

이러한 인공지능 경량화 알고리즘들은 애플리케이션이 요구하는 최소 지연시간을 Intel 프로세서와 같이 고성능 프로세서에서는 충족하지만, 자원이 부족한 임베디드 프로세서에서는 최소 지연시간을 충족시키지 못한다는 한계점이 있다.

이러한 한계점을 극복하기 위해 경량화된 인공지능 알고리즘을 효율적으로 적용하기 위한 기법 및 인공지능 알고리즘을 프로세서로 구현한 연구들이 제안되었다[14-18].

Woong Choi et al.은 binary neural networks에서 0, 1로 양자화된 특징맵의 합이 pop-count 연산으로 이루어진다는 점에 착안하여 pop-count 연산을 CAM(content addressable memory)으로 구현한 프로세서를 제안하고, 이를 통한 에너지 효율성을 입증하였다[14].

Michaela et al.은 비트 양자화 경량화 기법 중 binary neural networks 프레임워크인 FINN-R(second generation of fast, scalable quantized neural network inference on FPGAs)을 제안하였다. FINN-R은 1). 모델 컴포넌트 제공 2). 모델 소스 코드를 FPGA(field programmable gate array)에서 실행 가능한 코드로 생성해주는 컴파일러 제공 3). roofline 모델에 기반하는 성능 분석기를 제공한다. FINN-R은 본 논문의 목적과 가장 유사한 접근이나 적용 범위가 binary neural networks에 한정되어 있으며 Xilinx platform에서만 사용할 수 있다[15].

Maohua zhu et al.는 프루닝 기법에서 워크로드 불균형으로 인해 압축률이 높은 희소행렬의 연산 비효율성을 개선하기 위해서 sparse tensor core를 제안하였다. Sparse tensor core는 연산효율이 좋지 않은 희소행렬을 밀도행렬로 변환시키기 위해 generic sparsifying된 가중치의 희소행렬을 밀도행렬과 밀도행렬을 역변환하기 위한 인덱스 행렬로 변환하는 vector-wise sparse matrix encoding을 포함한다. 이를 통해 압축률 80%에서 프루닝을 적용하지 않은 모델 대비 정확도는 같게 유지하였으며, 연산 속도는 2.57배 향상되었다. 하지만 sparse tensor core는 프로세서가 아닌 NVIDIA GPU의 instruction set으로 구현되었다[16].

Weizhe Hua et al.은 중요도가 낮은 뉴런을 제거하는 관점의 기존 프루닝 개념에서 벗어나 특징맵에서 유의미한 특징 벡터만 추출하는 기법인 channel gating을 수행하는 프로세서를 제안하였다[17]. Channel gating은 이전 레이어에서 특징맵이 입력으로 들어오면 gate function을 통해 출력에 반영할 특징맵과 아닌 특징맵을 분리한다. 기존에 전체 출력 특징맵이 사용되었던 것과 달리 유의미한 특징맵만 사용되므로 메모리 접근을 최소화하여 연산을 가속할 수 있다. 프로세서 구조는 기존의 희소 행렬 연산이 가능한 CNN(convolutional neural network) 연산 프로세서 구조에서 채널을 분리하기 위한 gate function이 추가된 구조이다. channel gating 프로세서는 정확도는 유지하면서 메모리 공간과 연산 시간을 획기적으로 줄였으나, 이를 달성하기 위한 프로세서의 구현 복잡도가 증가한다는 trade-off가 있다.

Su Yeon Jang et al. 은 KNN(k-nearest neighbors)과 RBF(radial basis function network)가 구현되어 있는 프로세서에서 스톡캐스틱 컴퓨팅을 적용하였다. 이를 통해 기존에 사용하던 곱셈 연산을 하드웨어 친화적인 곱셈 연산으로 간소화하였다[18].

프로세서 설계 시, 프로세서에 대한 안정성 및 기능 검증이 중요하며, 시뮬레이터를 통해 분석뿐만 아니라 개발 시간 단축과 비용감소를 도모하게 된다. 이러한 시뮬레이터들은 프로세서 설계 검증 혹은 소프트웨어와 프로세서 간의 통합 테스트를 위한 용도로써 사용된다.

프로세서로 구현된 알고리즘은 소프트웨어로 구현된 알고리즘을 통해 얻은 결과와 프로세서에 구현된 알고리즘을 통해 얻은 결과를 비교하여 검증한다. 하지만 프로세서에 구현된 인공지능 알고리즘은 기존 알고리즘과 다르게 총 두 가지 검증 단계가 필요하다.

첫 번째는 인공지능 알고리즘이 정상적으로 구현되었는지에 대한 검증, 두 번째는 객체 인식과 같은 특정 문제를 풀기 위한 알고리즘으로 구현된 인공지능 프로세서

가 적합한지에 대한 검증이다.

특히 프로세서로 구현된 인공지능 알고리즘을 모사하는 성능 분석기가 있다면, 성능 분석 결과와 프로세서로 구현된 인공지능 알고리즘의 결과가 거의 차이가 없게 되므로 구현된 프로세서 없이 성능 분석기로 프로세서에 구현된 알고리즘의 성능과 한계점을 미리 테스트해볼 수 있다는 장점이 있다. 따라서 본 논문에서는 프로세서로 구현된 인공지능 알고리즘을 모사하는 내장형 인공지능 프로세서를 위한 성능 분석기를 제안한다.

2. KNN과 RBF

2.1 KNN(k-nearest neighbors)

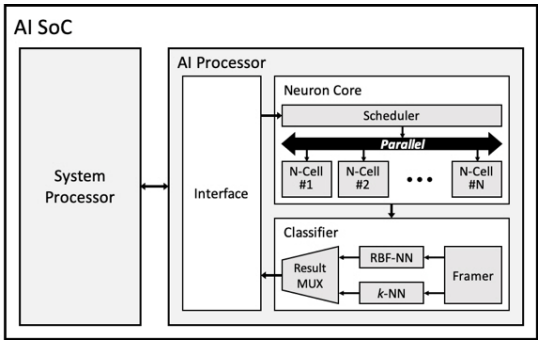
KNN 알고리즘은 분류나 회귀에 사용되는 대표적인 비모수 방식으로, 지도학습에 속한다. KNN 알고리즘은 임의의 데이터 X가 주어졌을 때, 기존 데이터에 제일 가까운 k개의 이웃 정보로 데이터를 예측한다[19].

KNN 알고리즘의 하이퍼 파라미터는 탐색할 이웃 수 k와 거리 측정 방법 두 가지로 나뉜다. k값이 적을 경우 데이터의 지역적인 특성을 지나치게 반영한다는 특징이 있으며, k값이 너무 큰 경우에는 과하게 정규화 한다는 특징이 있다. 거리 측정 방법은 문제 정의에 따라서 유클리디안, 맨해튼, 마할라노비스 상관 거리 중 하나를 선택해서 사용한다.

고차원의 데이터의 분류, 회귀 문제를 KNN 알고리즘으로 해결하려고 하는 경우, 근접 이웃 간의 평균 거리와 전체 데이터 간 평균 거리가 비슷해져 예측 성능이 떨어진다는 한계점이 있다. 이를 해결하기 위해 PCA(principal component analysis)와 같은 차원축소 기법을 통해서 고차원의 데이터를 저차원으로 사영시켜 문제를 해결한다.

2.2 RBF(radial basis function network)

RBF 알고리즘은 분류, 회귀에 사용하는 모수적 방식으로, 지도학습에 속한다. RBF는 인공신경망 중 하나로 MLP(multi- layer perceptron)와 비교했을 경우 1개의 은닉층, 가우시안 활성화 함수, 유클리디안 거리를 사용하는 활성화 함수 인자를 사용한다는 점에서 다르다[20]. 각 뉴런은 가우시안 활성화 함수로 인해 정규분포를 따르며 분류 문제에 사용할 경우, 카테고리의 개수만큼의 뉴런 수를 갖게 된다. 따라서 RBF에서의 학습된 신경망은 n개의 데이터 군집을 가우시안 혼합모델로 표현한다.



(그림 1) 인공지능 프로세서 블록도
(Figure 1) AI processor diagram

3. 인공지능 프로세서 성능 분석기

3.1 인공지능 프로세서〔1〕

본 논문에서는 KNN과 RBF를 구현한 인공지능 프로세서를 활용한다[1]. 프로세서는 그림 1과 같이 System processor와 AI processor로 구성된다. AI processor는 System processor로부터 데이터를 전달받는 Interface, KNN/RBF의 파라미터와 관련 데이터를 저장하고 관리하는 Neuron core, Neuron core를 활용하여 입력 데이터를 분류하는 Classifier로 구성된다.

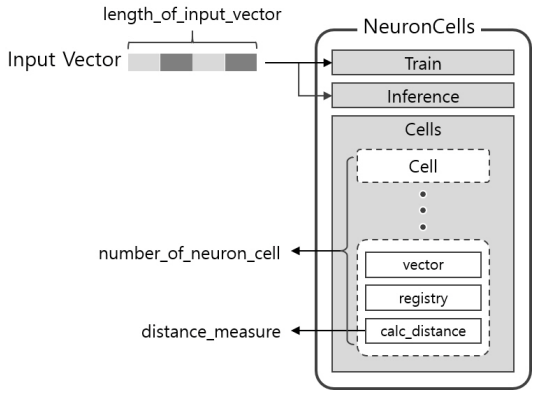
AI processor의 Neuron core는 KNN/RBF 관련 데이터를 저장하고 관리하는 N-cell(neuron cell)과 N-cell에 저장되어 있는 데이터를 병렬적으로 처리하는 Scheduler로 구성된다.

Classifier는 N-cell에 저장되어 있는 데이터와 입력 데이터의 연산 결과가 저장되는 Framer, 인공지능 연산 로직이 들어 있는 KNN/RBF-NN, 최종 분류, 회귀 결과를 출력하는 Result MUX로 구성된다.

학습 단계에는 입력 데이터 X와 정답 데이터 Y가 System processor에서 AI processor로 입력된다. 입력된 데이터는 Interface를 거쳐 Neuron core로 전송되며, Neuron core의 Scheduler를 통해 각 알고리즘에 맞게 데이터를 처리하여 그 결과를 N-cell에 저장한다. 예를 들어 KNN의 경우 N-cell에 입력 데이터 X와 정답 데이터 Y가 저장되며, RBF의 경우 N-cell에 가중치 값과 데이터의 평균값, 분산 값 등이 저장된다.

추론 단계에서는 입력 데이터 X가 인공지능 프로세서로 입력된다. Operator는 입력된 데이터를 N-Cell의 데이터를 알고리즘에 맞게 변환한 데이터와 연산하여 반환한다.

3.2 인공지능 프로세서 성능 분석기



(그림 2) 인공지능 프로세서 성능 분석기
(Figure 2) Performance analyzer for AI processor

인공지능 프로세서 성능 분석기는 인공지능 프로세서와 동작을 같게 만들기 위해 그림 2와 같이 인공지능 프로세서의 Neuron Core를 바탕으로 모델링하였다.

NeuronCells는 학습과 추론을 진행하는 Train, Inference, 학습된 데이터가 저장되는 Cells로 구성된다. Inference는 인공지능 프로세서의 Classifier와 대응되며 Cells는 N-cell과 대응된다. NeuronCells의 Cells는 Cell의 집합으로 각 Cell은 KNN 학습에 사용된 데이터를 저장하는 vector, Cell의 학습 여부를 표현하는 registry, 입력 데이터와 저장된 vector의 거리를 연산하는 calc_distance로 구성된다.

인공지능 프로세서의 성능 분석을 진행하기 위해 먼저 NeuronCells를 생성한다. NeuronCells는 입력 데이터의 크기와 생성할 Cell의 개수를 인자로 받는다. 입력 데이터 크기와 Cells 개수의 곱은 인공지능 프로세서의 메모리와 대응된다. 따라서 각 설정 값에 따라 다르게 생성된 NeuronCells는 다양한 메모리 크기를 갖는 가상의 인공지능 프로세서를 생성한 것과 같다. 메모리 크기가 다르게 생성된 NeuronCells를 활용하면 이미지, 음성 등 여러 애플리케이션에서의 인공지능 프로세서의 성능 분석이 가능하다.

NeuronCells의 학습 단계에서는 데이터 X와 정답 데이터 Y가 Train에 전송된다. Train은 NeuronCells 생성 시 입력된 입력 데이터 크기 값에 따라 입력 데이터 X의 크기를 변경하여 정답 데이터 Y와 함께 N-cell에 저장한다. 이때, 데이터 크기 변경 로직은 보간법 기반의 업/다운 샘플링을 사용한다.

추론 단계에서는 데이터 X가 Inference에 전송된다. Inference는 여러 Cell의 집합인 Cells를 순회하며 입력 데이터 X와 Cell에 저장된 데이터 간의 거리 연산 결과값들을 저장한다. Inference가 입력 데이터 X와 각 Cell에 저장된 데이터간 거리 연산을 마치면, 누적된 거리 값들 중에 최소 거리 값을 찾는다. 이후 Inference는 최소 거리 값을 갖는 Cell의 정답 데이터 Y를 반환한다.

4. 실 험

본 논문에서는 제안한 내장형 인공지능 프로세서를 위한 성능 분석기를 활용하여 이미지, 음성 기반 애플리케이션에 적합하도록 인공지능 프로세서 구조를 최적화하였다.

4.1 애플리케이션

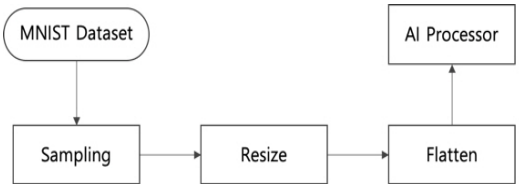
4.1.1 이미지 인식

MNIST dataset[21]은 그림3과 같이 0~9까지의 카테고리를 갖는 손글씨 이미지로 총 60,000장의 학습 데이터와 10,000장의 데이터세트으로 구성된다. MNIST dataset은 회색조 이미지로 픽셀의 질량 중심이 이미지 중앙에 위치하여 정규화 되었다는 특징이 있다.

(그림 3) MNIST 데이터세트
(Figure 3) MNIST dataset

이미지 분류 애플리케이션의 학습은 그림4와 같이 1). 균등 샘플링, 2). 리사이즈, 3). 차원변환, 4). 학습 순으로 진행된다. 먼저 NeuronCells를 카테고리별로 균등하게 학습시키기 위해서 MNIST dataset에서 0~9까지의 데이터를 균등하게 샘플링을 수행한다. 균등하게 샘플링 된 이미지 데이터는 NeuronCells의 입력으로 사용하기 위해서 설정된 데이터 크기에 맞춰 bicubic 보간 알고리즘을 통해 리사이즈 된다. 리사이즈 된 이미지를 NeuronCells의 입력

으로 사용하기 위해서 1차원으로 차원 변환을 수행한다. 1차원으로 변환된 데이터는 NeuronCells의 학습 데이터로 사용된다. 이때 NeuronCells에서의 거리계산 방법은 맨해튼 상관 거리를 사용하였다.

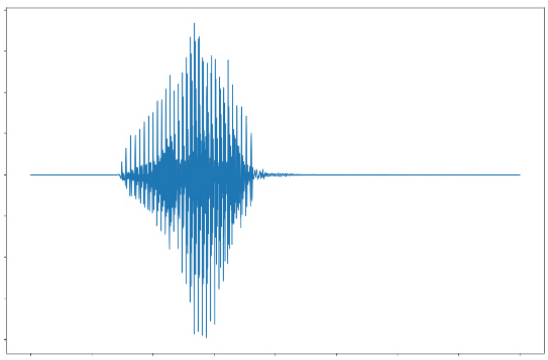


(그림 4) 이미지 인식 애플리케이션
(Figure 4) Image recognition application

4.1.2 음성 인식

Speech command dataset[22]은 음성 기술이 성장했음에도 불구하고 데이터세트의 종류가 확대되지 않은 문제를 해결하고자 Google Brain에서 구축한 데이터세트이다. Speech command dataset은 간단한 음성 명령어를 트리거로 작동하는 애플리케이션을 가정하고 수집되었다. Speech command dataset은 35 카테고리의 음성 파일을 제공하며 약 63,000개의 학습 데이터와 약 21,000개의 검증, 테스트 데이터로 구성된다. 각 음성파일은 1초의 길이를 가지며, 16kHz의 샘플링 레이트로 추출되었다. 아래 그림 5는 Speech command dataset의 일부를 보여준다.

본 연구에서는 인공지능 프로세서가 임베디드를 위한 프로세서임을 고려하여 Speech command dataset의 정답 값 중 “yes”, “no” 총 2개의 정답 값만 사용하였다.

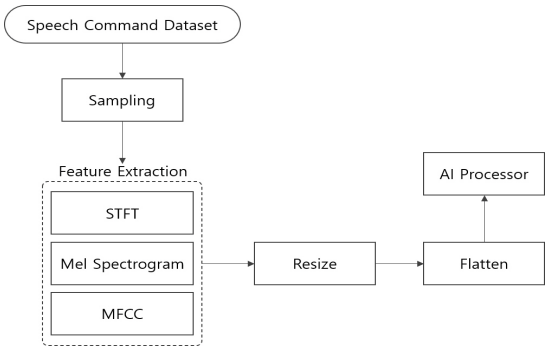


(그림 5) Speech command 데이터세트
(Figure 5) Speech command dataset

음성 분류 애플리케이션의 학습은 그림 6과 같이 1). 균등 샘플링, 2). 특징추출, 3). 리사이즈, 4). 차원변환, 5). 학습으로 진행된다. 앞서 설명한 이미지 애플리케이션의 학습 방법과 유사하게 NeuronCells를 카테고리별로 균등하게 학습시키기 위해서 “yes”, “no” 데이터를 균등하게 샘플링 한다. 샘플링 된 학습 데이터는 음성 신호에 많이 사용되는 특징 추출 방법인 STFT(short time fourier transform), Mel spectrogram, MFCC(mel-frequency cepstral coefficients)을 통해 2차원의 신호로 변환한다.

이때, STFT는 중첩 없는 12.5ms 크기의 hamming window를 사용하였다. Mel spectrogram은 128개의 mel filterbank를 사용하였다. MFCC는 discrete cosine transform을 사용하였으며 40개의 coefficient를 사용하였다.

2차원으로 변환된 음성신호는 NeuronCells의 입력으로 사용하기 위해서 설정된 데이터 크기에 맞춰 bicubic 보간 알고리즘을 통해 리사이즈 하였다. 리사이즈 된 데이터는 NeuronCells의 입력으로 사용하기 위해서 1차원으로 차원 변환을 수행한다. 최종적으로 1차원으로 변환된 데이터는 NeuronCells의 학습 데이터로 사용된다. 이때 NeuronCells에서의 거리계산 방법은 맨해튼 상관 거리를 사용하였다.



(그림 6) 음성 인식 애플리케이션
(Figure 6) Voice recognition application

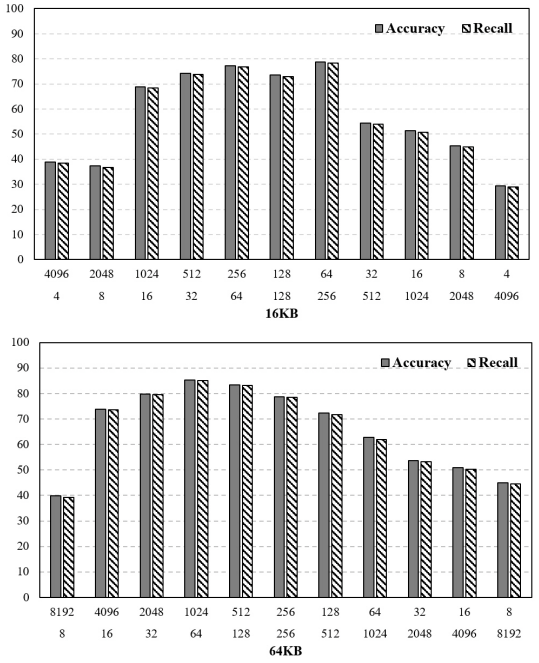
4.2 실험 결과

본 논문에서 제안한 성능 분석기의 유용성을 입증하기 위해 16KB, 64KB의 메모리 크기를 갖는 가상 인공지능 프로세서를 가정하여 실험을 진행하였다. 실험은 각 애플리케이션별로 진행하였으며, 학습은 각 데이터세트의 학습 데이터만 활용하여 진행하였다. 학습한 애플리케이션들의 성능을 평가하기 위해서 인공지능 알고리즘의 성능 지표 중 accuracy와 recall을 측정하였다.

그림 7, 8은 메모리 크기 별 이미지 인식, 음성 인식 애플리케이션의 성능을 보여준다. x축은 메모리별 Cell의 개수와 데이터 크기의 조합을 나타내며 상단은 Cell 개수를, 하단은 데이터의 크기를 나타낸다. y축은 accuracy와 recall의 성능을 백분율로 나타낸다.

이미지 인식 애플리케이션에서 16KB의 메모리 크기를 갖는 가상 인공지능 프로세서의 경우 Cell 개수가 256, 데이터 크기가 64일 때, accuracy가 77.3%, recall이 76.88%로 성능이 제일 뛰어났으며 64KB의 가상 인공지능 프로세서의 경우 Cell 개수가 1024, 데이터 크기가 64일 때, accuracy가 85.25%, recall이 84.99%로 제일 뛰어난 것을 확인할 수 있었다.

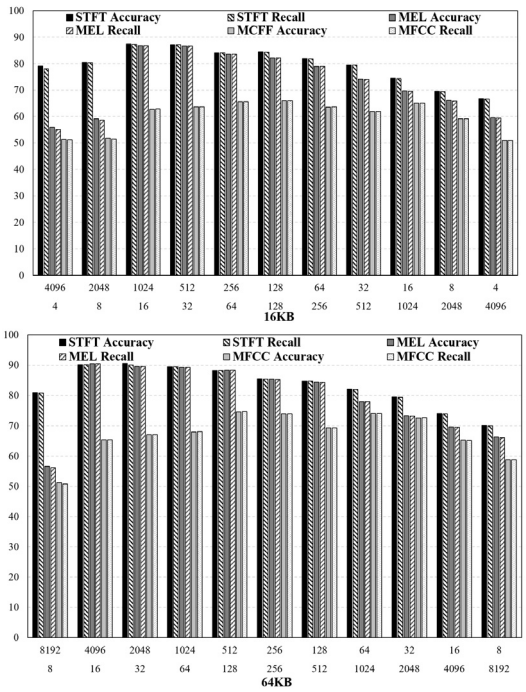
이를 통해 이미지 인식 애플리케이션의 경우 이미지 인식을 위한 최소 데이터의 크기가 64임을 확인할 수 있었으며, 데이터 크기가 64가 보장되는 경우 다양한 데이터를 수용하는 Cell의 수가 많아지면 많아질수록 성능이 향상됨을 확인하였다. 하지만 Cell의 개수가 512에서 1024로 2배 증가한 경우에 accuracy의 향상은 83.33%에서 85.25%로 1.92% 향상에 그쳐 더 많은 메모리 크기에서 큰 차이의 성능향상이 어려움을 확인하였다.



(그림 7) 메모리 크기 별 MNIST 이미지 인식 성능
(Figure 7) Image recognition performance in MNIST dataset

음성 인식 애플리케이션에서 16KB의 가상 인공지능 프로세서의 경우 Cell 개수 1024, 데이터 크기 16, STFT 특징에서 accuracy가 87.4%, recall이 87.3%로 성능이 제일 뛰어났으며 64KB의 가상 인공지능 프로세서의 경우 Cell 개수 2048, 데이터 크기 32, STFT 특징에서 accuracy가 90.66%, recall이 90.06%로 제일 뛰어난 것을 확인할 수 있었다.

음성 인식 애플리케이션의 경우 이미지 인식 애플리케이션과 다르게 최소 데이터 크기 32가 보장되었을 때, 다양한 데이터를 수용하는 Cell의 수가 많아지면 많아질수록 성능이 향상됨을 확인하였다. 음성 인식 애플리케이션 실험에서는 상대적으로 복잡한 음성 데이터로 인해 이미지 인식 애플리케이션보다 더 많은 Cell을 요구함을 확인할 수 있었다. 이미지 인식 애플리케이션과 마찬가지로 음성 인식 애플리케이션도 Cell의 개수가 1024에서 2048로 2배 증가하였음에도 불구하고 accuracy가 89.5%에서 90.6%로 성능향상이 1.1%에 그쳐 더 많은 메모리 크기에서 성능향상의 기대가 어렵다는 점을 확인하였다.



(그림 8) 메모리 크기 별 Speech command dataset 음성 인식 성능
(Figure 8) Voice recognition performance in speech command dataset

5. 결론 및 향후 연구과제

본 논문에서는 특정 인공지능 애플리케이션을 인공지능 프로세서에 적용했을 때, 성능과 한계점을 미리 탐색할 수 있는 내장형 인공지능 프로세서를 위한 성능 분석기를 제안한다.

내장형 인공지능 프로세서를 위한 성능 분석기는 인공지능 프로세서의 구조를 모사하는 구조로 구현되었으며, Cell과 데이터 크기를 가변으로 조절할 수 있도록 구성된다.

본 논문에서는 제안하는 성능 분석기를 활용하여 이미지, 음성 인식 애플리케이션에서 최적의 성능을 탐색하기 위해 다양한 메모리 크기에서 Cell 개수, 데이터 크기 조합별로 알고리즘 성능을 평가한다. 실험 결과 각 애플리케이션별로 최고의 성능을 달성하는 인공지능 프로세서의 메모리 크기와 해당 메모리 크기에서 Cell 개수, 데이터 크기 조합을 탐색할 수 있었다. 본 연구를 통해 인공지능 프로세서를 적용할 수 있는 애플리케이션 탐색 및 인공지능 프로세서의 개선점을 편리하게 탐색할 수 있을 것으로 기대된다.

본 성능 분석기는 현재 인공지능 프로세서의 KNN에 대한 성능분석만이 가능하다는 한계점이 있다. 따라서 향후 RBF 알고리즘 및 딥러닝 모델을 추가하여 기능들을 확장할 계획이다.

참고문헌(Reference)

[1] Yoon, Y.H, Hwang, D.H, Yang, J.H, Lee, S.E, "Intellino: Processor for Embedded Artificial Intelligence," Electronics 2020, 9, 1169. https://doi.org/10.3390/electronics9071169

[2] H. Yoon, K. Kim, J. Chun, "GAN-based shadow removal using context information," Journal of Internet Computing and Services, vol. 20, no. 6, pp. 29-36, 2019. https://doi.org/10.7472/jksii.2019.20.6.29.

[3] S. Yoo, "A study on counting number of passengers by moving object detection," Journal of Internet Computing and Services, vol. 21, no. 2, pp. 9-18, 2020. https://doi.org/10.7472/jksii.2020.21.2.9.

[4] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, "Distilling the Knowledge in a Neural Network", 2015. https://arxiv.org/abs/1503.02531.

[5] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," 2017. https://arxiv.org/abs/1704.04861

[6] X. Zhang, X. Zhou, M. Lin, J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," 2017. https://arxiv.org/abs/1707.01083

[7] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," 2018. https://arxiv.org/abs/1801.04381.

[8] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, Jian Sun, "ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design," 2018. https://arxiv.org/abs/1807.11164.

[9] M. Courbariaux, Y. Bengio, J. David, "BinaryConnect: Training Deep Neural Networks with binary weights during propagations," 2015. https://arxiv.org/abs/1511.00363.

[10] Matthieu Courbariaux, Yoshua Bengio, Jean-Pierre David, "Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1," 2016. https://arxiv.org/abs/1602.02830.

[11] M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi, "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks," 2016. https://arxiv.org/abs/1603.05279.

[12] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, Ali Farhadi, "Regularized Binary Network Training," 2018. https://arxiv.org/abs/1812.11800.

[13] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, Trevor Darrell, "Rethinking the Value of Network Pruning," 2018. https://arxiv.org/abs/1810.05270.

[14] Choi Woong, Kwanghyo Jeong, Kyungrak Choi, Kyeongho Lee, Jongsun Park, "Content Addressable Memory Based Binarized Neural Network Accelerator Using Time-Domain Signal Processing," In Proceedings of the 55th Annual Design Automation Conference. Association for Computing

Machinery, 2018.
https://doi.org/10.1145/3195970.3196014

[15] M. Blott, T. B. Preußer, N. J. Fraser, G. Gambardella, K. O'brien, Y. Umuroglu, M. Leeser, K. Vissers, "Finn-r: An end-to-end deep-learning framework for fast exploration of quantized neural networks," ACM Transactions on Reconfigurable Technology and Systems (TRETS), 11 (3):16, 2018. https://doi.org/10.1145/3242897

[16] M. Zhu, T. Zhang, Z. Gu, Y. Xie, "Sparse tensor core: Algorithm and hardware co-design for vector-wise sparse neural networks on modern gpus," MICRO '52, pp. 359-371, 2019.
https://doi.org/10.1145/3352460.3358269

[17] W. Hua, Y. Zhou, C. Sa, Z. Zhang, G. Suh, "Boosting the performance of cnn accelerators with dynamic fine-grained channel gating," In Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 52, pp. 139150, 2019.
https://doi.org/10.1145/3352460. 3358283.

[18] S. Y. Jang, Y. H. Yoon, S. E. Lee, "Stochastic Computing based AI System for Mobile Devices," IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 2020, pp. 1-2, 2020.
https://doi.org/10.1109/ICCE46568.2020.9042978.

[19] Liu W., Chawla S., "Class Confidence Weighted kNN Algorithms for Imbalanced Data Sets," Advances in Knowledge Discovery and Data Mining. PAKDD. Lecture Notes in Computer Science, vol 6635. Springer, Berlin, Heidelberg, 2011.
https://doi.org/10.1007/978-3-642-20847-8_29

[20] Harpham, C., Dawson, C.W. & Brown, M.R, "A review of genetic algorithms applied to training radial basis function networks," Neural Computing & Applications 13, 193–201, 2004.
https://doi.org/10.1007/s00521-004-0404-5

[21] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, 86(11): 2278-2324, 1998.
https://doi.org/10.1109/5.726791

[22] Pete Warden, "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition", 2018. https://arxiv.org/abs/1804.03209

◐ 저 자 소 개 ◑

황 동 현(Dong Hyun Hwang)
2016년 한경대학교 전기전자제어공학과 학사
2019년 7월~현재 서울과학기술대학교 전자공학과 석사과정
관심분야 : Machine learning, digital system design, biomedical signal processing
E-mail : hwangdonghyun@seoultech.ac.kr

윤 영 현(Young Hyun Yoon)
2013년 3월 서울과학기술대학교 전자IT미디어공학과 학사
2019년 3월~현재 서울과학기술대학교 전자공학과 석사과정
관심분야: Computer architecture, microprocessor system, and system-on-chip design for machine learning
E-mail : yoonyounghyun@seoultech.ac.kr

한 창 엽(Chang Yeop Han)
2014년 3월 서울과학기술대학교 전자IT미디어공학과 학사
2020년 3월~현재 서울과학기술대학교 전자공학과 석사과정
관심분야 : Computer programming, digital system design, SoC for spiking neural network
E-mail : hanchangyeop@seoultech.ac.kr

이 승 은(Seung Eun Lee)
1998년 한국과학기술원, 전기 및 전자공학부 학사
2000년 한국과학기술원, 전자전산학과 석사
2008년 캘리포니아 대학교 어바인, 전기컴퓨터공학과 박사
2009년 3월~2010년 8월 플랫폼 아키텍트, 인텔, SoC 플랫폼 아키텍쳐 랩
2010년 9월~2016년 9월 서울과학기술대학교 전자공학과 조교수
2016년 10월~현재 서울과학기술대학교 전자공학과 부교수
관심분야 : Computer architecture, multi-processor system-on-chip design, low-power and resilient VLSI, and hardware acceleration for emerging applications
E-mail : seung.lee@seoultech.ac.kr