NPU를 위한 연산자 퓨전 기반 양자화 신경망 모델의 정확도 향상

Accuracy Improvement of Quantized Neural Network Model Based on Operator Fusion for NPU

이제민[1], 유미선, 권용인, 김영주, 김태호
한국전자통신연구원
(Jemin Lee, Misun Yu, Yongin Kwon, Young-Joo Kim, Taeho Kim)
(Electronics and Telecommunications Research Institute)

Abstract: Although convolutional neural networks(CNNs) are widely adopted in various domains due to remarkable performance, executing CNNs on embedded systems has remained a challenge due to their heavy computation comparing with traditional algorithms. One of the dominant methods to optimize CNN complexity is quantization that reduces memory access by mapping continuous values to a smaller set of discrete values. However, quantization leads to an accuracy drop. In this paper, we combine profile-based quantization and operator fusion on NPU for maintaining high accuracy. Our method improves by 10.14% better accuracy than the existing result that is based on a single scale on VTA.

Keywords : quantization, operator fusion, neural network compiler

Ⅰ. 서론

딥러닝 기술은 많은 응용 분야에서 폭넓게 사용되고 있다. 최근에는 임베디드 시스템에서 딥러닝 응용을 에너지 효율적으로 동작하기 위하여 NPU (Neural Processing Unit)를 활용한 딥러닝 연구가 활발히 진행되고 있다 [1].

출시된 여러 NPU들 중에서 워싱턴대학에서 개발한 VTA[2]는 공개 소스로, HLS 설계 코드와 JIT 컴파일러가 모두 공개되어 연구 분야에서 많이 활용되고 있다. VTA는 다른 모바일 엣지 NPU들과 마찬가지로 DRAM 접근과 산술 연산자의 전력 소모를 줄이기 위해 32bit 연산이 아닌 8bit 이하의 연산자만을 지원하도록 설계되었다. 그러므로 양자화된 신경망 모델을 입력으로 받는 것이 필수적이다. 그러나 단순한 양자화는 모델의 정확도를 많이 감소시킬 수 있으므로 이러한 문제를 해결하는 방법이 필요하다.

본 연구에서는 프로파일링 기반 방식과 연산자 퓨전이 VTA와 같은 모바일/엣지 NPU들을 위한 8bit 양자화 모델의 정확도 향상에 얼마나 영향을 미치는지를 분석하였다. 프로파일링 기반 양자화 적용을 위해서 Glow 컴파일러[3]를 사용했으며, 해당 컴파일러를 수정하여 VTA 동작에 맞춘 연산자 퓨전과 비트 시프트 기반 양자화 연산을 구현했다. 추가된 연산자 퓨전은 VTA의 GEMM과 ALU를 이용하여 컨볼루션과 ReLU 연산의 연속처리를 가능하게 한다. 비트 시프트 기반 양자화는 VTA에서 양자화를 처리하는 방식으로 Glow 컴파일러에 포함된 인터프리터를 수정하여 구현했다. 실험 결과 제안된 방법은 Resnet18v1 모델에 대해서 VTA 기존 결과[4]보다 10.14% 높은 이미지넷 Top1 정확도를 나타냈다.

Ⅱ. 배경 지식

[1] 교신저자(Corresponding Author)
이제민: 한국전자통신연구원
※ 이 논문은 2020년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2018-0-00769,인공지능 시스템을 위한 뉴로모픽 컴퓨팅 SW 플랫폼 기술 개발)

1. VTA 인터프리터

VTA는 워싱턴대학에서 개발한 공개 하드웨어로 HLS 코드로 내부 아키텍처가 모두 공개되어있다. VTA 연산기는 행렬 곱셈을 지원하는 GEMM과 비트 시프트, 산술 연산을 지원하는 ALU로 구성된다. VTA는 모두 8bit 연산만을 지원하도록 구성되어 있으므로 입력데이터는 8bit 정수형으로 변환해서 처리하는 양자화 과정을 거쳐야 한다. 본 연구에서는 양자화로 인한 연산 이득 최대화를 위해서 균등(Uniform) 대칭형(Symmetric) 방식을 사용 한다[5].

FP32 실숫값을 균일 대칭형 방식으로 8bit 양자화하는 방식은 수식(1)과 같다. *scale*은 입력된 실수 x_{fp32}의 절대 최댓값을 8bit 표현 범위의 절반으로 나눈 값을 의미한다. 양자화된 x_{i8}을 실수 x_{fp32}로 역양자화하는 방법은 수식(2)와 같이 *scale*을 곱하는 방식을 취한다.

$$x_{i8} = QUANTIZE(x_{fp32}) = ROUND\left(\frac{x_{fp32}}{scale}\right), \quad (1)$$

$$scale = \frac{ABS(MAX(x_{fp32}))}{2^{(Nbit-1)}-1}$$

$$x_{fp32} = DEQUANTIZE(x_{i8}) = x_{i8} \times scale \quad (2)$$

VTA에서는 양자화 과정에서 필요한 *scale* 연산을 효율적으로 수행하기 위해 실제 *scale* 곱셈을 수행하는 대신 비트 시프트 연산을 수행한다. 본 연구에서는 이러한 VTA의 비트 시프트 양자화 방식을 Glow 컴파일러의 인터프리터에 구현하였으며 이것을 VTA 인터프리터라 부른다.

2. Glow 컴파일러의 양자화 방식

Glow 컴파일러는 양자화를 위해서 두 단계를 수행한다. 첫 번째 단계는 엑티베이션들의 텐서 값 범위를 알기 위해서 프로파일링을 수행하는 단계이다. 그림1은 이러한 프로파일링을 수행하기 위한 그래프 중간표현의 변화를 보여준다. 프로파일링은 FP32의 연산정밀도를 요구하므로 실수를 처리할 수 있는 CPU와 같은 타겟에서 수행된다. 프로파일링은 신경망 모델 학습에 사용된 이미지를 입력으로 사용해 이루어진다. 두 번째 단계는 수집된 엑티베이션 텐서들의 데이터를 기반으로 양자화 파라미터인 스케일을 계산하고, 각각의 연산자들에 삽입하는 단계이다. 본 연구에서는 두 번째 단계 이후에 컨볼루션과 ReLU를 퓨전하는 최적화 패스를 추가하였다.

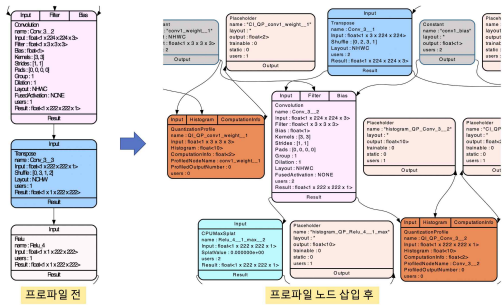

그림 1 엑티베이션 텐서들의 분포를 얻기 위해서 프로파일 노드가 삽입된 그래프 중간표현

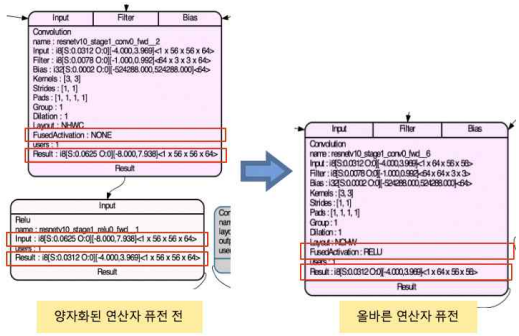

그림 2 그래프 중간표현 수준에서의 양자화된 컨볼루션과 ReLU의 연산자 퓨전

3. 양자화를 고려한 Glow 컴파일러의 연산자 퓨전

양자화된 연산자들을 퓨전 하기 위해서는 양자화 파라미터들을 연산자별로 유지하는 것이 중요하다. 그림2는 컨볼루션과 ReLU를 정상적으로 퓨전했을 때와 그렇지 않을 때를 나타낸다. 본 연구에서 제시하는 연산자 퓨전은 ReLU의 출력값 *scale*을 컨볼루션 출력 *scale*로 치환한다. 이 경우, 컨볼루션의 출력값 *scale*을 유지할 필요 없으며 컨볼루션의 출력값과 Relu의 입력값에 대한 양자화 과정이 없어지게 되어 전체 모델의 정확도가 향상되고 전체 계산 오버헤드도 감소하는 이점이 있다.

*scale*치환과 정확도 향상을 수식으로 설명하면 다음과 같다. 수식(3)은 bias를 생략하여 컨볼루션 일부분을 단순화한 것으로, 양자화된 피쳐맵 x_{i8}와 양자화된 w_{i8}를 곱하고 실수 y_{fp32}를 얻기 위해 역양자화를 하는 과정을 나타낸다. 수식(4)는 컨볼루

션과 ReLU의 연속된 연산과정을 나타낸다. 수식(4)의 전개를 통해 S_y가 소거됨을 알 수 있다. S_y는 그림2에서 ReLU에 의해서 치환된 컨볼루션의 출력값 *scale*이다.

ReLU 특성에 따라 입력값 *scale*은 항상 출력값 *scale*보다 크거나 같으므로, $S_a \leq S_y$를 따른다. 퓨전에 의한 정확도 향상은 $S_a < S_y$의 경우에 발생한다. 이 경우 ReLU 동적 출력을 더 정확하게 반영한 *scale*인 S_a로 한 번만 양자화되므로 정밀도 손실이 줄어든다.

$$y_{fp32} = S_x S_w \left(\sum_{n=1}^{N} (x_{i8}^n w_{i8}^n) \right) \tag{3}$$

$$y_{i8} = ROUND \left(\frac{S_x S_w}{S_y} \left(\sum_{n=1}^{N} (x_{i8}^n w_{i8}^n) \right) \right)$$

$$a_{i8} = \frac{S_y}{S_a} F \left(ROUND \left(\frac{S_x S_w}{S_y} \sum_{n=1}^{N} (x_{i8}^n w_{i8}^n) \right) \right) \tag{4}$$

$$a_{i8} = \frac{S_y}{S_a} MAX \left(0, ROUND \left(\frac{S_x S_w}{S_y} \sum_{n=1}^{N} (x_{i8}^n w_{i8}^n) \right) \right)$$

$$a_{i8} = MAX \left(0, ROUND \left(\frac{S_x S_w}{S_a} \sum_{n=1}^{N} (x_{i8}^n w_{i8}^n) \right) \right)$$

$$a_{i8} = MAX \left(0, ROUND \left(\frac{y_{fp32}}{S_a} \right) \right)$$

Ⅲ. 실험 결과

1. 실험 방법

비교를 위해서 기존 VTA 연구에서 사용된 Mxnet Gluon의 Resnet18v1[1]을 ONNX 형식으로 변환해서 실험에 사용했다. 사용한 타겟은 VTA의 비트 시프트 양자화 기능을 Glow 인터프리터에 구현한 VTA 인터프리터를 사용했다. 사용한 데이터셋은 이미지넷12 검증 데이터셋 이미지 50,000개이며, 이를 이용해서 Top-1 정확도를 측정했다. 정확도 비교 실험은 양자화를 하지 않고 실행한 경우(Glow(FP32)), 프로파일링 기반으로 양자화를 실행한 경우(Glow(I8)), 연산자 퓨전 적용 후 실행한 경우(Glow_w/_Fusion(I8))에 대하여 수행하였다. 그리고 해당 실험 결과를 기존 결과(TVM-VTA(I8))과 비교 했다.

2. 실험 결과

실험 결과, 그림3과 같이 Glow를 이용한 프로파일링 기반 양자화 방식은 기존 VTA 결과(TVM-VTA(I8))[4]보다 정확도가 4.91% 향상된 결과를 보였고 퓨전을 이용하면 10.14% 향상된 결과를 얻었다. 기존 VTA 결과가 37%로 낮은 정확도를 나타내는 이유는 TVM[6]에서 제공되는 Relay[7]를 이용한 방식이 전체 모델에 대해서 단일 스케일을 사용하기 때문이다. 반면에 본 연구에서 제안한 프로파일링 기반 방법을 사용할 경우 이보다 향상된 결과를 얻을 수 있고 앞서 설명한 것과 같이 연산자를 퓨전 하면 중간 양자화 과정이 생략되고 더 정밀한 동적 범위로 양자화되므로 정확도 손실을 줄여 가장 FP32 모델과 근접한 결과를 얻을 수 있었다.

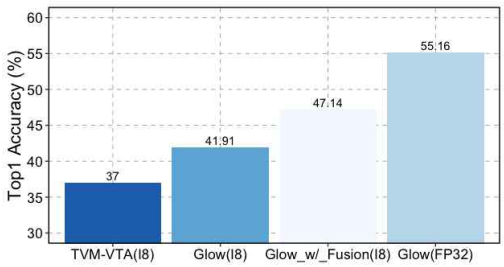

그림 3 양자화 기법에 따른 Resnet18v1 모델의 이미지넷 검증 데이터 Top1 정확도 결과

Ⅳ. 결 론

본 연구에서는 연산자 퓨전이 양자화에 미치는 영향을 분석했다. Glow 컴파일러에서 지원하는 프로파일링 기반 양자화 방식은 기존 VTA에서 수행하는 단일 스케일 방식보다 더 좋은 정확도를 보였고 연산자 퓨전을 추가로 적용하면 더 높은 정확도가 달성된다. 향후 연구로 해당 기법을 다양한 모델에 적용해볼 계획이다.

참 고 문 헌

[1] Sze, V., Chen, Y. H., Yang, T. J., & Emer, J. S. (2017). Efficient processing of deep neural networks: A tutorial and survey. Proceedings of the IEEE, 105(12), 2295-2329.

[2] Moreau, Thierry, et al. "A hardware - software blueprint for flexible deep learning specialization." IEEE Micro 39.5 (2019): 8-16.

[3] Rotem, Nadav, et al. "Glow: Graph lowering compiler techniques for neural networks."

[1] https://mxnet.apache.org/

arXiv preprint arXiv:1805.00907, 2018.

[4] Moreau, Thierry, Tianqi Chen, and Luis Ceze. "Leveraging the vta-tvm hardware-software stack for fpga acceleration of 8-bit resnet-18 inference." Proceedings of the 1st on Reproducible Quality-Efficient Systems Tournament on Co-designing Pareto-efficient Deep Learning. 2018. 1.

[5] Wu, Hao, et al. "Integer Quantization for Deep Learning Inference: Principles and Empirical Evaluation." arXiv preprint arXiv:2004.09602 (2020).

[6] Chen, Tianqi, et al. "TVM: An automated end-to-end optimizing compiler for deep learning." USENIX Symposium on Operating Systems Design and Implementation, 2018.

[7] Roesch, Jared, et al. "Relay: A new ir for machine learning frameworks." Proceedings of the 2nd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages. 2018.