Glow 컴파일러 기반 타겟에 독립적인 연산자 퓨전을 활용한 CNN 추론 가속화

(Accelerating the Inference of CNN using Target-Independent Operator Fusion based on the Glow Compiler)

이 제 민, 유 미 선, 권 용 인, 김 영 주, 김 태 호
한국전자통신연구원
(Jemin Lee, Misun Yu, Yongin Kwon, Young-Joo Kim, Taeho Kim)
(Electronics and Telecommunications Research Institute)

Abstract: A variety of convolutional neural networks(CNNs)-based applications ranging from computer vision to neutral language processing has been rapidly evolved. However, the demand for high computation is a hurdle to adopt CNNs in embedded systems due to their limited resources. To afford costly CNNs on embedded systems, accelerating them is gaining attention. Operator fusion that reduces off-chip memory accesses is one of the dominant approaches to accelerate the inference of CNNs. In this paper, we investigate performance gains by target-independent operator fusion based on the Glow compiler. As a result, the operator fusion that batch normalization operators are directly merged to their preceding convolution operators leads to latency improvements of 14%, 11%, and 9% in MobileNet, ResNet152, and ResNet18, respectively.

Keywords : deep learning compiler, operator fusion, convolutional neural network

I. 서 론

컨볼루션 신경망은 컴퓨터 비전과 자연어처리 등의 여러 응용 분야에서 활용되고 있다. 하지만 컨볼루션 신경망은 추론을 위해서도 많은 연산이 필요하여 임베디드 시스템에서는 그 활용이 제한적이다. 이러한 문제를 해결하기 위해서 최근 임베디드 시스템에서 딥러닝 추론을 가속화하는 연구들이 다양하게 제안되고 있다 [1].

신경망 추론을 가속화 하는 방법의 하나인 연산자 퓨전은 신경망 연산자를 타겟 하드웨어에 맞춰서 하나로 합치는 것을 의미한다. 연산자 퓨전은 추론에 따른 중간 결과들을 따로 저장하지 않으므로 DRAM 접근을 줄여 연산속도와 에너지 소모를 현격히 줄이게 된다. 일반적으로 multiply-and-accumulates (MACs)과 같은 직접적인 연산보다 메모리 접근이 1,000배 이상 에너지 소모를 많이 발생시키며 추론 지연 시간 측면에도 세 배 이상 차이가 있는 것으로 알려져 있다 [2, 3].

본 연구에서는 Glow [4] 컴파일러를 활용하여 신경망 모델에 따라 연산자 퓨전이 추론 지연 시간 감소에 얼마나 효과적인지를 분석했다. 사용한 연산자 퓨전은 컨볼루션과 배치 정규화[5]를 대상으로 한다. 해당 퓨전은 컴파일 시간에 배치 정규화가 완전히 앞선 컨볼루션 연산에 병합되는 형태로 수행된다. 이는 타겟 하드웨어에 상관없이 항상 연산상 이득을 가져오게 되므로 CPU 하드웨어 상에서도 그 효과를 알 수 있다. 따라서 실험은 CPU 환경에서 세 가지 신경망 모델에 대해서 수행했으며, 실험 결과 연산자 퓨전은 최대 14%의 추론 지연 시간을 감소시켰다.

*Corresponding Author (leejaymin@etri.re.kr)
이제민, 유미선, 권용인, 김영주, 김태호: 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).

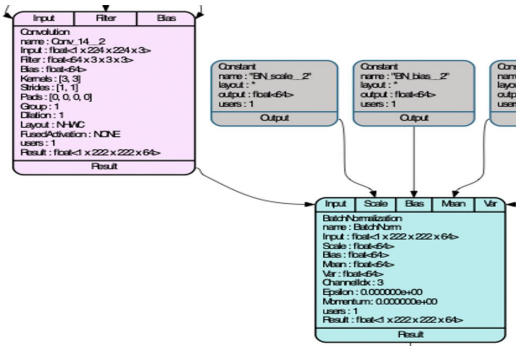

그림 1 OptimizeBatchNorm 패스로 퓨전이 가능한 컨볼루션과 배치 정규화의 형태

Ⅱ. 배경 지식

1. Glow 컴파일러의 연산자 퓨전

Glow 컴파일러는 학습이 완료된 모델을 입력으로 받아 상위 수준과 하위 수준 중간 표현을 각각 생성하는 흐름을 거친다. 마지막으로 타겟이 CPU일 경우 LLVM을 이용해서 최종 머신 코드를 생성한다. 본 연구에서는 타겟에 독립적인 연산자 퓨전인 OptimizeBatchNorm 패스에 대해서 분석한다. 해당 패스는 그림1과 같이 컨볼루션 연산자 뒤에 배치 정규화 연산자가 이어지면 두 연산자를 퓨전하는 최적화를 담당한다.

2. 컨볼루션과 배치 정규화 퓨전 방식

컨볼루션 연산 다음에 배치 정규화 연산을 해야 한다면, 이어지는 배치 정규화 연산은 컨볼루션의 가중치와 바이어스 값에 포함되도록 미리 계산될 수 있다. 수식 (1)은 컨볼루션 연산과 배치 정규화 연산을 순차적으로 실행했을 때 생성되는 i번째 계층의 Y_i값을 나타낸다. Conv(x)는 입력 x에 대한 컨볼루션 연산을 나타내며, 가중치 W_i와 입력 X_i의 행렬 곱셈과 바이어스 B_i의 덧셈으로 이뤄진다. $\mu_i, \sqrt{\sigma_i^2}, \gamma_i, \beta_i$는 배치 정규화에 해당하는 가중치들을 의미한다. 해당 연산을 전개하고 치환하면 수식 (2)와 (3)과 같다. 최종적으로 수식 (4)와 같이 배치 정규화는 변형된 컨볼루션 가중치 A와 바이어스 C로 통합되어 실행 시 연산상 이득을 가져오게 된다.

$$Y_i = \gamma_i\left(\frac{Conv(x)_i - \mu_i}{\sqrt{\sigma_i^2}}\right) + \beta_i \quad (1)$$
$$, Conv(x)_i = W_i * \ X_i + B_i$$

$$Y_i = \gamma_i\left(\frac{W_i * \ X_i + B_i - \mu_i}{\sqrt{\sigma_i^2}}\right) + \beta_i \quad (2)$$

$$Y_i = \delta_i(W_i * \ X_i) + (\delta_i B_i - \delta_i \mu_i + \beta_i) \quad (3)$$
$$, \delta_i = \frac{\gamma_i}{\sqrt{\sigma_i^2}}$$

$$Y_i = A_i * \ X_i + C_i \quad (4)$$
$$, A_i = \delta_i * \ W_i$$
$$, C_i = (\delta_i B_i - \delta_i \mu_i + \beta_i)$$

위와 같은 연산자 퓨전 최적화는 타겟 하드웨어 상관없이 컴파일 시간에 상수 연산을 미리 수행하는 방식이므로 추론 지연 측면에서 항상 이득을 가져온다. 본 논문에서는 위와 같이 선형 관계에 있는 연산자만을 퓨전의 대상으로 고려한다.

Ⅲ. 실험 결과

1. 실험 방법

ONNX 형식으로 저장된 ResNet18, ResNet152, MobileNet을 ONNX Model Zoo에서 내려받아서 Glow 컴파일러를 이용해서 실행했다. 컨볼루션과 배치 정규화의 퓨전에 따른 효과를 분석하기 위해서 Glow 컴파일러를 수정하여 OptimizeBatchNorm 패스를 활성화했을 때와 그렇지 않을 때의 조건으로 3개의 CNN 모델을 각각 실행했다. 평균 추론 지연 시간과 표준 오차를 계산하기 위해서 같은 입력으로 각각의 모델에 대해서 10번씩 반복 수행했다.

2. 실험 환경

실험을 위해서 2018-Macbook-Pro 랩톱 인텔 i7 2.6 Ghz 메모리 32GB를 사용 했다. 소프트웨어는 모하비 10.14.6 운영체제 버전에 Glow는 llvm-8을 사용 했다.

3. 실험 결과

딥러닝 모델에 따라서 컨볼루션과 배치 정규화 연산자의 수가 다르므로 서로 다른 모델 3개를 표1과 같이 선택해서 실험했다. 기본적으로 세 개의 모델에서 대부분의 컨볼루션 연산자 뒤에는 배치 정규화 연산자가 이어져서 나오므로 모두 퓨전 되어 수행된다. ResNet 계열들은 모든 컨볼루션이 퓨전 되었고 MobileNet의 경우 맨 마지막 1x1 컨볼루션을 제외하고 모두 퓨전 되었다.

그림2는 세 모델의 추론 지연 시간을 나타내

표 1. 실험에 사용된 딥러닝 모델의 컨볼루션 연산의 수와 퓨전 된 컨볼루션 연산의 수

모델	# of Conv	# of Fused Conv
ResNet18	20	20
ResNet152	155	155
MobileNet	54	53

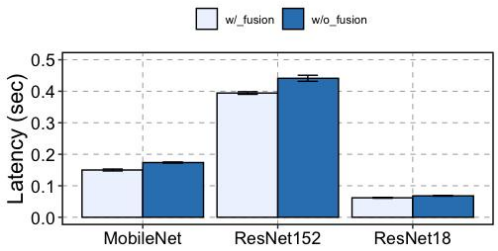

그림 2 절대 적인 추론 지연 시간

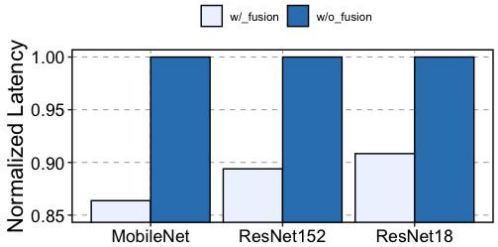

그림 3 상대적인 추론 지연 시간

고 그림 3은 정규화를 통한 상대적인 추론 지연 시간을 나타낸다. 결과적으로 연산자 퓨전을 적용했을 경우 MobileNet, ResNet152, ResNet18 각각의 모델에 대해서 14%, 11%, 9%의 추론 지연 시간이 줄어들었다. 이러한 결과를 통해서 컨볼루션과 배치 정규화 연산자 퓨전의 수가 많을수록 퓨전에 따른 이득이 큰 것을 알 수 있다.

Ⅳ. 결 론

본 연구에서는 컨볼루션 신경망 모델의 추론 지연을 줄이기 위한 가속화 방법으로 사용되고 있는 연산자 퓨전에 대해서 분석했다. Glow 컴파일러에 의해서 제공되는 타겟에 독립적인 연산자 퓨전인 컨볼루션과 배치 정규화를 병합하는 최적화는 해당 패턴이 많을수록 효과가 큰 것을 보였다. 앞으로는 타겟 하드웨어를 직접 고려하여 더 많은 연산자 퓨전을 수행하는 연구를 진행할 예정이다.

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] M. Horowitz, Computing's energy problem (and what we can do about it), in International Solid-State Circuits Conference (ISSCC), 2014.

[3] B. Chen and J. M. Gilbert, Introducing the CVPR 2018 on-device visual intelligence challenge, Google AI Blog, 2018. https://ai.googleblog.com/2018/04/introducing-cvpr-2018-on-device-visual.html

[4] Rotem, Nadav, et al. "Glow: Graph lowering compiler techniques for neural networks." arXiv preprint arXiv:1805.00907, 2018.

[5] S. Ioffe and C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in International Conference on Machine Learning (ICML), 2015.