모바일 장치에서 Low-bit Precision을 활용한 딥 러닝 모델의 가속 및 성능 분석

김호승 O, 홍경환, 신동군
성균관대학교 전자전기컴퓨터공학과
ghyutjik123@gmail.com, redcarrottt@gmail.com, dongkun@skku.edu

영문제목

Hoseung Kim O , Gyeonghwan Hong, Dongkun Shin
Department of Electrical and Computer Engineering, Sungkyunkwan University

요 약

최근 모바일 장치의 CPU나 GPU의 성능이 향상됨에 따라, 모바일 장치에서 직접 딥 러닝을 수행하여 여러 분야에서 활용하고 있다. 그러나, 복잡한 딥 러닝 모델을 요구하는 해상도가 큰 이미지의 분류나 물체 인식과 같은 작업을 실시간으로 수행하기는 어렵다. 딥 러닝 모델의 성능을 유지하며 복잡도를 줄이기 위해 다양한 연구가 진행되고 있는데, 많은 연구들이 기존 모델을 적은 bit수만 사용하여 좋은 성능을 보이고 있다. 그러나 모바일 장치의 경우 다양한 precision을 제공하지 않기 때문에 적은 bit의 모델을 수행할 때 성능 향상에 한계가 있다. 본 논문에서는 이러한 모바일 장치의 한계를 극복하기 위해 여러 precision을 소프트웨어적으로 제공하였을 때의 성능에 대해 분석하고, 다양한 precision으로 딥 러닝 모델을 수행한다. 최종적으로, 다양한 precision을 사용하여 ResNet-20을 수행했을 경우 기존보다 정확도는 1.08%감소한 반면, 1.18배의 성능 개선과 메모리 사용량에서 15.99배의 이득을 얻는다.

1. 서 론[1]

딥 러닝은 영상 및 이미지 분류, 물체 인식, 그리고 자연어 처리와 같은 많은 분야에서 뛰어난 성능을 보이고 있다. 그러나, 이런 어려운 작업들을 정확히 수행하기 위해서 딥 러닝 모델이 복잡해지고, 그에 따라 많은 계산 양과 스토리지 사이즈를 요구한다.

이러한 문제점을 극복하기 위해 딥 러닝 모델의 압축에 관한 연구들이 활발하게 진행 중이다. 대표적으로 적은 bit수를 사용하는 Quantization[1,2]이 있다. Quantization을 사용하면, precision에 따라 일반적인 하드웨어에서도 스토리지 및 실제 수행 시간에서 큰 이득을 보며 기존 모델의 정확도를 유지시킬 수 있다.

표 1은 서버 급 GPU인 Nvidia GPU의 Volta, Turing architecture와 ARM기반 모바일 장치의 특정 CPU와 GPU에서 지원하는 data type을 정리하여 비교한 것이다. 표 1에서 볼 수 있듯이, NVIDIA GPU의 경우 다양한 type을 지원하는 반면 ARM기반 모바일 장치의 경우 지원하는 type도 다양하지 않기 때문에 NVIDIA GPU에 비해 Quantization의 장점을 크게 활용하지 못한다.

본 논문에서는 ARM 기반 모바일 장치에서의 CNN 추론을 가속하기 위해, 8bit 이하의 low-bit precision 연산을 제안한다. 이를 위해, 모바일 장치용 딥 러닝 라이브러리인 ACL(ARM Compute Library)[4]의 General Matrix Multiply(GEMM)을 확장하였다. 또한, 기존 모델에 low-bit precision 연산을 도입하여 이를 더욱 효율적으로 활용하기 위해 계층 별 quantization[3]을 적용하였다. 기존 CNN 모델에 계층 별 quantization을 적용하고, 실제 ARM 기반 모바일 장치에서 low-bit precision 연산으로 추론하여 실험하였다. 실험에서는 precision을 변경함에 따라 정확도와 성능 간의 trade-off가 있음을 보인다.

이 논문은 2019년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.IITP-2017-0-00914, 지능형 IoT 장치용 소프트웨어 프레임워크)

2. 관련 연구

일반적으로, 딥 러닝에서 좋은 성능을 보이는 데 있어서, 높은 precision을 사용하는 것이 크게 중요하지 않다는 것을 [1,2]등 많은 연구 들에서 알 수 있다. 심지어 XNOR-Net[1]에서는 binary 값만 사용하면서 XNOR과 bit-count로만 모든 연산을 대체하여 32bit 연산 대비 58배나 빠른 성능을 보여준다.

대부분의 딥 러닝 모델에서 사용하는 convolution 연산을 수행하는데 있어서 가장 중요한 것은 행렬

표 1. Nvidia GPU와 ARM이 지원하는 Data Type

	Nvidia GPU		ARM	
	Volta	Turing	ARMv7-a	Mali-T628
FP32	√	√	√	√
FP16	√	√		√
(U)INT16	√	√	√	√
(U)INT8	√	√	√	√
(U)INT4		√		
1-Bit		√		

연산인 GEMM이다. 따라서, convolution의 성능을 향상시키기 위해선 GEMM을 최적화하는 것이 매우 중요하다. 본 논문에서는 기존 ACL에 내장된 (1) 커널을 확장하여 8bit이하 precision에 대해서 low-bit GEMM을 구현하였다.

3. Low-bit GEMM

본 논문에서 제안하는 low-bit GEMM은 GEMM 연산을 확장하여 만들었다. CNN 모델을 추론할 때 각 convolution layer가 low-bit GEMM 커널을 호출하면, 해당 커널 내부에서 quantization stage, GEMM stage 순서로 진행된다.

Quantization stage에서는 식 (1)과 같이 full precision의 입력과 weight를 low-precision으로 변환한다. x는 w_{min}과 w_{max}사이의 값을 가지는 full precision 입력이고, $\hat{x}$는 quantize된 k-bit 결과이다.

$$normalize(x) = \frac{x - w_{min}}{w_{max} - w_{min}},$$

$$\hat{x} = \frac{1}{2^k - 1} round\left((2^k - 1) \cdot normalize(x)\right).$$

또한 이에 대한 결과를 GEMM stage에서 활용하기 위해 메모리에 연속적으로 저장하게 된다.

그림 1은 GEMM stage에서 4bit의 weight와 input 16개를 연산하는 과정을 나타낸다. 먼저, 메인 메모리에 저장되어 있는 weight와 input을 vector register로 load한다. 4bit weight와 input을 8bit SIMD MAC 명령어로 연산하기 위해, vector register로 load한 weight와 input을 좌측 4bit의 데이터에 대해서 shift, 우측 4bit에 대해 masking을 통해 서로 다른 register에 저장한다. 그 후 해당 register에 저장된 input과 weight에 대해 SIMD MAC 연산을 한다. 연산 결과를 메인 메모리에 저장되어 있던 형태로 만들어 주기 위해서, shift와 masking을 적용하여 결과를 저장한다. 이 때, 데이터를 불러오고 저장하기 위한 메인 메모리 접근이 크게 줄어드는 대신 shift, masking 연산이 추가 발생한다.

4. 실험

4.1. 실험 환경

실험은 ARMv7-a 아키텍처 기반으로, Cortex-A15와 Cortex-A7을 장착한 모바일 장치인 Odroid-XU4를 사용하였다. 특히, ARM 프로세서에서 사용할 수 있는 SIMD(Single Instruction Multiple Data)확장 도구인 NEON기술을 활용하였고, 실제 모델의 추론 및 연산은 딥 러닝 라이브러리인 ACL 18.11버전을 사용하였다. 현재 ACL은 8bit 정수형과 32bit 실수형 GEMM 연산에 대한 커널을 제공하여 이를 사용하고, 그 외의 것들은 8bit 정수형 GEMM 연산 커널을 기반으로 직접 구현하였다.

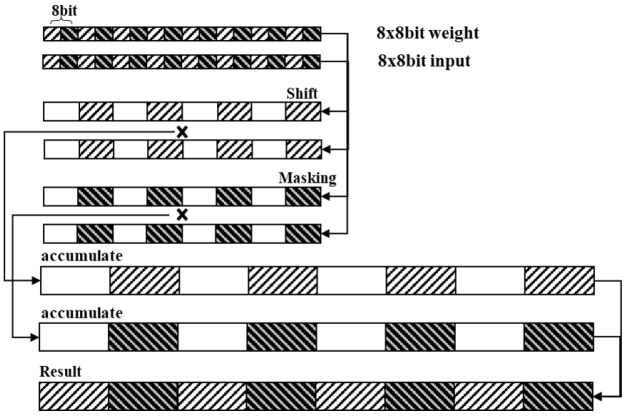

그림 1. 4bit weight와 input에 대한 GEMM 연산 과정

다양한 precision으로 모델을 학습시키기 위해서, Pocketflow[5]라는 강화학습 기반의 프레임워크를 사용하였다. Pocketflow는 동일한 스토리지 사이즈에서 reward 함수를 모델의 정확도로 하여 계층 별 precision을 학습한다. 딥 러닝 모델은 ResNet-20을 사용하였고, 대표적인 이미지 분류 데이터셋인 Cifar-10을 학습시켰다. 기존 pretrained된 full precision 모델은 91.93%의 정확도를 가진다.

4.2. Precision별 Low-bit GEMM 성능 분석

그림 2는 1024x1024 크기의 행렬에 대해 서로 다른 precision을 적용한 low-bit GEMM 연산의 소요 시간을 나타낸다. 그래프 중 GEMM은 GEMM stage의 소요 시간, O/H는 quantization stage와 dequantization stage의 소요 시간이다. x축의 32는 32bit 실수형, 나머지는 n-bit 정수형을 나타낸다. 16bit와 3bit를 제외하고 작은 precision을 사용함으로써 성능이 개선되었다.

구체적인 성능 차이에 대한 이유는 그림 3을 통해 알 수 있다. 4bit 이하의 경우 input과 weight에 대한 shift, masking을 더 수행하기 때문에 instruction이 크게

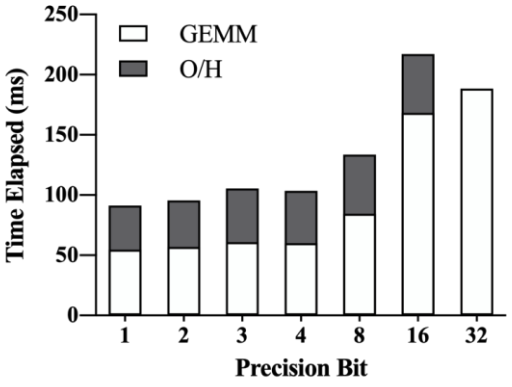

그림 2. Precision 별 Low-bit GEMM 성능

증가하였지만, 데이터를 불러오기 위해 메인 메모리 접근 횟수를 줄여 cache miss가 감소하였다.

precision 간의 trade-off를 관찰하기 위해 ResNet-20의 모든 계층을 같은 precision으로 학습시켰다. 그림 4의 y축 값이 작을수록 효율적인 precision이다. 8bit 이하의 precision의 경우 full precision 모델과 비교해 정확도가 감소하는 것에 비해 모델 수행 시간이 많이 감소하여 대체적으로 낮은 값을 가진다. 하지만, 1bit의 경우 그림 2에서 가장 좋은 성능을 보임에도 불구하고 급격하게 정확도가 떨어지는 구간이기 때문에 효율적이지 않은 것을 알 수 있다.

4.3. 계층 별 Quantization을 적용한 CNN 성능 분석

그림 5는 계층 별로 다양한 precision을 사용하였을 때와 동일한 precision을 사용하였을 때 유사한 메모리 사용량에서 ResNet-20의 정확도를 비교하여 나타냈다. 이렇게 다양한 precision을 사용하여 계층별 동일한 precision을 사용하는 모델 대비 성능을 향상시켰고, full precision을 사용하여 학습시킨 모델과 비교하는 실험을 하여 표 2와 같은 결과가 도출되었다.

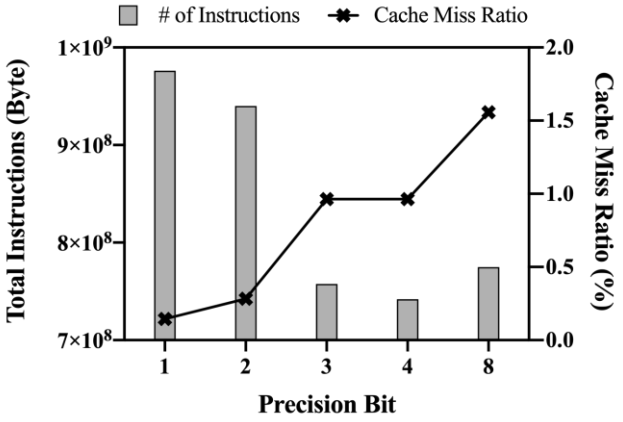

그림 3. Precision별 전체 instruction 크기와 Cache Miss Ratio

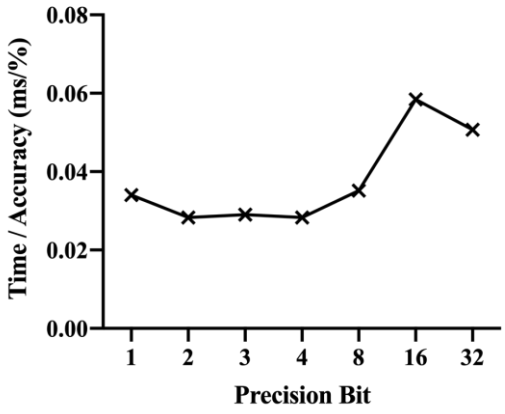

그림 4. Precision별 Time과 Accuracy간의 Trade-off

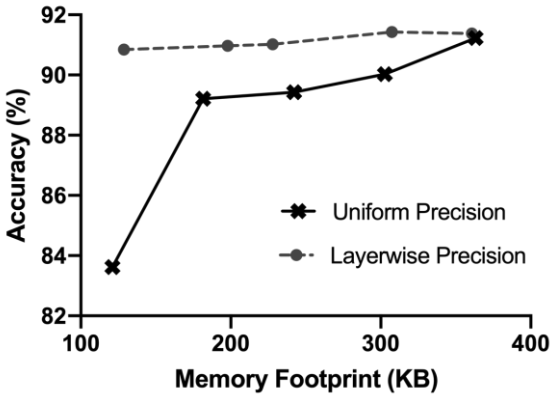

그림 5. 다양한 Precision을 적용한 ResNet-20의 Memory footprint와 Accuracy 간 Trade-off 비교

표 2. Layerwise Precision 모델과 Full Precision 모델의 비교

Memory Footprint (KB)	121	182	242	363
Accuracy (%)	90.85	90.97	91.02	91.38
Speed-up	1.178x	1.167x	1.162x	1.138x
Reduction Rate	15.99x	10.67x	7.99x	5.33x

5. 결 론

본 논문에서는 ARM기반 모바일 장치에서 딥 러닝을 사용할 때 계층별로 다양한 precision을 사용하였고, 이를 효율적으로 활용하기 위해서 low-bit GEMM 연산 과정을 구현하였다. 또한 이와 같은 과정을 통해 모델의 정확도와 실제 성능 및 메모리 사용량의 trade-off를 보여줬다. 121KB의 메모리 사용량을 가진 모델의 경우 정확도가 1.08% 감소하였지만, 기존에 비해 1.18배 성능이 개선되고 메모리 사용량이 15.99배 감소한다.

참고 문헌

[1] M. Rastegari, et al. "Xnor-net: Imagenet classification using binary convolutional neural networks." European Conference on Computer Vision. Springer, Cham, 2016.

[2] S. Zhou, et al. "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients." arXiv preprint arXiv:1606.06160 (2016).

[3] Zhou, Yiren, et al. "Adaptive quantization for deep neural network." Thirty-Second AAAI Conference on Artificial Intelligence. 2018.

[4] https://github.com/ARM-software/ComputeLibrary

[5] https://github.com/Tencent/PocketFlow