딥러닝 컴파일러 성능비교

유미선O, 김영주, 이제민, 김태호
한국전자통신연구원
msyu@etri.re.kr, kr.yjkim@etri.re.kr, leejaymin@etri.re.kr, taehokim@etri.re.kr

Performance Comparison of Deep Learning Compilers

Misun YuO, Young-Joo Kim, Jaemin Lee, Taeho Kim
Electronics and Telecommunications Research Institute

요 약

딥러닝 모델은 이미지 분류, 음성 인식, 게임 등 다양한 분야에서 인간의 능력을 넘어서는 높은 성능을 보이며 응용 분야를 점차 확대해 가고 있다. 이에 따라 딥러닝 모델을 실행하는 플랫폼도 고성능 서버에서 모바일 임베디드 시스템으로 확대되어 가고 있으며, 플랫폼의 종류 또한 다양화되고 있다. 딥러닝 컴파일러는 딥러닝 모델을 하드웨어 백엔드 코드로 자동 변환해 주는 도구이며, 하드웨어 플랫폼이 다양화 되면서 활용도가 점차 높아지고 있다. 본 논문에서는 딥러닝 컴파일러 선택 및 개발에 도움을 주기 위해, 대표적인 딥러닝 컴파일러인 TVM과 GLOW의 성능을 7종의 최신 CNN 기반 모델을 대상으로 측정한 결과를 제시한다.

1. 서 론

딥러닝 신경망 모델 (이하 딥러닝 모델)은 이미지 분류, 음성 인식, 자연어 처리 등의 다양한 분야에서 활용되고 있으며, 게임과 같은 특정 분야에서는 사람의 능력을 뛰어 넘는 성능을 보여주고 있다. 딥러닝 모델을 활용 가능한 응용 분야가 넓어짐에 따라 클라우드가 아닌 모바일 디바이스나 임베디드 시스템 내에 딥러닝 모델을 배치(deploy)하여 실행하고자 하는 요구도 커지고 있다.

딥러닝 모델을 특정 디바이스 상에서 효율적으로 동작시키기 위해서는 동작시킬 딥러닝 모델을 타겟 디바이스에서 최적의 속도와 정확도를 낼 수 있는 머신 코드로 변환해야 한다. 이러한 코드 변환 작업을 자동으로 지원해주는 도구를 딥러닝 컴파일러라고 한다. 그림 1은 딥러닝 컴파일러의 개념도를 보여준다.

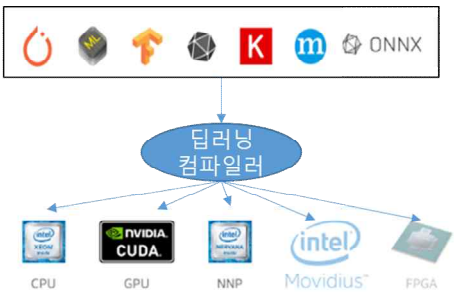

그림 1. 딥러닝 컴파일러 개념도

그림 1에서 볼 수 있듯이 딥러닝 컴파일러는 다양한 딥러닝 플랫폼에서 학습된 딥러닝 모델을 입력으로 받아 특정 하드웨어에서 동작 가능한 머신 코드 (또는, 백엔드 코드)를 자동으로 생성한다.

최근 제안된 XLA[4], TVM[5], Glow[6]와 같은 딥러닝 컴파일러들은 TensorFlow[1], Pytorch[2], MxNet[3]에서 생성된 모델이나 ONNX와 같은 표준 형식으로 작성된 모델을 입력으로 하여 LLVM이나 OpenCL과 같은 CPU 및 GPU용 백엔드 코드를 생성한다.

본 논문에서는 7종의 동일한 딥러닝 모델을 이용하여 최근에 활발하게 연구가 진행되고 있는 대표적인 두 종류의 딥러닝 컴파일러인 TVM과 GLOW의 성능을 비교한 결과를 제시한다.

2. 딥러닝 컴파일러의 구성

그림 2는 TVM과 GLOW와 같은 딥러닝 컴파일러의 구조를 보여준다.

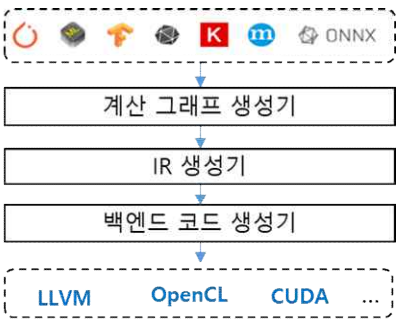

그림 2. 딥러닝 컴파일러 구조

2.1 계산 그래프 생성기

계산 그래프 생성기는 다양한 딥러닝 플랫폼에서 생성된 딥러닝 모델을 로딩하여 그래프 구조로 재구성한 후, 모델에서 명시한 오퍼레이터들을 프리미티브 오퍼레이터로 변환하여 계산 그래프를 생성한다. 그래프가 생성된 후에는 불필요한 노드 제거, 오퍼레이터 결합 (operator fusion), 양자화(Quantization), 상수 폴딩 (constant-folding)과 같은 그래프 최적화를 기법들을 적용하여 최종적으로 최적화 된 그래프를 생성한다.

2.2 IR 생성기

계산 그래프 생성기가 생성한 그래프는 하드웨어에 독립적인 정보 (오퍼레이터 간 입출력 관계, 오퍼레이터의 입출력 데이터의 크기 및 값)만을 표현할 수 있다. 그러나 컴파일러는 딥러닝 모델의 실행 성능을 높이기 위해 모델이 동작할 하드웨어의 아키텍처를 고려한 '하드웨어 의존적 최적화'를 필수적으로 수행해야 한다. 그러므로 계산 그래프를 하드웨어 의존적 최적화를 적용 가능한 형태인 중간 표현 (IR: Intermediate Representation)으로 변환하는 과정이 필요하다.

참고로, IR은 입출력 변수 (또는 Tensor)들에 대한 선언과 변수들을 사용하는 연산식에 대한 정보를 포함하고 있다. 컴파일러는 생성된 IR을 바탕으로 메모리 할당, 병렬화, 연산식 실행 순서 결정 등의 하드웨어 의존적 최적화 작업을 수행한 후 코드를 생성한다.

요약하면, IR 생성기는 컴파일러가 하드웨어 의존적 최적화를 용이하게 수행하도록 하기 위해 계산 그래프를 IR로 변환하는 역할을 한다.

2.3 백엔드 코드 생성기

백엔드 코드 생성기는 IR을 기반으로 딥러닝 모델 워크로드가 배치될 타겟 하드웨어 (CPU, GPU, TPU 등)의 아키텍처에 최적화 된 백엔드 코드를 생성한다. 즉, 타겟 하드웨어 별로 다를 수 있는 병렬로 처리 가능한 데이터의 크기, 캐시 크기, 효율적으로 수행 가능한 딥러닝 오퍼레이터의 타입과 실행 순서 등을 참조하여 스케쥴 정보를 생성한 후, 해당 하드웨어에 최적화 된 백엔드 코드를 생성한다.

3. 성능 비교 및 분석

3.1 실험 방법

TVM과 GLOW 컴파일러를 이용하여 동일한 딥러닝 모델을 컴파일 한 후, 동일한 이미지를 입력으로 주었을 때 추론에 걸리는 시간을 구하였다. 백엔드는 'llvm'으로 설정하였으며, 컴파일러에서 제공하는 CPU 런타임을 이용하여 CPU 상에서 모델을 실행하였다.

모델의 입력은 그림 3과 같은 244x244 크기의 이미지이며, 컴파일 된 모델을 100번 실행하여 추론에 걸린 총 시간을 측정하였으며, 총 추론 시간을 바탕으로 1회 평균 추론 시간을 구하였다.

그림 3 입력 이미지 (244x244)

TVM의 자동 튜닝 (autotuning) 기능과 GLOW의 프로파일 기반 양자화 (quantization) 기능은 사용하지 않았으며, 두 컴파일러 모두 기본으로 제공하는 최적화 기능만을 사용하여 컴파일을 수행하였다.

3.2 실험 환경

컴파일 된 모델을 실행한 환경은 아래와 같다.

- 운영체제: Ubuntu 16.04 64bit
- 메모리: 32GB
- CPU: Intel Core i7-8700 CPU 3.70GHz
- LLVM 버전: 6.0
- TVM 버전: 2019/4/19 최종 업데이트 버전
- GLOW 버전: 2019/4/16 최종 업데이트 버전

3.2 대상 딥러닝 모델

ImageNet으로 훈련 된 7종의 최신 CNN 기반 모델을 사용하였으며 표 1은 실험에 사용한 모델의 특성을 보여준다. 제시 된 모델의 특성은 모델의 크기, 사용된 연산자의 수 (# Oper.), 컨볼루션 연산자의 수 (# Conv.), 행렬 곱 연산자의 수 (# Gemm) 이다. 컨볼루션 연산과 행렬 곱 연산은 다른 연산자에 비해 많은 연산을 필요로 하므로 모델 수행 시간에 큰 영향을 미칠 수 있는 연산자들이다.

표 1. 실험 대상 딥러닝 모델

모델명	크기 (MB)	# Oper.	# Conv.	# Gemm
VGG19	575	46	16	3
AlexNet	244	24	5	3
ResNet50 v2	103	174	53	1
DenseNet-121	33	910	121	0
MobileNet v2	13.6	155	54	0
ShuffleNet	5.3	202	49	1
SqueezeNet v1.3	5	66	26	0

GLOW의 경우 모든 딥러닝 모델을 ONNX Model Zoo [7]에서 다운로드 받아 컴파일에 사용하였으며, TVM의 경우 AlexNet과 DenseNet만 로딩 상의 오류로 Gluon Model Zoo[8]에서 제공하는 모델을 사용하고, 나머지는 GLOW와 동일한 모델을 사용하였다.

3.2 성능 비교 및 분석

표 2는 TVM과 GLOW에서 컴파일 된 각 모델의 실행 속도를 나타낸다. 컴파일 된 각 모델은 같은 입력 이미지에 대해 동일한 분류 결과를 생성하였다.

표 2. 컴파일 된 모델의 실행 속도

모델명	실행 속도 (ms)	
	TVM	GLOW
VGG19	447	437
AlexNet	24	77
ResNet50 v2	101	112
DenseNet-121	71	396
MobileNet v2	15	124
ShuffleNet	74	64
SqueezeNet v1.3	9	31

표 2에서 볼 수 있듯이 TVM은 AlexNet, ResNet50, DenseNet, MobileNet, SqueezeNet에서 GLOW보다 우수한 성능을 보였다. 특히, DenseNet과 MobileNet에서는 GLOW가 컴파일한 코드보다 TVM이 컴파일 한 코드가 5배 이상 빨랐다. GLOW의 경우는 VGG19와 ShuffleNet에서 2%-14% 정도로 빠르게 동작하는 코드를 생성했다. 즉, CPU 상에서 수행된 모델의 실행 속도만으로 판단하였을 때, 모델의 크기, 연산자의 개수 및 종류에 관계없이 TVM이 GLOW에 비해 전반적으로 GLOW보다 높은 성능을 보였다.

두 컴파일러의 동작 속도 차이는 계산 그래프 생성기에 의해 수행되는 그래프 최적화와 백엔드 코드 생성기에 의해 수행되는 하드웨어 의존적인 최적화에 의해 발생 가능하다.

그러나 TVM과 GLOW의 계산 그래프 생성기는 모두 operator fusion, constant-fondling과 같은 동일한 정적 그래프 최적화 기법을 사용하고 있으며, TVM 계산 그래프 생성기에서 수행하는 타일링 (tiling) 작업 또한 GLOW도 동일하게 수행하는 것으로 명시되어 있다[6].

그러므로 TVM과 GLOW의 성능 차이는 딥러닝 모델에 명시 된 고수준의 연산이 어떠한 백엔드 인스트럭션 리스트 (micro-kernal calls)로 변환 되는가의 차이에 의해 발생할 것으로 추정된다. TVM의 경우, 개발자에 의해 직접 작성된 라이브러리 (handcrafted micro-kernels)를 백엔드 코드 생성에 이용할 수 있도록 지원하고 있어 좀 더 최적화 된 백엔드 코드 생성이 가능하다.

4. 결론 및 향후 연구

본 논문에서는 TVM과 GLOW에 의해 컴파일 된 7종의 모델을 CPU 상에서 실행시켜 봄으로써 두 딥러닝 모델 컴파일러의 컴파일 성능을 측정하였다. 측정 결과, TVM이 GLOW보다 5종의 모델에서 더 좋은 성능을 보였으며, 이 중 2종 모델에서는 5배 이상 수행속도가 빠른 코드를 생성했다. 향후에는 CPU 뿐만이 아니라 GPU나 NPU상에서도 컴파일러의 성능을 비교해 보고, 각 플랫폼 상에서 컴파일러들이 성능차이를 발생시키는 원인에 대해 자세히 조사해 볼 계획이다.

Acknowledgement

이 논문은 2019년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.1711080972, 인공지능 시스템을 위한 뉴로모픽 컴퓨팅 SW 플랫폼 기술 개발)

참 고 문 헌

[1] Abadi, Martín, et al. "Tensorflow: A system for large-scale machine learning." OSDI, 2016.
[2] Pytorch, https://github.com/pytorch/pytorch
[3] Chen, Tianqi, et al. "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems." arXiv preprint arXiv:1512.01274, 2015.
[4] XLA, https://www.tensorflow.org/xla
[5] Chen, Tianqi, et al. "TVM: An automated end-to-end optimizing compiler for deep learning." OSDI, 2018.
[6] Rotem, Nadav, et al. "Glow: Graph lowering compiler techniques for neural networks." https://arxiv.org/pdf/1805.00907.pdf, 2019.
[7] ONNX Model Zoo, https://github.com/onnx/models
[8] Gluon Model Zoo, https://mxnet.incubator.apache.org/api/python/gluon/model_zoo.html