

논문 2023-60-10-4

범용 AI 컴파일러의 비공개 NPU 코드생성을 위한 공통 인터페이스 설계 및 검증

(Design and Verification of a Common Interface for Proprietary NPU Code Generation in General-purpose AI Compilers)

이 제 민*, 권 용 인*

(Jemin Lee and Yongin Kwon[Ⓞ])

요 약

본 논문에서는 NPU(Neural Processing Units)의 제조사별 비공개 백엔드 컴파일러와 범용 AI 컴파일러를 연결할 수 있는 공통 인터페이스를 제안한다. 이를 통해 다양한 AI 모델에 대한 지원과 기업 자산 보호가 가능하다. 제안된 인터페이스는 ONNX 표준을 따르며, ETRI의 NEST-C 컴파일러와 오픈엣지의 ENLIGHT 컴파일러를 해당 인터페이스로 통합했다. 실험 결과, 통합된 컴파일러는 ENLIGHT만을 사용한 것과 Resnet50과 MobileNetV2 두 종류의 모델에 대해서 100% 결과가 일치했다. 따라서 제안된 인터페이스는 NPU 백엔드 컴파일러의 범용성을 향상하면서도 비공개성을 유지할 수 있는 유용한 방안을 제공한다.

Abstract

This paper proposes a common interface for connecting proprietary back-end compilers of NPU (Neural Processing Units) manufacturers with general-purpose AI compilers, enabling support for various AI models while protecting corporate assets.

*비회원, 한국전자통신연구원(Electronics and Telecommunications Research Institute)

Ⓞ Corresponding Author(E-mail : yongin.kwon@etri.re.kr)

※ Acknowledgment 이 논문은 2023 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임(No. 2022-0-00454, 스마트 엣지 디바이스 SW 개발 플랫폼 개발).

Received : September 8, 2023 Revised : October 9, 2023

Accepted : October 10, 2023

The proposed interface adheres to the ONNX standard and integrates ETRI's NEST-C compiler with OPENEDGES's ENLIGHT compiler through this interface. Experimental results showed that the integrated compiler achieved 100% result consistency for two types of models, ResNet50 and MobileNetV2, compared to using ENLIGHT alone. Therefore, the proposed interface offers a valuable solution for enhancing the versatility of NPU back-end compilers while maintaining their proprietary nature.

Keywords: Neural processing unit, Compiler, Deep learning

I. 서 론

딥러닝 모델의 다양한 활용으로 에너지 소모를 줄이고 연산 효율을 증가시키기 위한 전용 하드웨어인 NPU(Neural Processing Units)의 개발이 활발히 이뤄지고 있다^[1]. CPU, GPU와 같은 전통적인 연산 장치들과 다르게 IP 사업을 주력으로 하는 대부분의 NPU 스타트업 제조사들은 내부 하드웨어를 최대한 노출시키지 않기 위해서 자체적인 백엔드 컴파일러를 개발하고 있다. 자체 개발된 NPU 백엔드 컴파일러는 하드웨어 정보를 최대한 활용해서 최적의 코드를 탐색하여 생성하므로 가장 최적화된 코드를 생성할 수 있는 장점이 있다^[2]. 하지만 제조사별 독자적으로 개발한 컴파일러는 TVM, Glow, MLIR, NEST-C와 같은 범용 오픈소스 AI 컴파일러들보다 다양한 AI 모델들을 지원하지 못하고 있다^[3~5]. 따라서 NPU 제조사 별로 자체 개발된 비공개 백엔드 컴파일러와 범용 AI 컴파일러 프론트엔드를 연결하여 비공개성을 보장하면서도 범용성을 향상할 방법이 필요하다.

본 연구에서는 NPU 제조사별로 개발한 컴파일러와 범용 AI 컴파일러를 연결하기 위한 공통 인터페이스를 제안한다. 제안된 인터페이스를 통해 기존 범용 AI 컴파일러를 재활용하여 다양한 응용을 지원하도록 범용성을 향상하며, NPU 하드웨어 내부 로직이 노출되는 저수준 코드생성 부분을 비공개 함으로써 기업의 자산을 보호할 수 있다.

제안된 공통 인터페이스는 대부분의 NPU 백엔드 컴파일러가 지원하는 ONNX 표준형식을 따르며, 제안된 인터페이스로 비공개 NPU 컴파일러가 2종의 테스트 AI 모델에 대해서 정확히 동작함을 보였다.

실험에 사용한 오픈소스 범용 AI 컴파일러는 한국전자통신연구원에서 개발한 NEST-C이며, 비공개 NPU 및 컴파일러는 오픈넷지에서 개발된 ENLIGHT이다. Xilinx Alveo U280 FPGA 상에서 오픈넷지 ENLIGHT에서 지원하는 Resnet50^[6]과 MobileNetv2^[7] 2종 모델에 대해서 정확도 검증 실험 수행했다. 실험 결과 제조사 비공개 컴파일러인 ENLIGHT만을 이용했을 때와 NEST-C와 ENLIGHT를 결합했을 때의 추론 결과는 100% 일치했음을 보였다.

II. 관련연구

1. ONNX

ONNX(open neural network exchange)는 딥러닝 모델을 서로 다른 딥러닝 프레임워크 간에 공유하고 실행하기 위해서 만들어진 개방형 표준형식이다. ONNX는 AI 모델을 공통 표현으로 변환하여 프레임워크 간의 호환성을 높여준다. 이러한 공통 표현은 AI 모델의 구조와 파라미터를 설명하며, 여러 다양한 프레임워크에서 해석되어 실행될 수 있다.

2. NEST-C

NEST-C는 한국전자통신연구원에서 개발한 범용 AI 컴파일러이다. 그림 1은 NEST-C 컴파일러가 하드웨어 타겟에 맞춰서 코드를 최적화하는 과정을 나타낸다. NEST-C는 먼저 ONNX 등의 형태로 표현된 AI 모델을 입력으로 받아 타겟 하드웨어에 독립적인 그래프 수준 최적화를 수행한다. 그다음, 머신 코드 수준의 타겟 하드웨어에 의존적인 최적화를 수행한다. NEST-C 컴파일러에서 직접 지원하는 NPU 하드웨어는 EVTA와 에임퓨처의 DQ1이 있다¹⁾. 두 하드웨어의 경우 저수준 코드생성 부분까지 모두 공개되어있다.

3. ENLIGHT NPU

ENLIGHT NPU²⁾는 객체 검출(Object Detection), 이미지 분류(Image Classification)와 같은 시각 분야 응용을 위해 최적화된 하드웨어 가속기로 ENLIGHT NPU는 중간 활성화 데이터의 DRAM 트래픽을 줄이기 위해 그룹화된 레이어 분할과 스케줄링과 같은 최적화된 네트워크 모델 컴파일러를 포함하고 있다. 이러한 기능들은 사용자들이 시장에서 요구하는 응용에 맞게

코어 크기와 성능을 쉽게 조정할 수 있는 유연성을 제공한다. 또한, ENLIGHT NPU는 4bit와 8bit의 혼합 정밀도 양자화 기술을 최초로 도입하여 용량, 전력 소비, 성능, 그리고 DRAM 대역폭 측면에서 효율성을 높였다.

그림 2는 ENLIGHT NPU와 컴파일러의 구조도를 나타낸다. 딥러닝 모델이 입력으로 들어오고 ENLIGHT 컴파일러에 의해서 모델이 변환되고 실행 커맨드들이 생성된다. 전처리, 후처리와 같은 기본 코드들은 NN App에 저장되어 메인 함수 코드로 관리된다. 이러한 데이터들은 PCIe 인터페이스를 통해서 Xilinx Alveo U280 FPGA의 DRAM에 저장된다. U280에는 NPU IP와 기타 컨트롤러 IP들이 합성되어 올라가 있는 상태이다. 따라서 해당 실행 명령어와 딥러닝 모델 그리고 입력 이미지들을 순차적으로 처리하여 최종 추론 결과를 계산하는 과정을 수행한다.

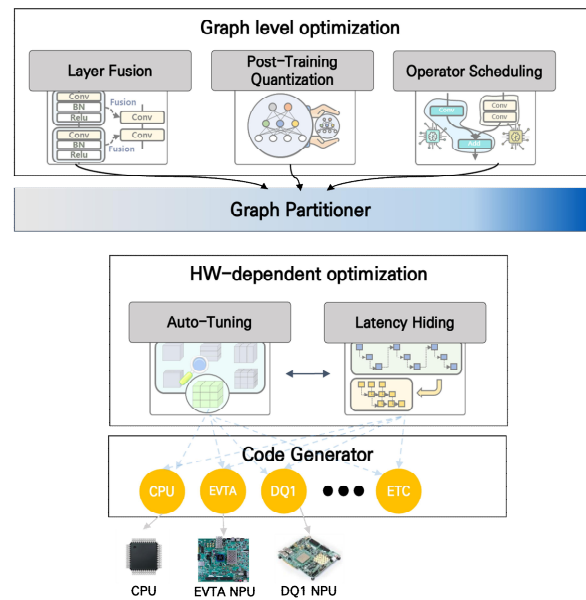


그림 1. NEST-C 컴파일러의 구조도
Fig. 1. Overview of the NEST-C Compiler.

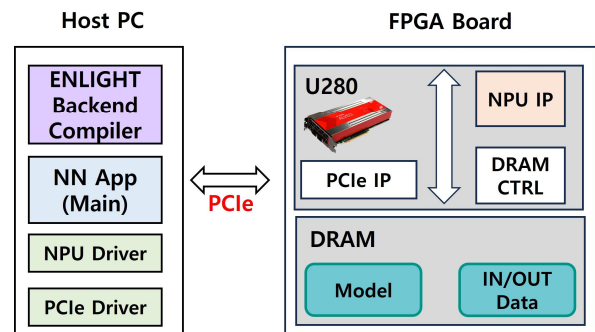


그림 2. ENLIGHT NPU와 컴파일러의 구조도
Fig. 2. Overview of the ENLIGHT NPU and Compiler.

1) <https://github.com/etri/nest-compiler>
2) <https://www.openedges.com/ko/npu>

표 1. ENLIGHT 단독 추론 결과와 NEST-C와 ENLIGHT를 통합한 추론 결과의 비교

Table 1. Comparison of Inference Results Between ENLIGHT and Integrated NEST-C and ENLIGHT.

	NEST-C (Intel i7-8700)		ENLIGHT (U280 FPGA)		NEST-C + ENLIGHT (U280 FPGA)		Original Model	Sanity Check
	Accuracy (FP32)	Latency	Accuracy (8bit)	Latency	Accuracy (8bit)	Latency	Accuracy (FP32)	
MobileNet-v2.7	70.6%	59.9ms	68.624%	9ms	68.624%	9ms	70.6%	100%
Resnet50-v2	79.26%	136.9ms	73.038%	107ms	73.038%	107ms	79.26%	100%

II. 제안하는 통합 인터페이스 구조

ONNX형태로 저장된 이미 학습된 모델 파일을 범용 컴파일러인 NEST 컴파일러의 입력으로 받는다. 하드웨어 전용 컴파일러와의 연결을 위해서 하드웨어 독립적인 최적화를 수행한 후에 메타 형식인 yaml 형식에 따라 모델이 직렬화되어 저장된다. 저장된 메타 파일을 다시 ONNX 포맷에 맞추기 위한 정규화 작업을 수행한 후 ENLIGHT 컴파일러의 입력을 위한 최종 ONNX를 생성한다.

제안하는 컴파일러 통합 인터페이스 구조를 포함한 NEST-C 컴파일러 구조는 그림 3과 같으며, 각각에 대해서 상세히 설명하면 다음과 같다.

- Partition to yaml: 입력 모델을 원하는 크기의 파티션으로 구성하기 위한 정보를 담고 있는 yaml 파일을 생성한다. 파티션은 모델 전체가 될 수 있고 특정 연속된 연산자들만을 포함할 수도 있다. yaml 파일의 구성은 그래프 중간표현에 기술된 연산자의 이름을 담고 있다. 해당 yaml 파일을 이용해서 파티션에 맞는 코드를 생성하거나 전용 백엔드 컴파일러와 연동된다.
- ONNX IR Legalizer: 그래프 중간표현에서 다시 ONNX 중간표현으로 변환하기 위해서 두 중간표현 사이에서 표현상의 차이를 통일하는 정규화 작업을 수행한다.
- Sanity Checker with ONNXRuntime: 정상적으로 ONNX로 변환되었는지를 확인하기 위해 범용 CPU 용 컴파일러인 ONNXRuntime을 이용해서 모델의 결과값을 확인한다.
- Partial ONNX Exporter: 메타파일(yaml)을 ONNX 형식에 맞춰서 파일로 저장하는 기능을 담당한다.

본 논문에서는 비공개 백엔드 컴파일러인 ENLIGHT와 연동되는 과정을 통합하여 개발했다. 이러한 방식으로 제3의 비공개 컴파일러도 통합되어 개발될 수 있다.

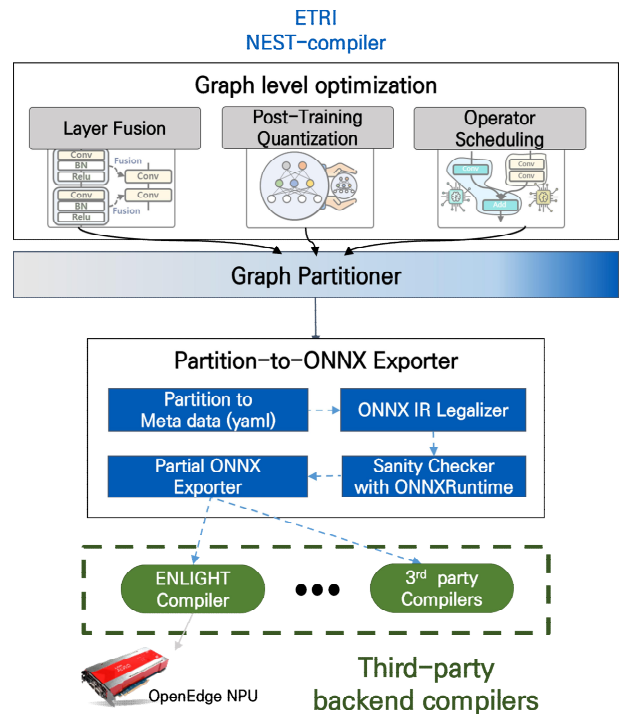


그림 3. 통합 인터페이스를 통한 범용 AI 컴파일러와 비공개 컴파일러 간의 통합 구조

Fig. 3. Integrated Structure Between General-Purpose AI Compiler and Proprietary Compiler Through Unified Interface.

III. 실험

제안하는 ONNX 기반 공통 인터페이스를 통해서 범용 AI 컴파일러와 비공개 NPU 컴파일러가 정상적으로 연동되었는지를 실험을 통해서 입증한다. 비교 평가를 위해서 Resnet50과 MobileNetV2 기존 모델들을 ENLIGHT 백엔드 컴파일러로 컴파일한 것과 NEST-C와 결합하여 컴파일된 결과에 대해서 각각 이미지넷 검증 데이터셋 5만 장을 이용해서 TOP1 정확도와 추론 지연시간을 측정했다.

표 1은 실험 결과를 나타내며, Resnet50과 MobileNetV2에 대해서 논문에 기록된 원본 FP32 정확도 및 각각의

컴파일러를 이용해서 실행한 모델의 정확도를 나타낸다. 정확도에 있어서 ENLIGHT NPU에서 실행될 때는 32bit 정밀도 모델이 8bit로 양자화되므로 이로 인한 정확도 감소가 존재한다. 감소한 정확도를 기준으로 두 컴파일러 실행 결과는 100% 모두 일치함을 보인다.

추론 지연시간의 경우 NEST-C 컴파일러만을 이용해서 실행할 경우 호스트 컴퓨터로 활용된 CPU로 추론되어 MobileNetV2와 Resnet50에 대해서 각각 59.9ms와 136.9ms의 추론시간을 달성했다. ENLIGHT 백엔드 컴파일러와 결합하여 ENLIGHT NPU에서 실행할 경우 9ms와 107ms를 MobileNetV2와 Resnet50에 대해서 달성했다. 이는 ENLIGHT 백엔드 컴파일러만을 이용해서 추론했을 때와 같은 지연시간이다. 따라서 정확도 및 추론 지연시간에 있어서 같은 결과를 나타내므로 정상적으로 두 컴파일러가 통합되었음을 알 수 있다.

IV. 결 론

본 논문에서는 제조사별로 비공개된 NPU 백엔드 컴파일러와 범용 AI 컴파일러 간을 연동할 수 있게 하는 통합 인터페이스를 제안했다. 제안한 인터페이스는 다양한 AI 모델을 지원하고 기업의 민감한 자산을 보호할 수 있도록 지원한다. 제안한 인터페이스는 ONNX 표준을 준수하였고, ETRI의 NEST-C와 오픈엣지의 ENLIGHT 컴파일러를 제안한 인터페이스를 통해서 결합했다. 실험 결과 제안한 통합 인터페이스를 사용한 컴파일러는 ENLIGHT만 사용했을 경우와 Resnet50과 MobileNetV2 모델에 관한 결과가 100% 일치하였다. 따라서 제안된 인터페이스는 NPU 백엔드 컴파일러와 범용 AI 컴파일러를 연동하기 위한 인터페이스로 타당함을 보였다.

향후 연구로는 범용 AI 컴파일러와 비공개 NPU 백엔드 컴파일러를 결합했을 때 단순히 호환성을 보장하는 것을 넘어서 추론 지연시간 감소 또는 양자화 정확도 향상 등과 같은 시너지를 달성할 수 있도록 할 예정이다.

REFERENCES

- [1] Chen, Yiran, et al. "A survey of accelerator architectures for deep neural networks." *Engineering* 6.3 (2020): 264-274.
- [2] Joo Hyoung Cha, Yongin Kwon, and Jemin Lee, "Profile-based Optimization for Deep Learning on Heterogeneous Multi-core CPUs", *Journal of The Institute of Electronics and Information Engineers* 60.7 (2023): 40-49.
- [3] Li, Mingzhen, et al. "The deep learning compiler: A comprehensive survey." *IEEE Transactions on Parallel and Distributed Systems* 32.3 (2020): 708-727.
- [4] Zhang, Hongbin, et al. "Compiler Technologies in Deep Learning Co-Design: A Survey." *Intelligent Computing* (2023).
- [5] Lee, Jemin, et al. "Quantune: Post-training quantization of convolutional neural networks using extreme gradient boosting for fast deployment." *Future Generation Computer Systems* 132 (2022): 124-135.
- [6] He, Kaiming, et al. "Deep Residual Learning for Image Recognition." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [7] Howard, Andrew G., et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications." *arXiv preprint arXiv:1704.04861*, 2017.