Journal of The Institute of Electronics and Information Engineers Vol.61, NO.1, January 2024 https://doi.org/10.5573/ieie.2024.61.1.23
ISSN 2287-5026(Print) / ISSN 2288-159X(Online)

논문 2024-61-1-3

엣지 딥 러닝 가속기의 추론 성능 분석

(Performance Analysis of Deep Learning Accelerator for Edge Inference)

박 시 형*, 권 용 인**, 이 제 민**

(Sihyeong Park, Yongin Kwon, and Jemin Lee©)

요 약

엣지 장치에서 딥 러닝 기반 추론을 위해 추론 가속기가 탑재되고 있다. 딥 러닝 추론 가속기를 통해 연산 성능과 에너지 효율을 증가시킬 수 있다. 하지만 가속기에 최적화되지 않은 모델 구조와 설정을 사용하면 메모리 접근 등의 오버헤드로 인해 최적 성능을 낼 수 없다. 본 논문에서는 사전 학습된 MobileNet v2, ResNet50 v1 모델을 사용해 NVIDIA Jetson에서 Graphic Processing Unit (GPU)와 Deep Learning Accelerator (DLA)의 추론 성능을 분석하였다. 실험을 통해 DLA에 최적화되지 않은 모델을 실행하면 GPU보다 최대 5.1배 추론 시간이 증가함을 보였다. 특히, 프로파일링을 통해 DLA에서 지원하지 않는 연산을 GPU로 폴백 (fallback)하는 과정의 오버헤드로 추론 시간이 증가함을 보였다.

Abstract

Inference accelerators are currently being utilized for deep learning inference on edge devices. Deep learning inference accelerators can enhance computational performance and energy efficiency. However, it is important to note that optimal performance cannot be achieved if the model structure and settings (e.g. hyperparameters) are not optimized for the accelerator, which can result in overheads, such as frequent memory access. This study analyses the inference performance of the Graphic Processing Unit (GPU) and the Deep Learning Accelerator (DLA) on NVIDIA Jetson with pre-trained MobileNet v2 and ResNet50 v1 models. The results of our experiments show that running non-optimized models on the DLA results in up to 5.1 times longer inference time compared to the GPU. This paper showed through profiling that the increase in inference time is due to the overhead of GPU fallback to perform operations not supported by DLA.

Keywords : Deep learning, Deep learning accelerator, Inference, Edge device, Performance analysis

*비회원, 한국전자기술연구원(Korea Electronics Technology Institute)
**비회원, 한국전자통신연구원(Electronics and Telecommunications Research Institute)
© Corresponding Author(E-mail : leejaymin@etri.re.kr)
※ 이 논문은 2023년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임(No. RS-2023-00277060, 개방형 엣지 AI 반도체 설계 및 SW 플랫폼 기술개발)

Received ; September 22, 2023 Revised ; October 16, 2023
Accepted ; October 17, 2023

Ⅰ. 서 론

엣지 장치에서 딥 러닝 기반의 온 디바이스 추론의 필요성이 증가함에 따라 이러한 연산을 가속하기 위한 Edge Tensor Processing Unit (TPU), Coral, NVIDIA Deep Learning Accelerator (DLA), Graphic Processing Unit (GPU) 등의 가속 하드웨어가 탑재되고 있다[1~3].

엣지 장치는 제한된 하드웨어 자원으로 동작하므로 효율적인 딥 러닝 연산이 필요하다. 특히 딥 러닝 가속기에서의 모델 연산을 위해서는 가속기의 성능 분석과 최적화된 모델 구조가 필요하다. 가속기를 사용하더라도 최적화되지 않은 모델과 설정을 사용하면 엣지 장치와 가속기 간의 파라미터 전달, 메모리 복사, 동기화 등의 작업으로 연산 효율성 저하와 더 많은 에너지를 사용할 수도 있다[4]. 기존 가속기를 사용한 연구[1~3]에서는 가속기를 사용한 추론 성능 향상 방법을 주로 다루고 있지만 본 논문에서는 가속기의 설계, 사용에서 메모리 복사 등의 오버헤드와 모델 구조에 대한 고려가 필요함을 실험을 통해 보인다.

본 연구에서는 NVIDIA Jetson AGX Orin에서 딥 러닝 연산 가속을 위한 GPU와 DLA에 대한 성능 분석을 수행하였다. GPU와 DLA의 성능 분석을 위해 사전 훈련된 ResNet50과 MobileNet v2 모델을 사용해 이미지 추론 시간을 분석하였다. 또한, 본 논문에서는 엣지 장치에서 사용할 수 있는 사전 학습된 모델을 사용해 GPU와 DLA에서의 추론 과정을 프로파일링해서 추론 과정에 대한 하드웨어 사용 분석을 수행하였다. 실험을

Copyright © The Institute of Electronics and Information Engineers.
This is an Open-Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (http://creativecommons.org/licenses/by-nc/3.0) which permits unrestricted non-commercial use, distribution and reproduction in any medium, provided the original work is properly cited.

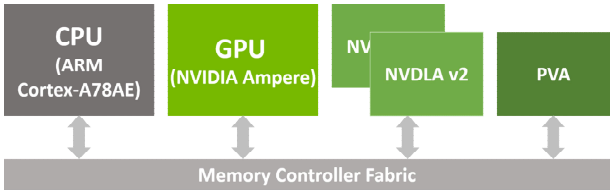

그림 1. NVIDIA Jetson 하드웨어 구조
Fig. 1. Hardware Structure of NVIDIA Jetson.

통해 DLA에 최적화된 모델을 사용하지 않는 경우 이미지 추론 시간이 GPU보다 최대 5.1배 증가함을 보였다. 이는 DLA가 지원하지 않는 연산을 처리하기 위한 추가적인 오버헤드로 인해 추론 시간이 증가하였다.

Ⅱ. 배경지식

대표적 엣지 장치인 NVIDIA Jetson은 딥 러닝 연산을 가속하기 위해 그림 1과 같은 하드웨어를 탑재하고 있다. NVIDIA Jetson은 NVIDIA Ampere 아키텍처 기반의 GPU와 딥 러닝 워크로드를 위한 딥 러닝 가속기 DLA 및 이미지 처리와 컴퓨터 비전 알고리즘 가속을 위한 Programmable Vision Accelerator (PVA) 등의 하드웨어를 탑재하고 있다.

DLA는 딥 러닝 추론 가속을 위한 개방형 하드웨어*로 다음과 같이 구성되어 있다:

- Convolution Core: 컨볼루션 최적화 엔진
- Single Data Processor: 활성 (activation) 함수를 위한 single-point 룩업 (lookup) 엔진
- Planar Data Processor: 풀링 (pooling)을 위한 2차원 평균화 엔진
- Channel Data Processor: 고급 정규화 기능을 위한 다중 채널 평균화 엔진
- Dedicated Memory and Data Reshape Engines: 텐서 재형성 (reshape) 및 복사 작업을 위한 메모리 간 변환 가속 엔진

NVIDIA Jetson의 DLA는 TensorRT**, Tensorflow, PyTorch로 실행할 수 있으며, 본 논문에서는 TensorRT를 사용하였다.

DLA에서 모델을 실행하기 위해 CUDA를 확장한 cuDLA를 사용해 DLA 하드웨어를 제어한다. DLA의 실행을 위해 GPU와 DLA 간 공유 버퍼를 사용해 데이터 전송 및 동기화를 수행한다. 본 논문에서 사용하는

* https://github.com/nvdla
** https://developer.nvidia.com/tensorrt

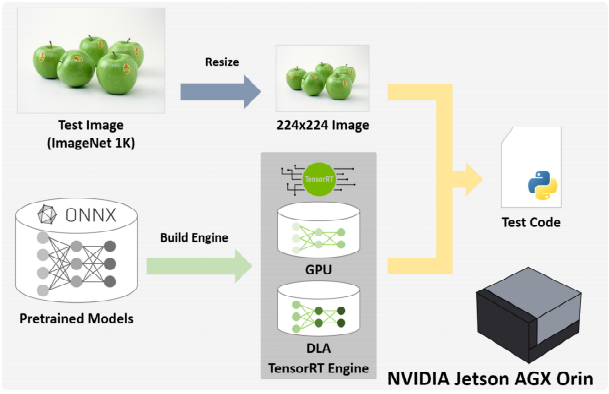

그림 2. DLA 추론 성능 평가 과정
Fig. 2. Evaluation Process for DLA Inference Performance.

표 1. TensorRT 엔진 빌드 결과
Table 1. TensorRT Engine Build Results.

	# of Total Layer	# of GPU Fallback Layer
MobileNet v2	104	10
ResNet50 v1	179	10

NVIDIA Jetson AGX Orin은 DLAv2를 탑재하고 있으며 FP16, INT8을 지원하고 레이어별 차원, 커널 크기와 지원 가능 연산 등의 제한이 있다. DLA에서 실행 불가능한 연산은 GPU로 폴백 (GPU Fallback)을 통해 수행된다.

본 논문에서는 NVIDIA Jetson AGX Orin의 GPU와 DLA를 사용해 딥 러닝 모델 기반 추론 성능 분석 결과를 다룬다.

Ⅲ. TensorRT 엔진 빌드

딥 러닝 기반 추론 성능 측정을 위해 그림 2와 같이 사전 학습된 딥 러닝 모델을 TensorRT (trtexec)를 사용해 TensorRT 엔진으로 빌드하였다. 실험을 위해서 ONNX Model Zoo***의 FP32로 사전 학습된 MobileNet v2[5]와 ResNet50 v1[6] 모델을 사용하였다.

엔진 빌드 과정에서 TensorRT가 자동으로 GPU와 DLA에서 실행할 수 있도록 레이어 구조 및 입출력 크기 등을 최적화한다. 특히, DLA에서의 모델 연산 최적화를 위해 일부 레이어의 정밀도가 FP32에서 FP16으로 변경된다. 또한, DLA의 하드웨어 제약으로 인해 실행 불가능한 레이어를 엔진 빌드 과정에서 GPU로 폴

*** https://github.com/onnx/models

표 2. 실험 환경
Table 2. Experimental Environments.

		Specification
HW	CPU	ARM Cortex-A78AE (12-core)
	Memory	32 GB (CPU, GPU shared)
	GPU	NVIDIA Ampere GPU (1792-core)
	Accelerator	DLAv2 (x2)
SW	OS	Ubuntu 20.04 (JetPack 5.1.1)
	CUDA	11.4 (pycuda 2022.2.2)
	Python	3.8.10
	TensorRT	8.5.2.2
	Profiler	NVIDIA Nsight Systems 2022.5.2

백을 통해 실행할 수 있도록 'allowGPUFallback' 옵션을 사용하였다. 엔진 빌드 결과는 표 1과 같다. DLA에서 각 모델의 Reduce, Shape, Constant, Gather, Shuffle, Concatenation 등의 레이어는 실행 불가능하여 GPU로 폴백되어 수행된다. MobileNet v2의 경우에는 10.4%, ResNet50 v2은 17.9%의 레이어가 GPU로 폴백되었다.

Ⅳ. 실험 환경 및 결과

1. 실험 환경

본 논문에서는 NVIDIA Jetson AGX Orin Developer Kit 32GB를 사용해 GPU와 추론 가속기 DLA의 추론 성능을 분석하였다. 실험 환경은 표 2와 같다. 추론 성능 측정은 그림 2와 같이 파이썬을 사용해 ImageNet 데이터 세트*의 이미지를 1,000회 반복하여 이미지 분류 시간을 측정하였다. 이미지 분류 시간 측정은 GPU 메모리 영역으로 데이터를 복사해 이미지 추론을 수행하고, 추론 결과를 다시 CPU의 메모리 영역으로 복사하는 과정을 포함한다. 실험을 위해 장치의 동작 모드는 'MAXN'으로 설정하였다. 'MAXN' 모드는 Jetson의 모든 GPU와 GPU 및 메모리를 최대 동작 주파수로 동작시킨다. 추론 중의 하드웨어 프로파일링을 위해 NVIDIA Nsight Systems를 사용하였다.

2. 실험 결과

TensorRT를 통해 변환된 MobileNet v2와 ResNet50 v1 엔진을 NVIDIA Jetson AGX Orin의 GPU와 DLA로 이미지 추론을 수행한 결과는 그림 3과 같다.

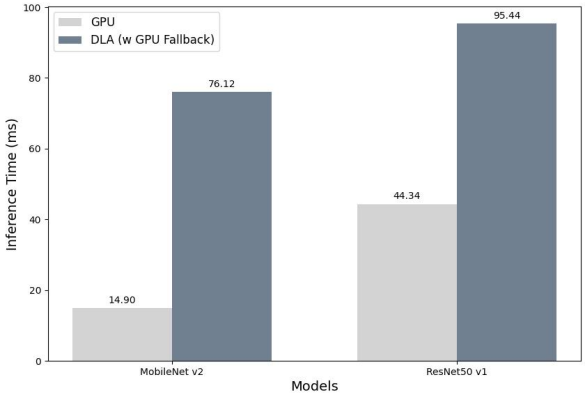

그림 3. 추론 성능 결과
Fig. 3. Inference Performance Results.

MobileNet v2의 경우 GPU와 DLA에서 추론에 든 시간은 각각 14.90, 76.12ms로 DLA에서 수행하였을 때 약 5.1배 증가하였다. ResNet v1의 추론 시간은 각각 44.34, 95.44ms로 DLA에서 수행할 때 GPU의 수행 시간보다 약 2.1배 증가하였다. 엔진 빌드 과정에서 DLA에서 실행하기 위해 정밀도를 FP32에서 FP16으로 감소시켰음에도 불구하고 시간이 증가하였다.

그림 4와 5는 GPU와 DLA에서 추론을 수행할 때 NVIDIA Nsight Systems로 하드웨어 프로파일링 결과를 나타낸다. 그림 4와 같이 GPU에서 모델을 실행하기 위해 TensorRT/CUDA API로 메모리 할당, 데이터 복사와 추론을 한다. 반면 DLA에서 모델을 실행하기 위해서는 그림 5와 같이 CUDA 메모리 할당 과정 이외에도 cuDLA를 통해 DLA 사용을 위한 모델 최적화와 메모리 할당, DLA 버퍼로의 데이터 복사 등을 추가로 수행한다. 또한, DLA에서 실행 불가능한 레이어를 GPU로 폴백해서 GPU와 DLA 사이의 데이터 스트림 생성과 데이터 복사 과정이 추가로 발생한다. GPU에서 추론하였을 때는 실행 이후에 결과 동기화를 위해 약 2.9ms가 소요되었지만 DLA에서 실행할 때 약 4배 이상 증가한 12.3ms 가 소요되었다.

실험 결과를 바탕으로 DLA에 최적화되지 않은 모델을 실행하면 GPU를 사용했을 때보다 더 느린 추론 시간이 걸릴 수 있음을 보였다. 이러한 문제를 해결하기 위해 사전 학습된 모델과 같이 DLA 구조에 최적화되지 않은 모델을 실행하기 위해서는 TensorRT가 제공하는 기본 최적화 기법 이외에도 DLA에 적합하도록 레이어 구조를 변경하거나 양자화와 같은 최적화 기술을 통해 GPU 폴백을 줄이는 것이 필요하다.

* https://www.image-net.org/

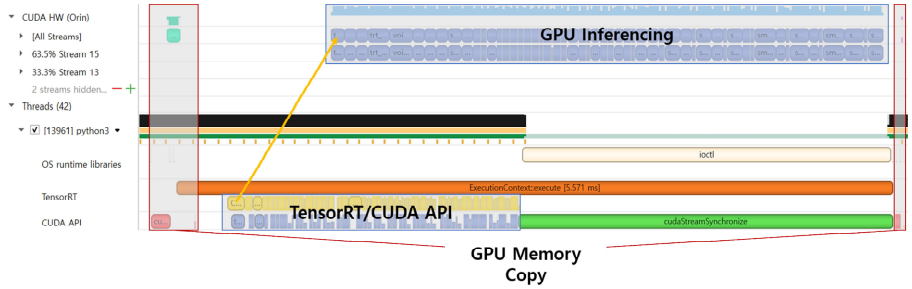

그림 4. GPU 추론 프로파일링 결과
Fig. 4. GPU Inference Profiling Results.

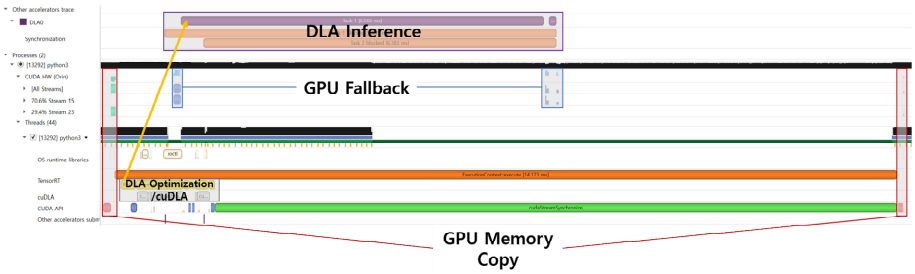

그림 5. DLA 추론 프로파일링 결과
Fig. 5. DLA Inference Profiling Results.

V. 결론 및 향후 연구

본 논문에서는 NVIDIA Jetson AGX Orin에 탑재된 GPU와 딥 러닝 추론 가속기인 DLA의 성능을 분석하였다. 사전 학습된 MobileNet v2와 ResNet50 v1 모델을 사용해 각 장치에서 추론을 수행한 결과 최대 약 5.1배 정도 DLA가 GPU보다 추론 시간이 증가함을 보였다. 이를 통해 DLA에 최적화된 실행을 위해서는 가속기 구조에 맞춰 모델 설계 및 최적화가 필요하고 메모리, GPU, DLA 간 데이터 복사 작업을 최소화해야 함을 보였다.

향후 연구로 가속기 하드웨어 프로파일링을 세분화해서 분석하고, 이를 바탕으로 DLA를 사용한 추론 성능 최적화 방법과 모델 구조를 연구할 계획이다. 또한, CPU, GPU, DLA 간 딥 러닝 연산 파이프라이닝 최적화를 위한 방법도 연구한다.

REFERENCES

[1] K. Seshadri, B. Akin, J. Laudon, R. Narayanaswami, and A. Yazdanbakhsh, “An Evaluation of Edge TPU Accelerators for Convolutional Neural Networks,” *arXiv:2102.10423*, pp. 1-13, October 2022.

[2] S.P. Kaarmukilan, A. Hazarika, A. Thomas K., S. Poddar, and H. Rahaman, “An Accelerated Prototype with Movidius Neural Compute Stick for Real-Time Object Detection,” in *Proc. 2020 International Symposium on Devices, Circuits and Systems (ISDCS)*, pp. 1-5, Virtual, October 2020.

[3] E.J. Jeong, J.R. Kim, S. Tan, J.S. Lee, and S.H. Ha, “Deep Learning Inference Parallelization on Heterogeneous Processors With TensorRT,” *IEEE Embedded Systems Letters*, vol. 14, no. 1, pp. 15-18, March 2022.

[4] L. Song, F. Chen, Y. Zhuo, X. Qian, H. Li, and Y. Chen, “AccPar: Tensor Partitioning for Heterogeneous Deep Learning Accelerators,” in *Proc. 2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 342-355, San Diego, CA, United States, February 2020.

[5] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” in *Proc. IEEE conference on computer vision and pattern recognition (CVPR) 2018*, pp. 4510-4520, Salt Lake City, Utah, United States, June 2018.

[6] K. He, X. Zhang, S. Ren, and J. Sun, “Identity Mappings in Deep Residual Networks,” in *Proc. 14th European Conference on Computer Vision (ECCV)*, pp. 630-645, Amsterdam, The Netherlands, October 2016.