Zero-copy 방식을 활용한 임베디드 환경에서의 딥러닝 성능 최적화*

이민학, 강우철
인천대학교 임베디드시스템공학과
e-mail:mnaklee@gmail.com, wchkang@inu.ac.kr

The optimization of deep learning performance for embedded systems using a zero-copy technique

Minhak Lee, Woochul Kang
Dept of Embedded System Engineering, Incheon National University

요 약

딥러닝의 대표적 개발 환경 중 하나인 Caffe를 임베디드 시스템의 메모리 구조를 고려하여 최적화하고 실제 측정 실험으로 기존의 방식보다 처리시간과 소비 전력량의 이득이 있다는 것을 확인하였다. 구체적으로 통합 메모리를 사용하는 임베디드 시스템 환경의 특성에 적합한 zero-copy기법을 적용하여 CPU와 GPU 모두 접근이 가능하도록 메모리 영역을 맵핑하는 방식으로 메모리 복제에 따른 오버헤드를 줄였으며, GoogLeNet 네트워크 모델에 대하여 10%의 처리 속도 향상과, 36% 소비 전력 감소를 확인하였다.

1. 서론

최근 딥러닝은 이미지 식별과 같은 영상처리 산업을 넘어 자연어 처리, 통계 분석 등 다양한 분야에서 활용되고 있으며, Caffe, TensorFlow와 같이 딥러닝을 사용하기 위한 다양한 들이 개발 되고 있다[1]. 딥러닝 기술의 발전 배경으로는 GPU와 General-Purpose computing on Graphics Processing Units(GPGPU) 기술의 발전으로 딥러닝의 방대한 양의 연산을 GPU를 이용하여 병렬로 처리하는 것이 가능해졌기 때문이다. 최근에는 임베디드 산업에서 GPGPU를 활용 가능한 임베디드 하드웨어가 등장함에 따라 임베디드 환경에서도 GPU를 이용한 딥러닝 기술을 사용할 수 있게 되었고, 임베디드 산업에서 딥러닝은 자동차의 자율주행, IoT등 다양한 분야에 활용되고 있다.

그러나 기존 딥러닝을 위해 만들어진 개발 환경은 일반적인 PC 서버 환경에서 적합하게 개발되었기 때문에 일반적인 PC와는 상이한 아키텍처를 가지고 있는 임베디드 시스템 환경에서는 별도의 최적화가 필요한 것이 현실이다. 특히 제한된 전력을 사용해야하는 임베디드 시스템의 특성상 불필요한 작업으로 인한 전력소비를 줄이는 최적화 작업이 필수적이라고 할 수 있다.

본 논문에서는 CPU에서 사용하는 호스트 메모리와 GPU에서 사용하는 디바이스 메모리를 공유하는 통합 메모리 구조를 가진 임베디드 시스템 환경에서 Caffe 개발 환경을 통합 메모리 구조에 적합한 zero-copy 기법을 적용하여 최적화하고, GoogLeNet 네트워크로[2] 학습한 모델을 사용한 이미지 식별의 처리 속도와, 전력 소비량을 측정하여 개선 정도를 비교한다.

2. 본론

본 논문에서 측정에 사용한 Caffe 개발 환경은 대부분의 딥러닝 개발 환경과 같이 내부적으로 NVIDIA CUDA 라이브러리를 사용하여 많은 양의 신경망을 GPU상에서 병렬처리 하는 방식이다. 일반적인 PC 환경에서의 GPU 병렬처리는 기본적으로 CPU에서 접근 가능한 호스트 메모리에 저장된 데이터를 PCI 슬롯에 장착된 GPU의 디바이스 메모리로 복제하여 병렬처리 후 결과를 다시 호스트의 메모리로 복제하는 방식을 사용한다. 그러나 대부분의 임베디드 환경에서는 CPU와 GPU에서 모두 접근이 가능한 통합 메모리 구조를 사용하기 때문에 호스트와 디바이스 사이에 메모리를 복제하여 사용하는 것은 불필요한 작업이다.

기존 Caffe 개발 환경에서 사용하는 방식은 디바이스 메모리 영역에 CUDA 라이브러리의 cudaMallocHost함수를 사용하여 메모리 영역을 생성하고 호스트와 디바이스 메모리 사이에서 cudaMemCpy 함수로 데이터를 복제하는 방식을 사용하고 있다. 본 논문에서 사용한 방식은 기존 방

*이 논문은 2016년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (NRF-2014R1A1A1005781)

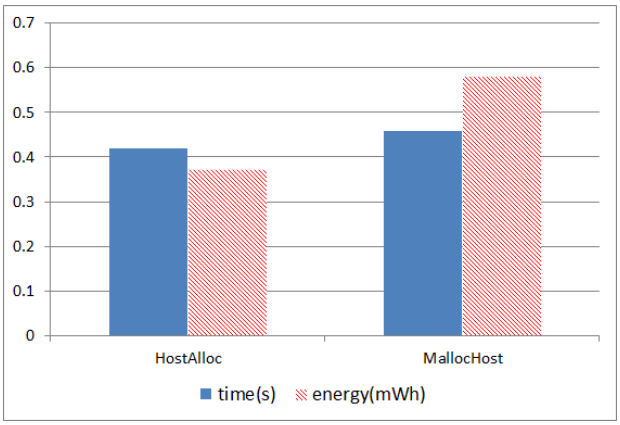
식을 임베디드 환경의 통합 메모리방식에 적합하게 수정한 zero-copy 방식으로 호스트와 디바이스 사이에 복제가 아닌 맵핑 방식을 사용한다. 구체적으로 cudaHostAlloc 함수를 사용하여 맵핑 가능하도록 데이터 영역을 생성하고, cudaHostGetDevicePointer 함수로 CPU와 GPU 모두 사용 가능한 데이터 영역으로 설정하여 통합된 메모리 영역에 CPU와 GPU 모두 접근이 가능한 zero-copy방식을 적용하였다.

성능의 측정은 사전에 GoogLeNet 네트워크와 Imagenet의 90026장의 학습 이미지, 3550장의 검증 이미지로 학습된 모델을 사용하였고, 224×224pixel 크기의 동일한 이미지 식별을 수행하는데 걸린 수행 시간과, 소비 전력량을 측정하였다.

<표 1> 테스트 환경

Jetson TK1	
CPU	NVIDIA 4 Cores ARM Cortex-A15 CPU
GPU	NVIDIA Kepler 192 CUDA Cores
Memory	2GB
전력 소비량 측정	
YOKOGAWA WT310E	

실험에서는 표 1에서 기술한 것과 같이 임베디드 보드로는 NVIDIA Jetson TK1 보드를 사용하였다. CPU는 2.32GHz로 설정된 4개의 코어, GPU는 804MHz로 클락 속도를 고정하였다. 세부적인 성능 측정은 사전 학습된 모델이 메모리에 읽어져 있는 상태에서 하나의 이미지를 판별하는데 걸린 시간을 기존의 Caffe 개발 환경을 사용한 MallocHost 방식과 zero-copy 기법을 적용하여 수정된 HostAlloc 방식을 각각 100회 수행한 평균을 비교하였다. 전력 소비량은 0.001초마다 한 번씩 전력량을 측정 가능한 YOKOGAWA WT310E 파워미터를 사용하여 하나의 이미지를 판별하는데 소비되는 에너지를 100회 측정한 평균으로 나타냈다.



(그림 1) 이미지 식별 처리 시간 및 에너지 소비량

그림 1의 처리 시간에서 나타나듯 cudaHostAlloc함수를 사용한 zero-copy 방식이 기존 cudaMallocHost함수를 사용한 방식과 비교했을 때 0.41초와 0.46초로 대략 10%의 처리 속도 향상을 확인 할 수 있었다.

에너지 소비량은 그래프에서 나타나듯 전력 효율이 확연히 개선되는 것을 확인할 수 있다. zero-copy 방식으로 최적화 적용 후 하나의 이미지를 처리하는 동안 평균적으로 0.37mWh의 에너지를 소비하였으며 이는 기존방식의 측정 결과인 0.58mWh와 비교하면 36% 감소하였다.

성능 향상의 원인으로는 CPU와 GPU 사이에 메모리 복제로 데이터를 전달하여 GPU에서 병렬로 처리하는 기존의 방식은 데이터를 복제하는데 걸리는 전송 시간으로 인해 대기 시간이 발생하였고, 연산을 수행하는 GPU 역시 수많은 코어들 중 많은 부분이 연산을 처리하지 않고 대기상태인 것을 확인하였다. 그러나 zero-copy 기법으로 최적화한 실험에서는 CPU와 GPU 사이 메모리 복제에 따른 대기시간을 최소화하기 때문에 GPU의 거의 모든 코어들이 대기 상태 없이 동작하는 것을 확인하였다.

3. 결론

GPGPU를 지원하는 다양한 임베디드 보드가 보급됨에 따라 산업에서도 임베디드 환경에서 GPGPU를 활용한 다양한 연구와 기술 개발이 진행되고 있으며 특히 딥러닝을 이용한 자율주행 차량 연구와 같은 분야에서 사용되고 있다. 본 논문은 임베디드 보드인 Jetson TK1 보드에 Caffe 개발 환경의 메모리 사용을 최적화하고 기존 대비 처리 속도와 전력 효율 상승을 실제 측정을 통해 확인하였다. 그 결과 최적화 전과 비교하면 처리 속도는 10% 향상되었고, 전력 효율은 36% 개선되는 것을 확인하였다. 향후 하나의 네트워크 모델이 아닌 다양한 모델을 대상으로 실험하고 추가적인 최적화 방안에 대하여 연구를 진행 할 예정이다.

참고문헌

[1] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshik, Sergio Guadarrama and Trevor Darrell, *Caffe: Convolutional Architecture for Fast Feature Embedding* http://caffe.berkeleyvision.org/, 2013

[2] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions," arXiv preprint arXiv:1409.4842, 2014.