모바일 GPU 에서의 딥러닝 모델 파티셔닝 (Partitioning) 성능 평가에 관한 연구

강민주, 고정길
연세대학교 IT 융합공학과 지능형반도체 IT 융합전공

minju.kang@yonsei.ac.kr, jeonggil.ko@yonsei.ac.kr

A Study on the Evaluation of Deep Learning Model Partitioning Performance on Mobile GPUs

Min Ju Kang, JeongGil Ko
School of Integrated Technology, Yonsei University

BK21 Graduate Program in Intelligent Semiconductor Technology

요 약

최근 모바일에서의 딥러닝 어플리케이션의 수요가 증가함에 따라, 적은 모바일 컴퓨팅 자원을 활용한, 효율적인 딥러닝 모델 추론을 위한 방법에 대한 연구가 많이 진행되고 있다. 본 논문에서는 모바일 GPU 에서의 model partitioning 의 필요성에 대해 논하고, 이를 활용한 성능 향상 방안에 대한 연구를 담고 있다.

Ⅰ. 서 론

최근 다양한 모바일 응용에서의 딥러닝 어플리케이션의 수요가 증가함에 따라, 모바일 기기 환경의 제한적인 자원으로 딥러닝 모델이 최대성능을 낼 수 있도록 하는 모델 스케줄링 연구가 많이 진행되고 있다 [3,4]. 특히, 모바일 GPU 는 모바일 프로세서 중 범용적으로 높은 성능을 보이기 때문에, 모바일 GPU 에서의 모델 스케줄링 연구는 더욱 중요해지고 있다. 서버 GPU 와는 달리, 모바일 GPU 는 메모리나 컴퓨팅 측면에서 제한이 있기 때문에 기존의 모델 추론 방식으로는 최대의 성능에 도달하기 힘들다. 또한 모바일 GPU 의 경우 서버의 GPU 와는 달리 모바일 기기의 화면 컨텐츠의 디스플레이를 위한 렌더링 작업의 수행으로 딥러닝 모델의 처리를 위한 유휴시간 확보의 어려움이 있다. 따라서, 본 논문은 제한된 모바일 GPU 자원을 최대한 활용하기 위한 model partitioning 의 필요성에 대해 논하고, 다양한 model partition 기준에 따른 추론 결과를 고찰하고자 한다.

Ⅱ. 모델 파티셔닝 (Model Partitioning)

Ⅱ-1 Partition 적용 유무에 따른 추론 속도 차이

기존의 모바일 딥러닝 프레임워크 (예: TensorFlow Lite) 작동 방식을 살펴보면, 해당 프로세서가 지원하는 연산 가능 환경 내에서, 모델이 해당 프로세서 내에 한 번에 동작한다. 하지만, 기존 방식으로는 모바일 자원이 효율적으로 사용되지 않으며[1], 모바일 GPU 는 서버 GPU 와는 다른 메모리 구조를 가지고 있기 때문에[2], 모바일 GPU 의 자원을 최대한으로 활용하기에는 기존의 딥러닝 프레임워크의 추론 방식은 적합하지 않다. Table 1은 VGG16.tflite 의 model partitioning 여부에 따른 모바일 GPU 추론 속도 비교 실험 결과이다. 해당 실험을 통해, 모델을 partition 하는 것이 partition 하지 않은 상황에 비해 약 1.8배의 추론 속도 향상이 있는 것을 확인하였다.

	w/ Partitioning	w/o Partitioning
Inference Time on GPU	52 ms	97 ms

Table 1: Model partitioning 여부에 따른 모바일 GPU 추론 속도 비교.

Ⅱ-2 Partition point 설정에 따른 추론 속도 차이

추가적으로, partition point 에 따른 성능 차이를 확인하기 위한 실험을 진행하였다. 실험에 활용한 모바일 기기는 Exynos 2200 SoC 를 탑재한 실험용 모바일 기기를 활용하였으며, 딥러닝 모델은 모바일 어플리케이션에서 주로 사용하는 VGG16.tflite 를 활용하였다. 해당 모델은 다양한 인공지능 응용에서 활용되는 모델로 크기는 553.4MB 이며 13 개의 convolution layer 와 3 개의 fully connected layer 를 포함한다. 해당 실험 환경은 Table 1 을 포함하여 본 연구의 모든 실험에 공통적으로 적용되었다. 해당 실험들은 100 회의 실험을 수행한 후 평균값을 나타낸다.

	Layer by layer partitioning
Inference Time on GPU	286ms

Table 2: Layer 마다 partitioning 했을 때의 모바일 GPU 추론 속도.

Table 2 는 VGG16.tflite 모델을 각 layer 단위로 partition 했을 때의 모바일 GPU 에서의 실험 결과이다. 단순히 layer 단위로 partition 한 모델의 추론시간은 partition 하지 않은 기본적인 접근에 비해 추론 시간이 길어지는 것을 확인하였다 (Table 1 결과 참조). 따라서 본 실험의 결과를 기반으로 판단했을 때 단순히 입력 모델의 각 layer 단위의 partitioning 을 통한 추론 보다는 효

과적인 partitioning 방식이 필요함을 해당 실험을 통해 검증할 수 있었다.

	K=2	K=3	K=4
Layer Length	66ms	69ms	80ms
Layer Type	**52ms**	X	X
Parameter Count	101ms	65ms	63ms

Table 3: model partition point 에 따른 모바일 GPU 추론 속도 비교.

Table 3 는 model partitioning 을 layer length, layer type, parameter 개수 등의 다양한 기준에 따라 진행한 후 모바일 GPU 추론 속도를 비교한 결과이다. 총 layer 개수, layer 의 neural network type, 그리고 parameter 를 partition point 기준으로 실험을 진행하였고, partition 개수 (K = 2, 3, 4)를 늘려가며 실험을 진행하였다.

구체적으로, "layer length" 실험에서는 VGG16.tflite 모델의 총 layer 개수를 일정한 간격으로 분리하였으며, "parameter count" 실험은 균등하게 파라미터 개수를 기준으로 모델을 partition 하였다. "Layer type"의 경우 VGG16.tflite 모델내에서 neural network 의 형식(type)이 달라지는 점을 기준으로 정했으며, 본 연구에서는 convolution layer 그룹과 fully connected layer 그룹으로 나누었다. 해당 실험은 2 개의 그룹만 분리되어 K=3. K=4 에 대한 실험 결과는 별도로 보고하지 않았다.

실험 결과, neural network type 별 2 개로 partition 하였을 때, 추론 속도가 가장 크게 향상되었다. 또한, layer length 와 달리, parameter 기준으로 partition 의 개수를 늘렸을 때, 추론 속도가 향상되었다. 따라서, model partitioning 관련 기준에 따라, 추론 속도의 향상 정도가 달라지는 것을 확인할 수 있었다.

Ⅲ. 향후 연구 방향

본 연구를 통해 모델 partitioning 을 통해 모바일 GPU 에서의 추론 속도 향상이 가능함을 확인하였다. 동시에 model partitioning 의 기법에 따라 추론 시간 성능이 큰 폭으로 달라짐을 확인하였다. 따라서 본 연구를 확장하여 향후 최적의 모델 partitioning 을 위한 알고리즘을 제안하고자 한다. 또한 다양한 baseline 딥러닝 모델을 통해 해당 partitioning 기법의 효용성을 검증할 예정이다.

Ⅳ. 결론

본 논문에서는, 모바일 GPU 에서, 딥러닝 어플리케이션의 성능 향상을 위한, model partitioning 의 필요성과 모바일 GPU 구조를 고려한 model partition point 를 실험을 통해 분석하였다.

전체 모델을 한 번에 추론하기보다는, model partitioning 을 적용하는 것이 추론 속도에 효과적이었으며, layer 의 neural network type 을 고려하여 partition 한 것이 가장 효과적이었음을 확인하였다.

ACKNOWLEDGMENT

이 논문은 2024 년도 정부(산업통상자원부)의 재원으로 한국산업기술진흥원의 지원을 받아 수행된 연구이며 (P0020535, 2024 년 산업혁신인재성장지원사업), 2023 년도 두뇌한국 21 사업(4 단계 BK21 사업)에 의하여 지원되었음.

참 고 문 헌

[1] Jingyu Lee, Yunxin Liu, and Youngki Lee. 2021. ParallelFusion: towards maximum utilization of mobile GPU for DNN inference. In Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning

[2] Rendong Liang, Ting Cao, Jicheng Wen, Manni Wang, Yang Wang, Jianhua Zou, and Yunxin Liu. 2022. Romou: Rapidly Generate High-Performance Tensor Kernels for Mobile GPUs. In Proceedings of the 24th Annual International Conference on Mobile Computing and Networking.

[3] Joo Seong Jeong, Jingyu Lee, Donghyun Kim, Changmin Jeon, Changjin Jeong, Youngki Lee, and Byung-Gon Chun. 2022. Band: coordinated multi-dnn inference on heterogeneous mobile processors. In Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services.

[4] Fucheng Jia, Deyu Zhang, Ting Cao, Shiqi Jiang, Yunxin Liu, Ju Ren, and Yaoxue Zhang. 2022. Codl: efficient cpu-gpu co-execution for deep learning inference on mobile devices. In Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services.