ISSN 2383-630X(Print) / ISSN 2383-6296(Online)
Journal of KIISE, Vol. 49, No. 3, pp. 256-260, 2022. 3
https://doi.org/10.5626/JOK.2022.49.3.256

엣지 컴퓨팅 환경에서의 딥 러닝 연산 오프로딩의 병렬 최적화

(Parallel Optimization of Deep Learning Computation Offloading in Edge Computing Environment)

신 광 용 †　　문 수 묵 ††
(Kwang Yong Shin)　(Soo-Mook Moon)

요 약 컴퓨팅 자원이 부족한 디바이스에 연산량이 많은 딥 러닝 애플리케이션을 실행하기 위해 주변에 있는 서버에 연산을 오프로딩하는 엣지 컴퓨팅 기술이 제안되었다. 그러나 딥 러닝 연산을 오프로딩하기 위해서는 서버에 모델을 먼저 업로드해야 하는 단점이 있다. 이를 해결하기 위해 모델을 점진적으로 전송하는 동시에 서버가 클라이언트 연산을 대신 수행하는 점진적 오프로딩 시스템이 제안되었다[1]. 점진적 오프로딩 시스템은 오프로딩에 걸리는 시간을 크게 단축했으나, 모델 구축 시간을 고려하지 않아서 전체 모델 업로드 시간이 늘어나는 단점이 있었다. 본 논문은 모델 구축과 모델 업로드의 병렬 최적화를 통해 기존 시스템의 문제점을 해결해서 기존 시스템 대비 전체 모델 업로드 시간을 최대 30% 개선했다.

키워드: 엣지 컴퓨팅, 연산 오프로딩, 딥 러닝, 모바일 컴퓨팅

Abstract Computation offloading to edge servers has been proposed as a solution to performing computation-intensive deep learning applications on devices with low hardware capabilities. However, the deep learning model has to be uploaded to the edge server before computation offloading is possible, a non-trivial assumption in the edge server environment. Incremental offloading of neural networks was proposed as a solution as it can simultaneously upload model and offload computation [1]. Although it reduced the model upload time required for computation offloading, it did not properly handle the model creation overhead, increasing the time required to upload the entire model. This work solves this problem by parallel optimization of model uploading and creation, decreasing the model upload time by up to 30% compared to the previous system.

Keywords: edge computing, computation offloading, deep learning, mobile computing

· 이 논문은 정부(과학기술정보통신부)의 재원으로 한국 연구재단의 지원을 받아 수행된 연구임(No. 2020R1A2B5B02001845)
· 이 논문은 2020 한국컴퓨터종합학술대회에서 '엣지 컴퓨팅 환경에서의 딥 러닝 연산 오프로딩의 병렬 최적화'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 서울대학교 전기정보공학부 학생(Seoul Nat'l Univ.)
danielshin32@snu.ac.kr
(Corresponding author임)
†† 종신회원 : 서울대학교 전기정보공학부 교수
smoon@snu.ac.kr

논문접수 : 2020년 9월 18일
(Received 18 September 2020)
논문수정 : 2022년 1월 4일
(Revised 4 January 2022)
심사완료 : 2022년 2월 24일
(Accepted 24 February 2022)

정보과학회논문지 제49권 제3호(2022. 3)

1. 서 론

딥 러닝을 활용하면 사진 인식이나 음성 인식 등 기존에 처리하기 어려웠던 작업을 쉽고 정확하게 수행할 수 있다. 하지만 딥 러닝은 높은 연산량 때문에 성능이 떨어지는 하드웨어 장치에 활용하기 어렵다. 예를 들어 AlexNet 모델의 경우, 사진을 한 번 인식하기 위해서는 약 7억 번의 소수점 연산이 필요하다. 스마트 글라스에서 딥 러닝을 활용한 사진 인식 및 정보 처리가 느리면 사용자 경험을 해칠 수 있고, 자율주행 자동차의 이미지 처리가 느리다면 생명에 위협을 가할 수도 있다.

연산 오프로딩은 전용 서버가 하드웨어 성능이 떨어지는 장치 대신 딥 러닝 연산을 수행하는 방법이다. 연산 오프로딩은 딥 러닝 연산을 컴퓨팅 자원이 충분한 서버가 대신 수행하기 때문에 하드웨어 성능이 떨어지는 장치에서 실행하는 것 보다 더 빠르다. 그러나 다수 사용자들의 요청이 한 중앙 서버에 몰리면 서버 및 통신 망에 과부하가 걸려서 응답 속도 및 성능이 느려질 수 있다. 이런 한계점을 극복하기 위해 한 개의 중앙 서버가 아닌, 여러 개의 서버가 지리적으로 분산된 엣지(edge) 컴퓨팅 환경이 제안되었다[2].

엣지 컴퓨팅 환경은 연산 오프로딩하는 서버가 클라이언트와 지리적으로 가까워서 통신 지연이 적고, 그만큼 전체 망의 통신 부담도 낮아서 더 큰 확장성을 가진다. 그러나 엣지 컴퓨팅 환경은 중앙 서버와 달리 다수의 서버가 분산되어 있어서 유저가 움직이면 지리적으로 가장 가까운 엣지 서버가 자주 바뀔 수 있다. 그래서 긍정적인 사용자 경험 위해서는 다수의 엣지 서버와의 부드러운 연결 및 연결 해제를 지원해야 한다.

새로 연결한 엣지 서버에 원하는 딥 러닝 연산을 오프로딩하기 위해서는 서버에 해당 모델 정보가 있어야 한다. 서버에 모델 정보가 없으면 모델을 서버에 업로드하는 과정을 거쳐야 오프로딩 가능하다. 갑작스럽게 연산 오프로딩을 진행하던 엣지 서버와 연결 끊고 새로운 엣지 서버와 연결하면, 새로운 엣지 서버에 모델 업로드하는 과정 때문에 연산 오프로딩이 끊기는, 사용자 경험을 해치는 현상이 발생할 수 있다.

이를 개선하고자 모델을 분할하고 업로드함으로써 오프로딩 가능한 모델 연산량을 점진적으로 늘릴 수 있는 점진적 오프로딩 시스템이 제안되었다[1]. 점진적 오프로딩 시스템에는 연결된 엣지 서버가 없으면 딥 러닝 연산을 클라이언트에서 수행하다가, 엣지 서버와 연결하면 모델 일부를 먼저 전송하여 해당 연산을 먼저 오프로딩하고, 유휴 시간에 나머지를 전송함으로써 모델 전송과 연산 오프로딩을 동시에 지원할 수 있다.

그러나 기존 점진적 오프로딩 시스템은 전체 모델을 업로드하는 시간이 지연되는 문제점이 있다. 구체적으로, ResNet-152 모델을 업로드하면, 24초에 업로드가 완료될 것으로 예상했음에도 실제로는 32초나 걸렸다. 기존 점진적 오프로딩 시스템은 서버가 모델을 받고 구축까지 하고 다음에 모델 업로드를 받는데, 모델을 구축하는 동안 다음 모델 업로드를 진행 안해서 발생한 문제였다.

본 논문은 모델 업로드와 연산 오프로딩과 더불어 모델 구축도 다른 스레드(thread)에 할당하는 병렬 최적화를 통해 성능을 개선했다. 실험 결과 기존 시스템 대비 전체 모델 업로드 시간을 최대 30%나 단축할 수 있다.

2. 기존 점진적 오프로딩 시스템과 문제점

점진적 오프로딩 시스템은 딥 러닝 모델이 레이어(layer)로 구성되어 있는 점을 활용한다. 인접한 레이어들을 묶어서 모델을 여러 개의 작은 모델로 분할한다. 그림 1은 4개의 레이어로 이루어진 딥 러닝 모델을 2개 레이어로 이루어진 2개의 딥 러닝 모델로 분할한 예시이다. 분할된 모델의 입출력을 연결해서 원래 모델의 연산과 동일한 연산을 수행할 수 있다.

이를 활용해 전체 모델 업로드가 완료되기 전에도 일부 연산을 오프로딩한다. 전체 모델을 분할하고 분할된 모델들을 순서대로 업로드 및 서버에 구축하면 구축된 모델에 해당하는 연산을 오프로딩할 수 있다. 업로드 및 오프로딩 작업을 동시에 실행하기 위해 각각 업로드 및 오프로딩 스레드에 할당해서 병렬적으로 처리한다. 업로

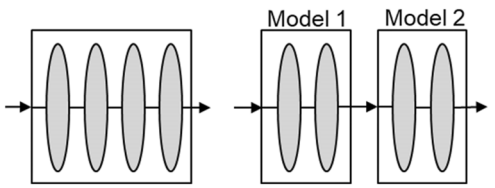

그림 1 레이어 4개의 딥 러닝 모델(좌)을 각각 레이어 2개의 모델 1, 2(우)로 분할

Fig. 1 Partitioning a deep learning model with 4 layers (left) into 2 2-layer models 1 and 2 (right)

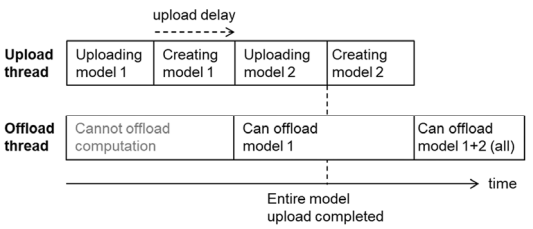

그림 2 기존 점진적 오프로딩 시스템

Fig. 2 Previous incremental offloading system

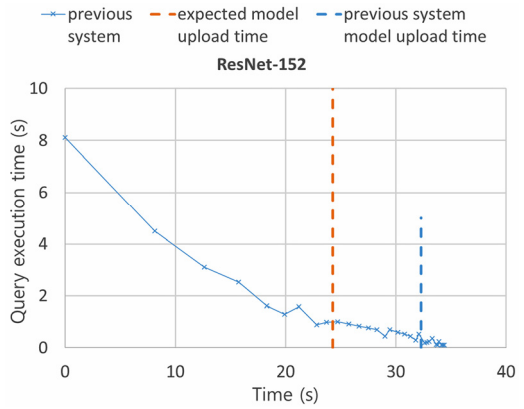

그림 3 기존 시스템에서의 연산 실행 시간 그래프
Fig. 3 Query execution time of the previous system

드 스레드는 모델 업로드 및 구축을 담당하고, 오프로딩 스레드는 가능한 만큼의 연산을 오프로딩한다. 기존 점진적 오프로딩 시스템은 그림 2와 같이 모델 1의 연산을 전체 모델 업로드가 완료되기 전에도 오프로딩할 수 있다.

해당 시스템에 ResNet-152 모델을 오프로딩하면 그림 3과 같은 그래프를 얻을 수 있다. 이 그래프는 0초에 클라이언트가 엣지 서버에 연결하고 점진적 오프로딩을 실행했을 때의 모델 실행 시점과 시간의 그래프다. 0초에 서버에 모델 업로드를 시작하는 동시에 첫 딥 러닝 연산 요청이 발생하고, 해당 연산이 완료되면 바로 다음 딥 러닝 연산이 발생한다. 첫 딥 러닝 연산은 8초가 소요됐다. 8초의 연산 동안 분할된 모델들의 업로드 및 구축이 완료됐기 때문에, 이후 발생하는 모델 연산 요청은 더 많은 연산을 오프로딩할 수 있어 더 빠르다. 그래서 8초 시점에 발생한 연산 요청은 0초 시점 요청보다 처리 속도가 빠르다. 시간이 지날수록 더 많은 연산을 오프로딩하기 때문에 실행 시간이 점진적으로 개선된다.

이 실험은 통신 속도를 80Mbps로 제한하고 실험했다. 이 통신 속도에 243MB 크기의 ResNet-152 모델을 업로드하면 소요시간은 24초로 예상된다. 그러나 그림 3에서 볼 수 있듯이 기존 점진적 오프로딩 시스템은 업로드가 32초 소요되는 지연 현상을 확인할 수 있었다. 연산 오프로딩에 필요한 모델 입출력 정보를 통신하느라 생기는 지연으로 보기에는 지나치게 긴 시간이다.

이는 모델을 구축하는 동안 모델 업로드를 진행 하지 못해 생기는 문제점이었다. 그림 2를 보면, 모델1 업로드가 완료되면 해당 모델을 서버에 구축하는 단계가 있다. 그러나 모델 구축을 모델 업로드도 담당하는 스레드가 처리하기 때문에 구축하는 동안 모델2 업로드가 진행되지 않는 것이다. 기존 점진적 오프로딩 시스템은 모델 구축도 중요한 요소임을 간과한 것이다.

3. 병렬적 모델 구축

모델을 구축하는 동안 모델 업로드의 진행이 중단되는 것이 문제이므로, 모델 업로드를 모델 구축과 동시에 실행하게 하는 병렬적 처리를 통해 해결할 수 있다. 기존 시스템은 업로드와 오프로딩 스레드를 구성해서 병렬 처리한 만큼, 제안 시스템은 여기에 구축 스레드를 추가한다.

그림 4에서 제안 시스템의 구축, 업로드 및 오프로딩 스레드의 역할을 볼 수 있다. 그림 2와 가장 큰 차이점은 모델 1 업로드를 완료하면 업로드 스레드가 아닌 구축 스레드가 모델 1을 구축한다는 점이다. 그래서 제안 시스템은 모델 1을 구축하는 동안에도 모델 2 업로드를 동시에 진행해서 전체 모델 업로드 완료 시점을 지연시키지 않는다. 모델 2 구축도 더 일찍 완료해서 전체 모델의 연산 오프로딩도 더 일찍 시작할 수 있다. 편의상 2개로 분할했을 때의 경우를 예시로 들었지만, 모델을 더 많이 분할하고 업로드할 때도 동일하게 적용할 수 있다.

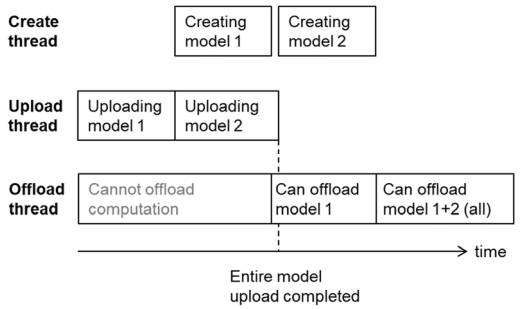

그림 4 모델 구축 스레드를 포함한 제안 시스템
Fig. 4 Proposed system with the create thread

전체 모델을 많은 개수의 작은 모델로 분할해서 업로드하면 한 개의 모델을 구축하는 동안 다수 모델 업로드가 완료될 수 있다. 그러면 업로드 완료된 다수의 모델 정보를 통합해서 한 번의 모델 구축으로 처리하도록 구현했다. 구축 스레드가 모델 구축을 완료했는데도 다음 모델 업로드가 진행 중이면, 구축 스레드는 업로드 스레드로부터 업로드 완료했다는 신호를 기다린 다음에 모델 구축을 시작한다.

4. 실험 결과

실험은 기존 시스템과 같은 환경에서 실험했다. 클라이언트는 Odroid XU4 보드를 사용했고 서버는 4코어 3.6GHz x86 CPU와 Titan XP GPU를 사용했다. WiFi 통신을 재현하기 위해 통신 속도를 80Mbps로 제한하고 실험했다. 모델 크기가 크고 152개 레이어로 구성된 ResNet-152, 모델 크기는 크지만 레이어 개수가 적은

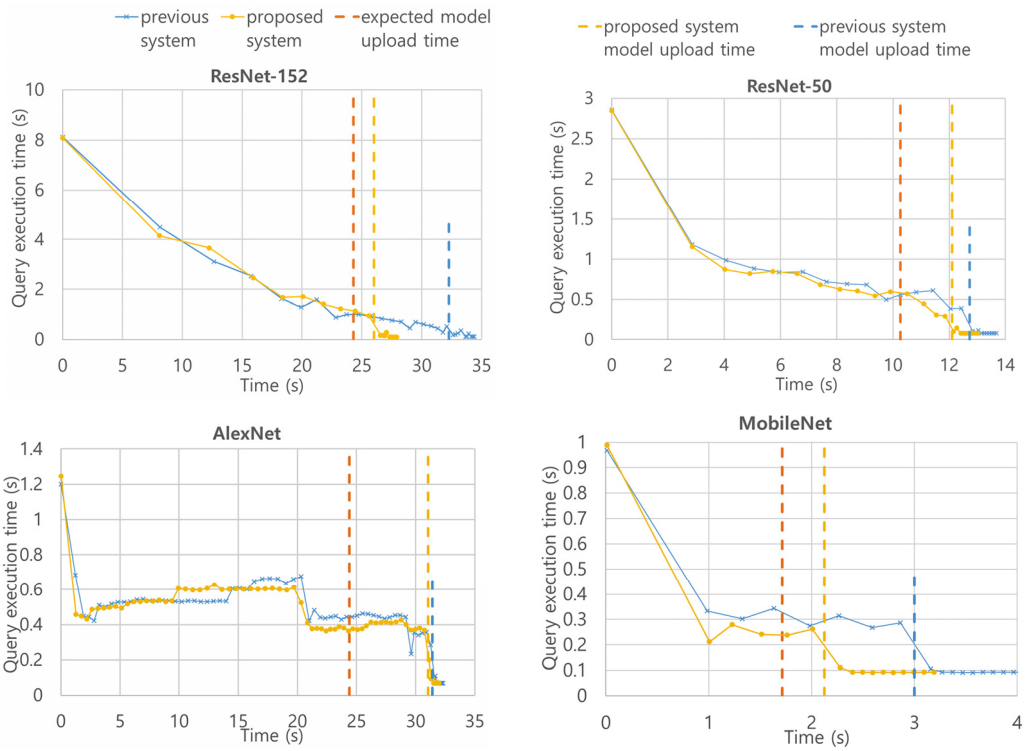

그림 5 기존 시스템과 제안 시스템에서의 실행 시간 그래프

Fig. 5 Query execution graph of the previous and proposed systems.

AlexNet, 중간 크기의 ResNet-50, 그리고 모델 크기가 작지만 레이어 개수가 많은 MobileNet, 총 4개의 모델로 실험했다. 모델을 분할하는 알고리즘은 모델 실행 시간을 빠르게 감소시킬 수 있는, 가능한 많은 개수의 모델로 분할하는 알고리즘을 사용했다[3]. 전체 모델 업로드 완료하면 추가로 10개의 모델 연산을 요청하고 실험을 종료했다. 기존 시스템과 제안한 시스템으로 각각 두 번씩 실험했고 시스템 및 모델이 동일하면 비슷한 양상이 나온다는 것을 확인했다. 딥 러닝 프레임워크로는 Caffe[4]를 사용했다.

그림 5는 기존 시스템과 제안 시스템에 점진적 오프로딩을 진행했을 때의 실행 시간 그래프다. 표 1은 각 모델의 크기, 모델 분할 개수 그리고 제안 시스템의 모델 업로드 시간 개선을 보여준다. 그림 5에 볼 수 있듯이, 제안 시스템은 기존 시스템의 점진적으로 빨라지는 모델 실행 속도를 유지하면서도 전체 모델 업로드를 더 일찍 완료한다. ResNet-152의 경우, 제안 시스템은 전체 모델 업로드 완료 시점을 32.3초에서 26.0초로 6.3초, 기존 시스템 대비 19.3% 개선한 것이다. 이는 모델 크기와 통신 속도를 고려하면 예상되는 24.3초에 근접함을 볼 수 있다. 제안 시스템에 ResNet-152에 해당하는 모델들의 구축 시간이 총 5.2초로 측정된 것을 확인할 수 있는데, 이는 기존 시스템의 전체 모델 업로드 시간 지연은 대부분 구축 시간 때문이라는 것을 뒷받침한다.

표 1을 보면, ResNet-152와 MobileNet 모델의 개선점이 가장 두드러진다는 것을 확인할 수 있다. MobileNet

표 1 각 모델의 크기, 분할 개수 및 업로드 시간 개선 비교

Table. 1 Comparison of size, number of partitions, and upload time speedup for each model

Model	Model Size (MB)	Number of partitions	Upload time speedup (s)	Upload time speedup
ResNet-152	243	41	6.3	19.3%
ResNet-50	103	11	0.6	4.7%
MobileNet	17	17	0.9	30.0%
AlexNet	244	7	0.3	1.1%

의 경우, 모델 업로드 완료 시점이 3초에서 2.1초로 0.9초 줄었는데, 이는 30% 개선된 것이다. 또한, 모델을 더 많이 분할할수록 제안한 병렬 최적화가 중요하다는 것을 볼 수 있다. ResNet-152은 모델을 41개로 가장 많이 분할했고, MobileNet 또한 모델 크기 대비 모델 분할 개수가 많았기 때문이다. AlexNet과 ResNet-50의 경우, 모델 크기 대비 모델 분할 개수가 적기 때문에 개선이 적었다.

5. 결 론

엣지 컴퓨팅 환경에는 높은 성능의 딥 러닝 연산 오프로딩이 가능하지만, 오프로딩에 필요한 모델 정보를 유저 지리적 위치에 가까운 엣지 서버에 업로드해야 되는 문제점이 있다. 엣지 서버 환경에 적합한 점진적 오프로딩 시스템은 서버의 모델 실행 시간뿐만 아니라 모델 구축 시간도 고려해야 함을 보여준다. 본 논문의 연구 결과를 바탕으로 향후 점진적 모델 오프로딩 시스템에서의 딥 러닝 모델을 효율적으로 분할하는 알고리즘을 연구할 수 있을 것으로 기대한다.

References

[1] H. J. Jeong, H. J. Lee, C. H. Shin, and S. M. Moon, "IONN: Incremental offloading of neural network computations from mobile devices to edge servers," *Proc. of the ACM Symposium on Cloud Computing*, pp. 401-411, Oct. 2018.

[2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," *Proc. of the first edition of the MCC workshop on Mobile cloud computing*, pp. 13-16, Aug. 2012.

[3] Shin, K. Y., Jeong, H. J., and Moon, S., M,, "Enhanced Partitioning of DNN Layers for Uploading from Mobile Devices to Edge Servers," *The 3rd International Workshop on Deep Learning for Mobile Systems and Applications*, pp. 35-40, Jun. 2019.

[4] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., ... and Darrell, T, "Caffe: Convolutional architecture for fast feature embedding," *Proc. of the 22nd ACM international conference on Multimedia*, pp. 675-678, Nov. 2014.

신 광 용

2018년 서울대학교 전기정보공학부 졸업(학사). 2018년~현재 서울대학교 전기·정보공학부 석박사통합과정. 관심분야는 엣지 컴퓨팅, 딥러닝

문 수 묵

서울대학교 컴퓨터공학과 졸업(학사). University of Maryland, College Park 졸업(석사, 박사). 1994년~현재 서울대학교 전기·정보공학부 교수. 관심분야는 가상머신, 명령어-수준 병렬화, 컴파일러 최적화, 웹클라이언트 최적화