모바일 디바이스에서 DNN 가속을 위한 Unaligned Pruning

이광배 ○, 신동군
성균관대학교 전자전기컴퓨터공학과
{kblee93, dongkun}@skku.edu

Unaligned Pruning for Fast DNN Inference on Mobile Devices

Kwangbae Lee○, Dongkun Shin
Department of Electrical and Computer Engineering, Sungkyunkwan University

요 약

Pruning은 깊은 신경망의 연산 및 메모리 접근 비용을 줄이기 위해 사용되는 유망한 네트워크 압축 기법이다. Pruning 기법은 크게 fine-grained pruning과 coarse-grained pruning으로 분류된다. Fine-grained pruning은 정확도의 손실을 줄일 수 있는 반면 불규칙한 네트워크가 생성되어 inference 시간을 줄이기 힘들다. 이와 반대로 coarse-grained pruning은 하드웨어 친화적인 네트워크를 생성하지만, sparsity가 높아지면 정확도가 낮아진다는 문제가 있다. 두 가지 방법의 문제점을 보완하고자 기존 연구에서는 medium-grained pruning을 제안하였는데, 위 방법에서는 탐색 공간을 줄이기 위해 제한된 범위 내에서 가중치 그룹을 선정할 수 있게 하였다. 본 논문에서는 기존 방법에 존재하는 그룹 선택 영역에 관한 제약을 제거하고, greedy 알고리즘을 통해 가중치 그룹을 선택함으로써 같은 정확도를 갖는 조건에서 더 높은 sparsity를 얻을 수 있다.

1. 서 론

깊은 신경망 (Deep Neural Network)은 자연어 처리, 음성 인식, 컴퓨터 시각 과제와 같은 다양한 문제에서 주목할 만한 성과를 거두었다. 복잡한 문제에 대한 높은 정확도를 얻기 위해, 깊은 신경망 모델들은 점점 더 커지고 깊어지고 있다. 결과적으로, 최근의 깊은 신경망 모델을 수행하기 위해서는 엄청난 컴퓨팅 비용과 에너지 소비를 필요로 한다. 특히, 컴퓨팅, 메모리, 전력 자원이 제한된 모바일 기기에서 이러한 크고 깊은 신경망 모델을 실하는 것은 어렵다.

이러한 문제를 해결하기 위해 정확성에 미치는 영향이 적은 가중치 연결을 제거하고, 연결이 제거된 네트워크의 정확성을 회복하기 위해 재 학습하는 네트워크 pruning 기법 [1,2]이 제안되었다. 크고 깊은 신경망은 대게 대량의 불필요한 가중치들을 내부에 가지고 있으므로, pruning 방법을 통해 깊은 신경망의 모델 크기는 큰 정확성의 손실 없이 줄일 수 있다. Pruning을 통해 가중치를 줄이는 방법은 제거하는 granularity에 따라 달라진다. 그림 1(a)에서 볼 수 있듯이 fine-grained granularity pruning 과정은 작은 규모의 가중치만 제거함으로써 상당한 정확도 손실을 막을 수 있지만, 그것은 매우 불규칙한 네트워크를 생성한다. 따라서, 잘라낸 네트워크가 충분한 sparsity를 가지고 있지 않으면 일반 컴퓨팅 시스템에서 inference 속도를 향상시키기가 어렵다.

그림 1(c)와 같은 coarse-grained [3,4,5] 접근방식은 가중치를 영역으로 그룹화 하여 해당 그룹들 중 작은 규모를 가지는 가중치 그룹을 제거하는 방법이다. 예를 들어, 필터, 채널 또는 행/열은 그룹의 영역이 될 수 있다. 그러므로, pruning은 네트워크의 구조화된 sparsity를 생성하며, 네트워크의 실행시간은 모델의 sparsity에 비례하여 일반 컴퓨팅 시스템에서 줄일 수 있다. 그러나, 이 방법은 높은 sparsity를 가질 때, 정확도의 손실이 크다는 단점이 존재한다.

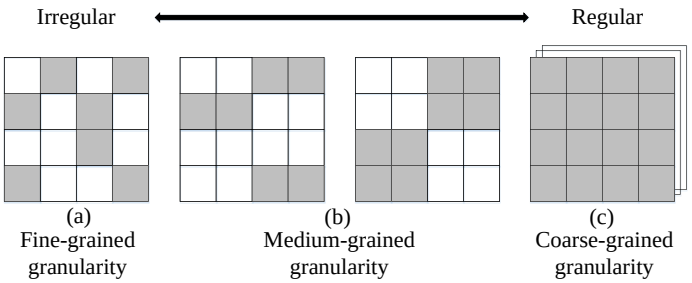

그림 1 4차원 가중치 텐서에서 다른 granularity에 따른 pruned 네트워크 형태

Coarse-grained pruning의 정확도 손실을 줄이기 위해 그림 1(b)와 같이 pruning되는 그룹의 크기를 보다 작게 설정하여 pruning을 진행할 수 있다. 그러나 크기가 작은 그룹을 사용하는 이전 그룹 레벨 pruning 기법 [6, 7]은 정렬된 접근법을 사용했는데, 여기서 주어진 가중치 매트릭스는 먼저 겹치지 않는 그룹으로 분할되며, 각 그룹은 그룹의 크기에 따라 제거되거나 제거되지 않거나 제거된다. 따라서, pruning 과정을 거치고 난 네트워크의 남은 각 그룹은 정해진 일정한 구역 내의 위치한다. 그 결과, 제거되지 않아야 하는 중요한 가중치들은 pruning을 통해 제거되어 정확도의 손실을 최소화하지 못하는 결과를 초래한다.

본 논문에서는 greedy 알고리즘을 통해 정확도의 큰 영향을

이 논문은 2019년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 No.IITP-2017-0-00914, 지능형 IoT 장치용 소프트웨어 프레임워크)

미치는 그룹을 찾아 해당 그룹이 pruning 과정 도중 제거되지 않도록 하는 방법을 제안한다. 이 방법은 기존 그룹 레벨 pruning의 문제점인 정렬된 구역이라는 제약을 제거하고, 가중치의 그룹이 선택될 수 있는 공간을 매트릭스 전체로 확장함으로써 정확성을 높일 수 있다. 그 결과, 기존보다 같은 sparsity를 가지는 네트워크에서 높은 정확성을 가질 수 있다.

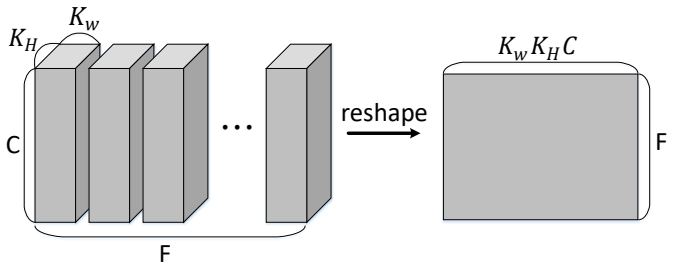

그림 2 GEMM 연산을 위한 가중치 매트릭스 변형

2. 관련 연구

Scalpel[7]은 데이터 병렬 하드웨어 구조를 활용하기 위해 SIMD 유닛에 알맞은 크기로 그룹을 설정하여 pruning하는 방법을 제안하였다. 모든 그룹의 크기는 SIMD 폭과 같으며, 따라서 SIMD 단위가 완전히 활용되도록 하여 성능을 향상시킨다. 하지만 그룹의 차원이 1차원으로 한정되어 있기 때문에 이상적인 성능을 얻기 위해서는 높은 90% 이상의 sparsity를 보유하여야 한다. 한편, 그룹의 차원을 2차원[6]으로 설정할 경우, tiling과 같은 다양한 최적화 기법을 적용할 수 있게 되므로 성능의 향상을 쉽게 끌어올릴 수 있다. 하지만, 높은 sparsity에서는 여전히 정확도의 손실을 막지 못한다.

본 논문에서는 그룹 크기를 1차원과 2차원으로 설정하였을 때, greedy 알고리즘을 적용하여 찾은 결과와 기존 방법을 사용하였을 때의 차이를 정확도와 성능 측면에서 각각 보여준다.

3. Greedy 알고리즘을 통한 그룹 선택

1차원 또는 2차원의 그룹을 선택하는 탐색 공간은 그림 2와 같이 GEMM (General Matrix Multiply) 연산을 위해 변형된 2차원 매트릭스이다. 그림 2는 특정 레이어가 $K_H * K_W$커널, C 채널 그리고 F개의 필터의 가중치를 가지고 있을 때, 이를 $K_H * K_W * C * F$를 가지는 2차원 매트릭스로 변형하는 과정을 보여준다. 최종적으로 변형된 2차원 매트릭스를 탐색공간으로 하여 설정된 sparsity에 맞게 그룹을 선택한다.

$$S_i = \sum_{w \in G_i} |w| \quad (1)$$

위의 식은 2차원 매트릭스에서 생성될 수 있는 그룹에 속하는 가중치들의 합을 S_i로 표현한 것이다. 그룹이 정확도에 영향을 미치는 정도는 해당 그룹의 내에 속하는 가중치들의 합 (S_i)을 기준으로 판단한다. 만약 어떤 그룹의 속하는 가중치들의 합이 작다면, 그 그룹은 정확도의 영향을 미치는 정도가 약하다고 판단되어 최종적으로 제거되지 않은 그룹에 포함되지 않는다.

그룹을 선택하는데 있어 가장 중요한 요소는 특정 레이어가 가져야 하는 sparsity이다. Sparsity가 s라고 주어졌을 때, 특정 레이어에서 살아남아야 하는 그룹의 수 (N)는 $N = (1-s) * K_H * K_W * C * F$와 같이 나타낼 수 있다. Greedy 알고리즘을 통해서 식 (1)에서 구할 수 있는 그룹의 가중치 합이 가장 큰 순서대로 N 개의 그룹을 고른다. 그림 3은 s가 0.625%로 주어졌을 경우, 기존 방법과 같이 그룹을 선택할 수 있는 구역을 제한한 경우와 제약을 해제한 후 greedy 알고리즘을 적용했을 때의 결과를 보여준다. 그림 3에서 볼 수 있듯이, greedy 알고리즘을 적용한 경우에 선택된 가중치의 합을 나타내는 W가 가장 큰 것을 확인할 수 있다.

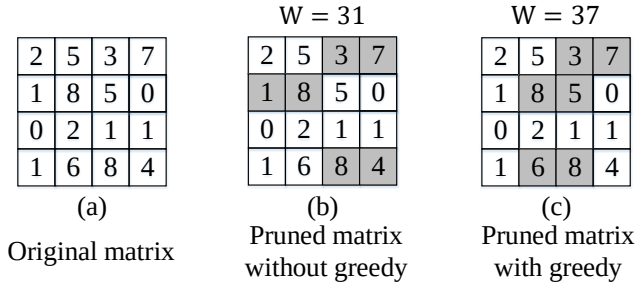

그림 3 그룹 선택에서 greedy 알고리즘 적용 유무에 따른 pruned 네트워크

그룹을 선택할 수 있는 구역의 제약을 제거하였을 때, 다른 행에 존재하는 가중치들이 한 그룹 내에 존재하게 되는 경우를 막아야 한다. 그 이유는 각 행은 다른 결과 값에 반영되기 때문이다. 즉, 각 행이 연산 된 결과는 다른 주소 값을 가지는 메모리 공간에 저장되게 된다. 그렇기 때문에 GPU를 사용하여 연산할 경우, 메모리 배리어와 같이 특별히 처리해주어야 하기 때문에 추가적인 비용이 생긴다. 이를 방지하기 위해서 1차원 그룹을 생성할 경우 2개의 행에 걸치는 것에 대한 예외 처리가 필요하다.

4. 실험

4.1 실험환경

실험에 사용된 하드웨어 플랫폼은 ARMv7 기반의 Cortex-A7 4개와 Cortex-A15 4개로 구성된 옥타코어 CPUs, ARM Mali-T628 GPUs로 구성된 ODROID-XU4이다. 실험에서는 총 6개의 shader core 중 4개만 가지고 있는 디바이스에서 실험하였다.

ACL (Arm Compute Library) [8]은 ARM에서 제공하는 컴퓨터 비전과 머신 러닝 태스크 수행에 필요한 최적화된 라이브러리이다. 아래 실험들은 dense한 네트워크를 ACL을 사용하여 수행하였을 때와 비교하였다.

본 논문의 기법을 평가하기 위해 유명한 VGG-11 네트워크와 CIFAR-10 데이터 셋을 사용하였다. 또한 실험 디바이스인 ODROID-XU4 특성에 따라 1차원 그룹의 경우 그룹 크기는 4, 2차원 그룹의 경우 그룹 크기는 16으로 설정하였다.

4.2 정확도 vs. Sparsity

그림 4에서는 CIFAR-10 데이터 셋과 VGG-11

네트워크에서의 sparsity에 따른 정확도를 나타내고 있다.

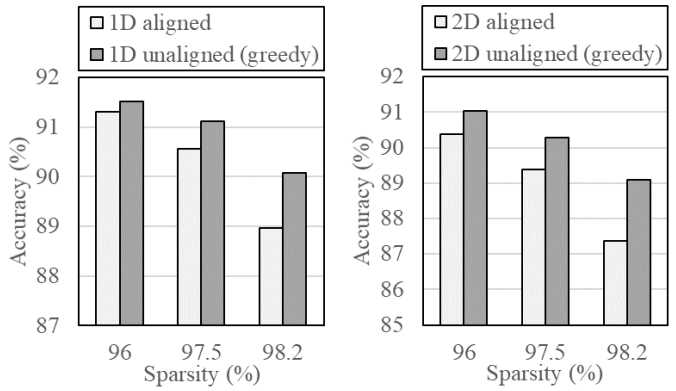

그림 4 VGG-11, CIFAR-10에서의 정확도 vs. Sparsity

그림 4에서 볼 수 있듯이 기존 방법들처럼 그룹을 생성하는 영역의 제약이 존재한다면 중요한 가중치들이 사라지게 되어 정확도의 손실이 발생한다. 이와 같은 현상은 그룹 크기가 1차원, 2차원에 상관없이 그림 4의 98.2%의 sparsity에서 볼 수 있듯이 네트워크의 sparsity가 높아지면 더 뚜렷하게 나타난다. Greedy 알고리즘을 통해 그룹을 찾는 방법에서는 sparsity가 98.2%일 경우 1차원과 2차원일 때 각각 1.12%, 1.71%의 정확도 향상을 얻을 수 있다.

4.3 성능 향상 vs. Sparsity

그림 5에서는 CIFAR-10 데이터 셋과 VGG-11 네트워크에서의 sparsity에 따른 성능을 보여주고 있다. 그림 5의 그래프에서 정규화 된 속도 향상은 dense한 네트워크를 ACL을 사용해 수행하였을 때 대비 향상된 속도를 나타낸 것이다.

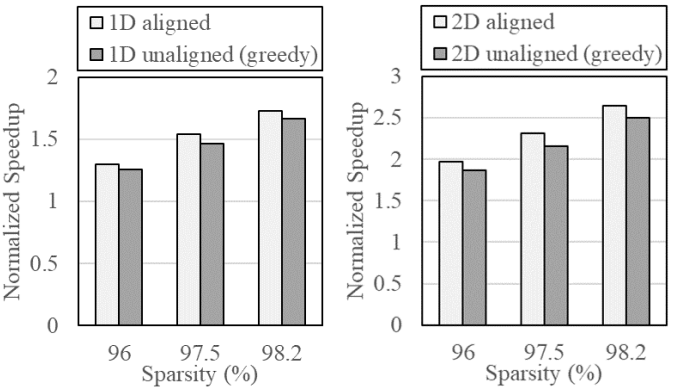

그림 5에서 볼 수 있듯이 2차원 그룹 레벨 pruning의 상대적인 속도 향상이 1차원 그룹 레벨 pruning에 비해 높다. 그 이유는 2차원 그룹 형태로 가중치가 존재하게 되면, 1차원 그룹과는 다르게 tiling을 통한 가중치 재활용 극대화 등의 최적과 기법이 쓰일 수 있기 때문이다. 그 결과, 98.2%의 sparsity를 가질 경우, 1차원과 2차원일 때 각각 1.73, 2.64배 정도의 속도 향상을 얻을 수 있다.

기존 방법과 greedy 알고리즘을 사용하여 pruning 한 네트워크를 수행했을 때, 그림 5와 같이 greedy 알고리즘을 사용했을 때 성능 향상이 약간 저하된다. 그 이유는 정해진 구역의 제약이 사라짐으로써 두 개의 캐시 라인에 데이터가 걸쳐서 존재하는 경우가 생기기 때문에 약간의 캐시미스가 더 발생한다. 그러나 일반적으로 캐시라인 크기는 64바이트 정도이고, 이 때 하나의 캐시라인에는 4개의 32비트 가중치를 포함한 그룹이 4개 존재할 수 있다. 그렇기 때문에 많은 캐시미스를 유발하지 않는다. 또한 sparsity가 높아짐에 따라 남아 있는 그룹의 수도 적기 때문에 이는 큰 영향을 미치지 못한다.

5. 결론 및 향후 연구

본 논문에서는 greedy 알고리즘을 통해 중요한 가중치들을 포함한 그룹을 찾음으로써 기존 pruning 방법보다 향상된 정확도와 sparsity를 가질 수 있다. 즉, 기존의 그룹 레벨 pruning 방법을 사용하면 정확도에 큰 영향을 미치는 중요한 가중치들을 구역의 제약으로 인해 제거하게 된다. 이러한 제약으로 인해 발생하는 중요한 가중치 제거의 문제점을 greedy 알고리즘을 통해 가중치 그룹을 결정함으로써 가중치 손실을 보다 완화시킬 수 있다.

Greedy 알고리즘을 통해 가장 큰 가중치들의 합을 보유하고 있는 그룹을 선택하는 것은 기존 방법보다는 정확도를 향상시키지만 최적의 선택은 아니다. 최적의 그룹을 선택하기 위해서는 다이나믹 프로그래밍을 통해 모든 가중치 그룹의 합을 비교하여 찾는 것이 성능과 정확도를 더욱 향상시킬 수 있을 것으로 기대된다.

참고 문헌

[1] Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." arXiv preprint arXiv:1510.00149 (2015).

[2] Han, Song, et al. "Learning both weights and connections for efficient neural network." Advances in neural information processing systems. 2015.

[3] He, Yihui, Xiangyu Zhang, and Jian Sun. "Channel pruning for accelerating very deep neural networks." Proceedings of the IEEE International Conference on Computer Vision. 2017

[4] Luo, Jian-Hao, Jianxin Wu, and Weiyao Lin. "Thinet: A filter level pruning method for deep neural network compression." Proceedings of the IEEE international conference on computer vision. 2017

[5] Wen, Wei, et al. "Learning structured sparsity in deep neural networks." Advances in neural information processing systems. 2016.

[6] Narang, Sharan, Eric Undersander, and Gregory Diamos. "Block-sparse recurrent neural networks." arXiv preprint arXiv:1711.02782 (2017).

[7] Yu, Jiecao, et al. "Scalpel: Customizing dnn pruning to the underlying hardware parallelism." ACM SIGARCH Computer Architecture News. Vol. 45. No. 2. ACM, 2017.

[8] https://github.com/ARM-software/ComputeLibrary