모바일 장치에서 빠른 뉴럴 네트워크 추론을 위한 비정렬된 블록 희소성 활용

이하윤°, 신동군
성균관대학교 전자전기컴퓨터공학과
lhy920806@skku.edu, dongkun@skku.edu

Exploiting Unaligned Block Sparsity for Fast Neural Network Inference on Mobile Devices

Hayun Lee°, Dongkun Shin
Department of Electrical and Computer Engineering, Sungkyunkwan University

요 약

최근 딥 러닝이 다양한 분야에서 우수한 성능을 보이지만, 큰 연산량으로 인해 자원이 한정된 모바일 장치에서 수행하기에는 제한이 있다. 이러한 문제를 해결하기 위해 pruning과 같은 딥 러닝 모델 압축 기술이 제안되었다. Pruning은 딥 러닝 모델의 가중치를 제거하여 모바일 장치에서도 딥 러닝 응용을 효과적으로 수행할 수 있도록 한다. 특히, 최근에 제안된 unaligned pruning은 블록 선택 영역에 관한 제약을 제거하여 기존 pruning 연구보다 우수한 성능을 달성하지만, unaligned pruning을 위한 학습 방법 및 라이브러리에 대한 고려가 부족한 상태이다. 본 논문에서는 unaligned pruning을 위한 프레임워크를 제안한다. 해당 프레임워크에서는 unaligned pruning을 위한 레이어별 sparsity를 구하고, ADMM을 통해 가중치와 pruning 마스크를 동시에 최적화한다. 또한, unaligned sparse 연산을 위한 딥 러닝 엔진을 개발하여 unaligned pruning으로 압축된 모델을 실제 모바일 장치에서 빠르게 수행할 수 있다.

1. 서 론

딥 러닝은 컴퓨터 비전, 음성 인식, 자율주행차 등 다양한 분야에서 좋은 성능을 보이며 다양한 응용으로 사용되고 있다. 그러나, 딥 러닝의 연산량은 매우 크기 때문에 한정된 자원을 가진 모바일 장치에서 수행하기에는 제한이 있다. 이를 해결하기 위해 pruning, quantization과 같은 딥 러닝 모델 압축 기술이 제안되어 왔다.

특히 pruning의 경우 channel 또는 filter 단위로 pruning을 수행하는 structured pruning[1, 2]이 많이 제안되었다. Structured pruning은 기존 딥 러닝 엔진으로 수행할 수 있다는 장점이 존재하지만, pruning 되는 단위가 커 정확도 손실이 크다는 문제가 존재한다.

최근에는 structured pruning의 문제점을 극복하기 위해 weight 또는 블록 단위로 pruning을 수행하는 unstructured pruning[3, 4]이 제안되고 있다. Unstructured pruning은 structured pruning보다 더 세밀한 단위로 pruning 할 수 있어 높은 압축률에서도 정확도 손실이 적다. 또한, 모바일 CPU의 SIMD(Single Instruction Multiple Data)를 활용하는 sparse 연산 라이브러리(XNNPACK[7])를 사용하여 기존 structured pruning보다 좋은 성능을 보여준다.

그러나 기존 블록 단위 pruning에서 모든 블록은 블록의 크기의 배수만큼 떨어져 있어야하는 align의 제약으로 인해 weight 단위 pruning과 비교하여 정확도 하락이 존재한다. 이를 개선하기 위하여 블록 간의 간격에 제약을 두지 않은 unaligned pruning[5]이 제안되었지만 다음과 같은 한계점이 존재한다. 첫번째, unaligned pruning을 위한 레이어별 sparsity를 구하는 방법이 없다. 두번째, one-shot pruning 방식을 사용하기 때문에 최적의 pruning 마스크를 찾기에 한계가 있다. 마지막으로 모바일 장치에서 unaligned pruning된 모델을 수행하기 위한 라이브러리가 없다.

본 논문에서는 unaligned pruning을 위한 프레임워크를 제안한다. 해당 프레임워크에서는 unaligned pruning을 위한 레이어별 sparsity를 구하고, ADMM[6]을 통해 가중치와 pruning 마스크를 동시에 최적화한다. 또한, unaligned sparse 연산을 위한 딥 러닝 엔진을 개발하여 unaligned pruning으로 압축된 모델을 실제 모바일 장치에서 빠르게 수행할 수 있다.

이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.IITP-2017-0-00914, 지능형 IoT 장치용 소프트웨어 프레임워크)

2. 관련 연구 및 배경 지식

Unaligned block-level pruning[5]은 살아남는 블록 간의 간격에 대한 제약을 없애면서 모바일 장치에서 모델 압축 및 가속을 수행하였다. Unaligned pruning을 수행하기 위해서는 최적의 unaligned 블록을 찾아야 한다. Unaligned 블록 pruning을 수행하는 함수를 B 라고 할 때, 방법에 따라 optimal 알고리즘과 greedy 알고리즘이 존재한다. 이때, Weight는 $\mathbf{W}$ 이고, 살릴 블록의 개수가 n 이고, $S(\mathbf{W}, n)$ 을 n 개의 unaligned 블록이 선택된 $\mathbf{W}$ 의 집합이라고 할 때, 각각의 알고리즘은 다음과 같이 정의된다.

$$B_{opt}(\mathbf{W}, n) = \arg\max_{\mathbf{W}'} \sum_{w \in \mathbf{W}'} |w|, \ s.t. \ \mathbf{W}' \in S(\mathbf{W}, n).$$

$$B_{greedy}(\mathbf{W}, n) = \mathbf{W}_n^g, \mathbf{W}_{i+1}^g = \mathbf{W}_i^g + B_{opt}(\mathbf{W} - \mathbf{W}_i^g, 1), \mathbf{W}_0^g = S(\mathbf{W}, 0).$$

Optimal 알고리즘은 greedy 알고리즘보다 좋은 성능을 보여주지만, 탐색 시간이 길다는 문제점이 있다.

3. Unaligned Block-level Pruning 프레임워크

그림 1은 unaligned block-level pruning 프레임워크의 전체 구성을 보여준다.

3.1 Density-based Score

기존의 pruning 연구에서는 레이어별 sparsity를 구하기 위해 주로 magnitude-based pruning(MP)[8]을 사용했다.

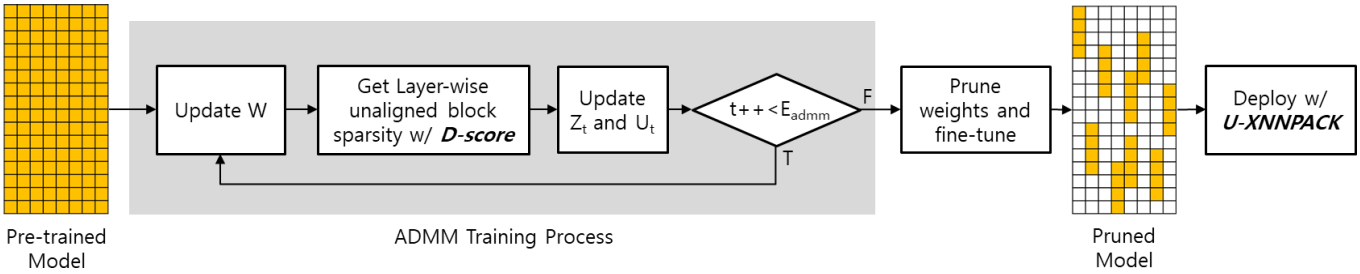


그림 1 Unaligned block-level pruning 프레임워크 전체 구조

MP에서는 전역(global) 또는 레이어별로 threshold 값을 정한 후, weight 값의 크기(magnitude)가 threshold보다 작으면 pruning 한다. 블록 단위 pruning에서 MP를 사용할 경우, 각 aligned 블록에 포함된 weight 크기의 합을 threshold와 비교하며, threshold보다 작을 경우 해당 블록을 pruning 한다.

이전 unaligned pruning 연구[5]에서는 레이어별 sparsity를 얻기 위해 unaligned에 대한 고려 없이 aligned pruning 방식으로 구한 sparsity를 사용했다. 본 논문에서는 unaligned pruning의 특성을 고려하려 레이어별 sparsity를 구하기 위해 density-based score(D-score)를 제안한다. D-score D 는 Weight $\mathbf{W}$에 대하여 unaligned 블록을 $n-1$개 선택했을 때와 n 개 선택했을 때의 weight 크기의 합의 차이를 의미하며, 다음과 같이 정의된다.

$$D(\mathbf{W},n)=\sum_{w\in B(\mathbf{W},n)}|w|-\sum_{w\in B(\mathbf{W},n-1)}|w|$$

즉, D-score는 B 함수가 찾은 해를 기반으로 살리는 블록의 개수에 따라 어느 정도의 효과가 있는지를 알려주는 함수이다. Target sparsity τ가 주어졌을 때, 총 l개의 레이어에 대한 레이어별 살리는 unaligned 블록의 개수 N 은 D-score를 이용하여 다음과 같이 구할 수 있다.

$$N=\underset{N=\{n_1,\dots,n_l\}}{\arg\max}\sum_{i=1}^{l}D(\mathbf{W}_i,n_i),\ s.t.\frac{\sum_{i=1}^{l}C_{zero}\left(B(W_i,n_i)\right)}{\sum_{i=1}^{l}C(W_i)}\le\tau$$

C 와 C_{zero} 는 각각 입력으로 받은 텐서의 원소의 개수와 값이 0인 원소의 개수를 반환하는 함수이다. Unaligned 블록 탐색 알고리즘은 $D(\mathbf{W},n)\ge D(\mathbf{W},n+1)$와 같이 동일한 weight $\mathbf{W}$에 대해 n의 값이 증가함에 따라 항상 D-score가 감소하는 특징이 있다. 따라서, N 에 대한 문제를 threshold t 를 도입하여 다음과 같이 동등하게 변형할 수 있다.

$$N=\underset{N=\{n_1,\dots,n_l\}}{\arg\max}\sum_{i=1}^{l}n_i\ ,\ s.t.\ D(\mathbf{W}_i,n_i)\ge t,i=1,\dots,l$$

이때, 블록의 크기는 BS이면 threshold t는 다음과 같다.

$$t=\text{percentile}\left(\text{sort}\left(\left\{D(\mathbf{W}_i,x_i)|1\le i\le l,1\le x_i\le\frac{C(W_i)}{BS}\right\}\right),\tau\right)$$

먼저 sort 함수를 통해 계산 가능한 모든 D-score를 오름차순으로 정렬한다. 그 후, percentile 함수를 통해 τ번째 백분위수 값을 반환한다.

그림 2는 D-score를 이용하여 블록 크기가 3인 unaligned 블록 pruning을 수행하는 예제를 보여준다. (a), (b), (c), (d)는 각각 unaligned 블록을 0, 1, 2, 3개 선택했을 때의 선택된 블록을 보여준다. Threshold가 5라고 가정할 때, 앞서 설명한

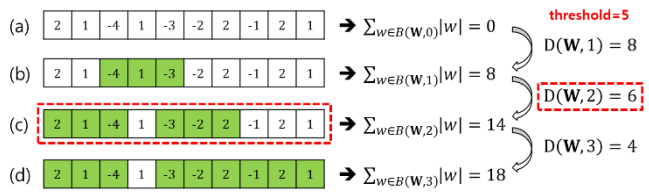


그림 2 D-score를 통한 unaligned 블록 pruning

수식에 따라 해당 레이어에서는 2개의 블록이 선택된다. D-score는 ADMM 훈련 과정 중 한 epoch에 대한 weight 훈련이 완료되었을 때, 레이어별 unaligned 블록의 개수를 다시 계산하기 위해 사용될 수 있다.

3.2 Unaligned sparsity를 고려한 ADMM 훈련 과정

f 는 기존의 목적함수를 의미할 때, unaligned pruning은 아래와 같이 weight 형태에 제한조건이 있는 최적화 문제로 정의할 수 있다.

$$\underset{\{\mathbf{W}_i\}}{\text{minimize}}\ f\left(\{\mathbf{W}_i\}_{i=1}^{N}\right),\ s.t.\ \mathbf{W}_i\in S(\mathbf{W}_i,n_i),i=1,\dots,N$$

본 논문에서는 해당 문제를 풀기 위하여 제한조건이 존재하는 최적화 문제를 푸는 방법인 ADMM[6]을 적용하였다. ADMM을 적용하기 위해서, 보조변수 (auxiliary variable) $\mathbf{Z}$와 쌍대변수 (dual variable) $\mathbf{U}$를 도입하여 아래의 식과 같이 두 개의 하위 문제로 분해했다.

$$\underset{\{\mathbf{W}_i\}}{\text{minimize}}\ f(\{\mathbf{W}_i\})+\sum_{i=1}^{N}\frac{\rho_i}{2}\|W_i-Z_i^k+U_i^k\|_F^2$$

ADMM 최적화에서는 $\mathbf{W}$, $\mathbf{Z}$, $\mathbf{U}$에 대한 업데이트를 번갈아 가며 최적화를 수행한다. 먼저, W는 기존의 손실 함수와 두 번째 하위 문제에 대한 손실 함수를 최적화하도록 업데이트 한다. $\mathbf{W}$에 대한 업데이트가 끝난 이후 D-score를 통해 N 을 다시 계산한 후, $\mathbf{Z}$와 $\mathbf{U}$를 각각 업데이트한다. 이때, $\mathbf{Z}$와 $\mathbf{U}$는 다음과 같이 해석적 방법에 의해 구해진 해로 업데이트를 수행한다. $\mathbf{Z}$ 업데이트시 n_t은 이전 단계에서 구한 레이어별 살리는 unaligned 블록의 개수 N을 사용한다.

$$\mathbf{Z}_t=B(\mathbf{W}+\mathbf{U}_{t-1},n_t),\qquad \mathbf{U}_t=\mathbf{U}_{t-1}+\mathbf{W}-\mathbf{Z}_t$$

ADMM 훈련은 총 E_{admm}만큼 반복적으로 수행한다. ADMM 훈련이 완료된 후에는 N에 기반하여 weight를 pruning하고, pruning된 weight를 제외한 상태에서 fine-tuning을 수행하여 최종 sparse 모델을 만들어낸다.

3.3 Unaligned sparse 연산을 위한 딥 러닝 엔진

Unstructured pruning을 수행한 모델은 XNNPACK[7]과 같이 sparse 연산을 제공하는 딥 러닝 엔진을 사용하여 효율적으로 수행할 수 있다. 그러나, 이러한 딥 러닝 엔진들은 모두 aligned pruning에 대해서만 고려하고 있기 때문에 unaligned pruning된 모델을 효율적으로 수행하지 못한다.

따라서, 본 논문에서는 unaligned pruning된 모델을 효율적으로 수행할 수 있는 U-XNNPACK을 개발하였다. 그림 3은 U-XNNPACK의 블록 크기가 2인 MRx2 SpMM 함수들의 파이썬 스타일 슈도 코드를 보여준다. 여기서 MRx2는 output tile의 크기를 의미한다. 또한, 설명의 간결함을 위해 sparse format이 아닌 dense format을 가정하여 작성하였으며, 실제 수행은 sparse format으로 수행하게 된다. 마지막으로 `in`, `w`, `out`은 각각 input, weight, output 데이터를 의미하고, `c0`, `c1`은 캐시에 상주하는 데이터를 의미한다.

그림 3 (a)는 기존 XNNPACK에 존재하는 MRx2 SpMM 코드이다. Output tile 하나가 전부 연산 될 때까지 결과는 `c0`와 `c1`에 누적되며, 최종적으로 `out`에 저장한 이후 다음 tile의 연산을 수행하게 된다. 블록 단위 커널이 weight 단위

커널보다 성능이 좋은 이유는 그림과 같이 input 데이터 _in이 블록의 크기만큼 재사용 될 수 있기 때문이다.

그림 3 (b)는 unaligned sparsity를 위해 추가된 unaligned MRx2 SpMM 코드이다. Unaligned 연산의 특성 상 (a)와 같이 블록을 유지하면서 output tile을 만들 수 없으므로 겹쳐진 output tile을 통해 연산을 수행한다. 이때, c1에 저장된 output의 결과를 c0으로 복사해서 아직 계산이 완료되지 않은 output을 캐시에 상주하도록 한다. 여전히 input 데이터 _in의 재사용이 이루어지고 있기 때문에 효율적이며, 이후 실험에서도 실행 속도는 Aligned MRx2 SpMM과 거의 유사한 것을 확인할 수 있다.

```
# Cache: c0[MR], c1[MR]
for m in range(0, M, MR):
  for n in range(0, N, 2):
    c0 = 0
    c1 = 0
    for k in get_k(m, n):
      _in = in[k, m:m+MR]
      _w = w[n:n+2, k]
      c0 += _in * _w[0]
      c1 += _in * _w[1]
    out[n  , m:m+MR] = c0
    out[n+1, m:m+MR] = c1
```

(a) Aligned MRx2 SpMM

```
# Cache: c0[MR], c1[MR]
for m in range(0, M, MR):
  c1 = 0
  for n in range(N-1):
    c0 = c1
    c1 = 0
    for k in get_k(m, n):
      _in = in[k, m:m+MR]
      _w = w[n:n+2, k]
      c0 += _in * _w[0]
      c1 += _in * _w[1]
    out[n, m:m+MR] = c0
  out[N-1, m:m+MR] = c1
```

(b) Unaligned MRx2 SpMM

그림 3 U-XNNPACK의 SpMM pseudo code

4. 실 험

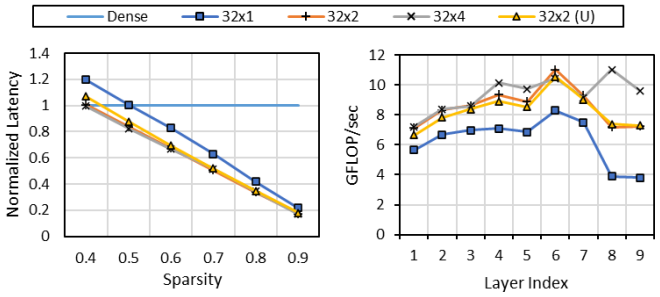
4.1 실험 환경

본 논문의 기법을 평가하기 위해 MobileNet V1 (MBv1) 모델과 CIFAR-100 데이터셋을 사용하였다. MBv1의 경우 첫 번째 convolution 레이어의 stride 값을 1로 변경하였다. Block-level Pruning을 위한 블록의 크기는 2로 설정하였으며, 1x1 Conv 레이어에 대해 pruning을 수행하였다. 또한, unaligned 블록을 찾기 위해 greedy 알고리즘을 사용하였다.

학습된 모델의 수행시간은 ARMv8 기반의 CPU가 탑재된 Pixel 1을 사용하여 측정하였다. 모든 실험은 U-XNNPACK을 사용하여 측정하였으며, CPU 쓰레드는 1개로 지정하였다.

4.2 실험 결과

그림 4는 U-XNNPACK의 SpMM 성능을 측정한 결과이며, 32x1, 32x2, 32x4는 각각 블록 크기 1, 2, 4인 커널을 의미한다. 또한, (U)는 unaligned 커널임을 의미한다. 그림 4 (a)는 M, N, K가 각각 16, 512, 512인 SpMM의 sparsity에 따른 실행 시간을 보여주며, 실행 시간은 dense의 수행시간을 기준으로 정규화 하였다. 32x2 커널과 32x2 (U) 커널의 수행시간이 거의 비슷하며, 이는 unaligned로 pruning을 하더라도 실제 장치에서 효율적으로 동작할 수 있음을 의미한다. 그림 4 (b)는 80% sparsity에서 MBv1에 존재하는 레이어들의 각 SpMM 커널에 따른 성능을 보여준다. 다양한 크기에 대해서도 32x2 (U)의 성능은 32x2와 거의 유사함을 볼 수 있다.



(a) Sparsity에 따른 latency (b) 80% sparsity에서 레이어별 연산 속도

그림 4 U-XNNPACK SpMM 성능

그림 5는 MBv1 1.0과 0.75 모델을 다양한 target sparsity로 압축한 결과를 보여준다. 실험에서 "Unaligned"의 학습은 [5]에서 수행한 방법과 같다. 또한, ADMM과 DS는 각각 ADMM과 D-score를 사용하였을 때를 의미한다. Unaligned 결과에서 DS를 사용하지 않는 경우 aligned의 sparsity를 사용하였다. 실험 결과 ADMM을 사용할 경우, 사용하지 않은 경우보다 더 높은 target sparsity에 대해서도 잘 학습되는 것을 볼 수 있다. 또한, D-score를 사용할 경우, 사용하지 않은 경우보다 더 좋은 결과를 보여준다. 이는 D-score가 unaligned pruning에서 weight 크기에 대한 정보를 잘 표현하기 때문이다.

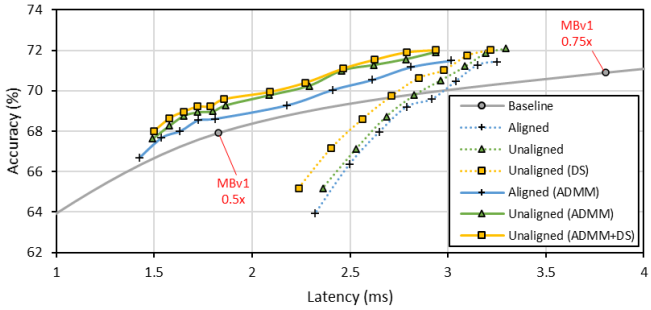


그림 5 CIFAR-100에서 MBv1의 Latency vs. Accuracy

5. 결론 및 향후 연구

본 논문에서는 unaligned block-level pruning을 위한 프레임워크를 제안한다. 해당 프레임워크를 통해 unaligned pruning을 위한 training을 수행하고, unaligned pruning된 모델을 딥 러닝 엔진을 통해 실제 모바일 장치에서 기존 pruning 방법보다 향상된 정확도와 수행 시간을 가질 수 있다.

현재 해당 프레임워크는 하드웨어를 고려하지 않고 pruning을 수행하고 있다. 향후 연구로는 다양한 모바일 장치에 대해서 더 개선된 pruning 및 수행을 위하여 커널 생성 및 하드웨어를 고려한 pruning을 연구할 계획이다.

참고 문헌

[1] 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

[2] 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

[3] Elsen, Erich, et al. "Fast sparse convnets." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020.

[4] Zhao, Tianli, et al. "Architecture Aware Latency Constrained Sparse Neural Networks." *CoRR*, abs/2109.00170, 2021.

[5] Lee, Kwangbae, et al. "Flexible group-level pruning of deep neural networks for on-device machine learning." *2020 Design, Automation & Test in Europe Conference & Exhibition*, IEEE, 2020.

[6] Ren, Ao, et al. "Admm-nn: An algorithm-hardware co-design framework of dnns using alternating direction methods of multipliers." *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019.

[7] XNNPACK. https://github.com/google/XNNPACK

[8] Han, Song, et al. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." *CoRR*, abs/1510.00149, 2015.