불균일 캐시 구조에 최적화된 컨볼루션 연산 가속 데이터 전처리 알고리즘

강석봉[1], 강민구[2], 한태희[3*]
[1,2]성균관대학교 전자전기컴퓨터공학과, [3*]성균관대학교 반도체시스템공학과
[1]bongbon9@g.skku.edu, {[2]mingu.kang, [3*]than}@skku.edu

Convolution Acceleration Data Preprocessing Algorithm Optimized for NUCA

Suk Bong Kang[1], Min Gu Kang[2], Tae Hee Han[3*]
[1,2]Department of Electrical and Computer Engineering, Sungkyunkwan Univ.
[3*]Department of Semiconductor Systems Engineering, Sungkyunkwan Univ.

요 약

인공 신경망에서 CNN 연산의 사전 작업으로 수행되는 IM2COL(Image to Column) 동작은 행렬 데이터를 GEMM 연산이 지원하는 데이터형으로 차원 변환하여 행렬 직접 컨볼루션 방식에 비해 총 연산 시간을 획기적으로 단축한다. 그러나 기존 IM2COL 알고리즘은 동일 데이터의 중복 요청이 잦음에도 데이터 요청 과정이 캐시 효율성을 고려하지 않아 과도한 메모리 접근을 유발했다. 본 논문은 IM2COL 알고리즘의 행렬 데이터 요청 순서를 개선하여 메모리 접근 횟수와 뱅크 충돌 횟수를 최소화하는 캐시 효율성 증대 IM2COL 알고리즘을 제안하고, 제안 알고리즘을 적용하였을 때 다수 코어와 불균일 캐시 구조 (NUCA)로 구성된 시스템에 적합한 태스크 매핑 방식을 논한다. SniperSim 시뮬레이션 환경에서 Darknet 프레임워크로 실험하였을 때, 인공 신경망의 이미지 인식 시 컨볼루션 연산 중 IM2COL 함수 실행시간 비중이 평균 14.12% 감소하였으며, 신경망 전체 실행시간은 평균 12.22% 감소하였다.

Ⅰ. 서 론

인공 신경망은 수많은 데이터를 처리하는 서버부터 모바일 디바이스까지 다양한 분야에 적용되고 있다. 인공 신경망의 과도한 연산 요구량을 극복하고 처리량을 향상하기 위하여, 행렬 연산을 병렬화하고 다수 코어를 이용하여 다수 쓰레드를 동시 수행하는 기법이 사용된다. 그러나, 이때 각 쓰레드의 데이터를 저장할 캐시의 용량 및 면적 증가 역시 요구되는데, 기존 균일 캐시 구조 기반 시스템은 면적이 증가하면 캐시 전체의 접근 지연 시간이 증가하는 문제점이 있다. 이를 해결하기 위해 불균일 캐시 구조(Non-Uniform Cache Architecture, NUCA)[1]가 제시되었다. NUCA 는 논리적으로는 하나의 캐시로 동작하지만, 물리적으로는 뱅크 단위로 분리된다. NUCA 에서 각 뱅크에 대한 접근 지연 시간은 균일하지 않으며, 캐시의 전체 면적과 상관없이 해당 뱅크와 요청 코어 간의 물리적 배선 길이가 결정한다.

코어들의 캐시 요청 상태에 따라, NUCA 의 접근 지연 시간은 데이터 전송 경로 및 뱅크 충돌로 인해 기대 시간보다 증가할 수 있으며, 이는 일련의 태스크를 수행할 각 코어와 그 처리 순서를 결정하는 태스크 매핑 정책을 개선하여 완화할 수 있다. 특히, 단순한 여러 개의 태스크로 분할 가능하며, 분할된 각 태스크가 중복 데이터 라인을 반복적으로 요청하는 동작에 대한 태스크 매핑의 최적화는 더 큰 개선 효과를 기대할 수 있다.

기존 IM2COL 알고리즘

```
1:  k = 0
2:  for (v = 0; v < H_k; v++)                          //ker_col
3:      for (u = 0; u < W_k; u++)                      //ker_row
4:          for (y = 0; y < H_o; y++)                  //img_col
5:              for (x = 0; x < W_o; x++)              //img_row
6:                  A_{x,y}[k] = a_{x·s+u, y·s+v}
7:                  B_{x,y}[k] = b_{u,v}
8:                  k++
```

그림 1. IM2COL 알고리즘

IM2COL[2]은 상기한 특성을 가지는 동작의 대표적인 예시로, 이미지 행렬에서 커널 행렬이 곱해지는 각각의 부분행렬들을 추출하고 GEMM 연산이 지원하는 벡터 형태로 변환하여 나열하여 다시 저장하는 동작이다. 그림 1 은 기존의 IM2COL 알고리즘으로, $a_{i,j}, b_{i,j}$는 각각 $H \times W$ 이미지 행렬, $H_k \times W_k$ 커널 행렬의 원소를 의미한다. 이때, $H_o \times W_o$ 출력 행렬의 (x, y)항은 차원 변환된 벡터 $A_{x,y}, B_{x,y}$ 에 대해 $A_{x,y}B_{x,y}^T$ 의 벡터곱으로 구할 수 있다. ($H_o = \frac{H-H_k}{s} + 1$, $W_o = \frac{W-W_k}{s} + 1$; s는 스트라이드)

사전 작업 수행 시간과 메모리에 중복 데이터를 추가 저장하는 오버헤드에도 불구하고, IM2COL 을 사용한 CNN 연산은 행렬 직접 컨볼루션 방식 대비 총 컨볼루션 연산 시간을 200 배 이상 단축한다. 하지만 이러한 동작은 과도한 메모리 접근을 발생시켜, 인공 신경망의 수행 시간이 코어의 연산 성능과 메모리 계층의 접근 시간에 종속적인 문제를 가진다. 본 논문은 이러한 문제를 해결을 위해 IM2COL 알고리즘 동작 시 캐시의 메모리 계층 접근 빈도를 최소화하는 관점으로 접근한다.

Ⅱ. 본론

캐시 효율성 증대 IM2COL 알고리즘

```
1:  k = 0
2:  for (v = 0; v < s; v++)                            //stride_col
3:      for (u = 0; u < s; u++)                        //stride_row
            l_H = ⌈(H_k − v)/s⌉
            l_W = ⌈(W_k − u)/s⌉
4:          for (g = 0; g < l_H; g++)                  //ker*_col
5:              for (f = 0; f < l_W; f++)              //ker*_ row
6:                  for (y = 0; y < H_o; j++)          //img_col
7:                      for (x = 0; x < W_o; i++)      //img_row
8:                          A_{x,y}[k] = a_{(x+f)·s+u,(y+g)·s+v}
9:                          B_{x,y}[k] = b_{f·s+u,g·s+v}
10:                         k++
```

그림 2. 캐시 효율성 증대 IM2COL 알고리즘

본 논문은 기존 IM2COL 알고리즘이 커널 행렬의 원소를 순차적으로 순회하는 (u,v) 이중 루프를 사용하는 것과 달리, 커널 행렬의 원소를 스트라이드마다 건너뛰며 접근하여 중복 데이터를 연속하여 처리하는 캐시 효율성 증대 IM2COL 알고리즘(그림 2)을 제안한다. l_H, l_W 은 (u,v)에 따라 커널 행렬에 대한 접근 횟수를 결정한다. 제안 IM2COL 알고리즘은 다음과 같은 특성을 가진다.

(1) 어떤 이미지 행렬 데이터 $a_{i,j}$가 차원 변환된 벡터 $A_{x',y'}[k']$에서 요청되었다면, 이후 동일한 데이터 $a_{i,j}$ 를 요청하는 모든 벡터 $A_{x,y}$ 는 $A_{x',y'}$ 의 순서*보다 작은 순서를 가짐. 즉, 각 출력 벡터 $A_{x,y}$의 생성을 병렬화하여 (그림 2 의 6, 7 번 루프) 서로 다른 코어에 배치하였을 때, 인접한 일련의 코어에 연속한 순서의 벡터의 생성을 배치하여 요청 데이터의 이동 경로를 단순화할 수 있음.
*벡터 $A_{x,y}$의 순서를 $x + W_o \cdot y$로 정의함.

(2) (u,v)가 변경되면(그림 2 의 2, 3 번 루프), 이미 요청된 데이터는 다시 요청되지 않음. 이미지 행렬 원소는 최대 $\max(l_H \cdot l_W) = \left\lceil \frac{H_k}{s} \right\rceil \cdot \left\lceil \frac{W_k}{s} \right\rceil$ 회 요청됨.

(3) (u,v,g) 가 동일한 경우, 어떤 f' 에서 데이터를 요청하지 않은 코어는 이후 (즉, $f > f'$ 인 경우) 데이터 요청 및 연산이 필요하지 않음.

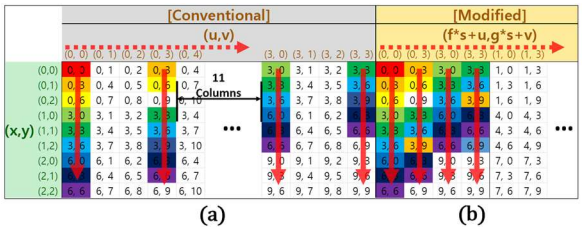

그림 3. IM2COL 알고리즘에 따른 데이터 요청 순서 (a) 기존 IM2COL (b) 캐시 효율성 증대 IM2COL

그림 3 은 IM2COL 알고리즘이 $[H = W = 11, H_k = W_k = 5, s = 3]$ 의 조건에서 이미지 행렬의 원소를 요청하는 순서를 비교한다. 표 내부의 좌표는 요청되는 이미지 행렬의 원소이며, 가로 방향 색인(회색, 노란색)은 루프 상의 커널 행렬의 원소 (u,v) 또는 $(f \cdot s + u, g \cdot s + v)$를, 세로 방향 색인(초록색)은 출력 행렬의 원소 (x,y) 를 나타낸다. 첫 번째 커널 원소 $(0,0)$에 곱해지는 이미지 행렬의 원소들에 대해, 이후 요청되는 동일한 원소를 각각 같은 색으로 칠하였다. 기존 IM2COL 알고리즘에서 중복되는 이미지 행렬 원소의 요청 간에 긴 시간 간격이 발생하는 것과 달리, 제안 IM2COL 알고리즘은 중복되는 데이터의 요청을 연속한 시간에 처리되도록 재정렬한다.

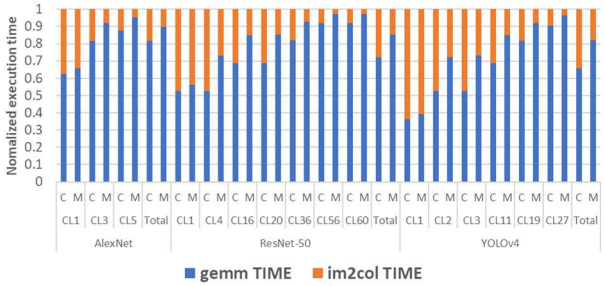

그림 4. Darknet 에서의 IM2COL 알고리즘 실행 결과 (C: 기존 IM2COL, M: 캐시 효율성 증대 IM2COL)

SniperSim[3] 시뮬레이터로 [CPU: intel x64, 1GHz clock. L1I, L1D: 32kB. L2: 512kB. L3: 16MB] 환경을 가정하고, 신경망 프레임워크 Darknet[4]을 이용하여 캐시 효율성 증대 IM2COL 알고리즘과 기존 IM2COL 알고리즘의 성능을 비교하였다. 그림 4 는 AlexNet[5], ResNet-50[6], YOLOv4[7]에서 768×576 크기의 3 채널 이미지 1 장을 인식하는 동안 IM2COL 함수 실행 시간 및 GEMM 연산 실행 시간을 정규화하여 그 비중을 나타낸 그래프다. 각 컨볼루션 레이어(그림 4 의 CL#) 별로 실행 시간을 측정하였고, 동일한 조건의 레이어는 표현을 생략하였으나 총합 실행 시간(그림 4 의 Total) 계산에 포함하였다. 제안 IM2COL 알고리즘은 기존 대비 전체 CNN 연산에서 IM2COL 함수 실행시간 비중을 평균 14.12% 감소하였으며, 전체 신경망 실행 시간의 평균 감소율은 12.22%로 유의미한 개선을 보였다. 단, CNN 연산의 첫 IM2COL 동작(CL1)에서 실행 시간 감소율이 평균 9.8%로 다른 레이어보다 작은데, 이는 이미지 행렬 데이터 초기 접근 시 데이터의 메모리 상 공간지역성이 부족하여 발생한 지연이 제안 IM2COL 알고리즘으로 얻은 이득을 상쇄하는 것으로 해석할 수 있다.

Ⅲ. 결론

본 논문에서는 데이터 요청 순서를 개선하여 캐시의 중복 메모리 요청을 최소화하는 캐시 효율성 증대 IM2COL 알고리즘을 제시하여, CNN 연산에서 데이터 전처리 시간의 비중을 평균 14.12% 감소하였다. 또한, 제안 알고리즘은 중복 요청되는 데이터의 요청을 (1) 인접한 코어에서 (2) 연속한 시간에 (3) 단방향의 데이터 전송 경로를 가지게 유도하여, 병렬 연산에서 캐시의 메모리 접근을 최소화하고, NUCA 내 데이터 이동 경로를 규칙적으로 형성하여 요청 데이터들 사이의 전송 경로 충돌로 인한 캐시 지연 시간 최소화를 기대할 수 있다.

ACKNOWLEDGMENT

본 연구는 2022 년도 정부(과학기술정보통신부)의 재원으로 한국연구재단-시스템반도체융합전문인력육성사업(No. 2020M3H2A1076786), 2022 년도 산업통상자원부 및 산업기술평가관리원(KEIT) 연구비 지원('20011074'), 교육부 및 한국연구재단의 4 단계 두뇌한국 21 사업(4 단계 BK21 사업)으로 지원된 연구임.

참 고 문 헌

[1] C. Kim, D. Burger, and S. W. Keckler, "An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches," ACM SIGPLAN Notices, vol. 37, no. 10. Association for Computing Machinery (ACM), pp. 211-222, Oct. 2002.

[2] Chellapilla, Kumar, Sidd Puri, and Patrice Simard. "High performance convolutional neural networks for document processing," Tenth international workshop on frontiers in handwriting recognition, Suvisoft, 2006.

[3] T. E. Carlson, W. Heirman, S. Eyerman, I. Hur, and L. Eeckhout, "An evaluation of high-level mechanistic core models," ACM Trans. Archit. Code Optim., vol. 11, no. 3, pp. 1-25, Oct. 2014.

[4] J. Redmon. "DarkNet: Open Source Neural Networks in C," 2016, (http://pjreddie.com/darknet/).

[5] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks," Communications of the ACM 60.6, pp. 84-90, 2017.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," arXiv, 2015.

[7] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," arXiv, 2020.