Vector engine 기반 NPU를 위한 grouped pattern-wise pruning

*서수희, 이하윤, 이상호, 신동군
성균관대학교 전자전기컴퓨터공학과
e-mail : *skyb41288@skku.edu, lhy920806@skku.edu, ilena7440@skku.edu, dongkun@skku.edu*

Grouped pattern-wise pruning for vector engine based NPU

* Soohee Seo, Hayun Lee, Sangho Lee, Dongkun Shin
Department of Electrical and Computer Engineering
Sungkyunkwan University

Abstract

Among the recent pruning techniques, it is conducted in various ways such as block-wise and channel-wise, and the accuracy is low because it is coarser than the relatively dense model. In addition, there is a disadvantage that hardware must be designed accordingly. To solve this problem, we propose an optimized pruning technique for VTA accelerator GEMM Core. Accuracy was 5.73% to 23.83% higher than block-wise pruned model. It is expected that there will be scalability that can be applied to various vector engine-base NPUs using this technique.

I. 서론

딥러닝은 머신러닝의 한 분야로, 컴퓨터 비전, 음성 인식, 자연어 처리 등 다양한 분야에서 놀라운 성과를 보여주고 있다. 그러나 딥러닝 모델들은 다층 신경망으로 구성되어 있고, 많은 weight와 매개변수를 포함하고 있어서 대규모의 데이터와 컴퓨팅 자원이 필요하다는 단점이 있다. 최근 CNN 모델들은 많은 파라미터와 복잡한 행렬 연산을 포함하고 있으며, 메모리 요구량과 연산 복잡도가 증가하고 있다. 이를 해결하기 위해 하드웨어적 관점에서는 행렬 연산이나 컨볼루션 연산을 가속화하는 하드웨어에 관한 연구가 진행되고 있다. 가속기 하드웨어 구조는 vector engine기반과 systolic array 기반으로 나뉘어 동작하는데, vector engine 기반 가속기로는 NVDLA[1], VTA[2], MAGNet[3], DNNWeaver[4], MAERI[5] 등이 있다. Systolic array 기반 가속기는 PolySA[6], DNN builder[7] 등이 있다.

본 논문에서는 vector engine 기반 NPU를 위한 pruning 기법을 제안하는데, 그 중 하나인 VTA(Virtual Tensor Accelerator)에 초점을 두어 실험하였다. VTA는 행렬 곱셈, 합성 곱, 텐서 차원 변환 등과 같은 기능을 제공하며, 딥러닝 모델의 수행 시간을 감소시키고 에너지 효율성을 향상시킬 수 있다. 또한, VTA는 TVM을 통해 딥러닝 모델을 VTA 가속기에 배포하고 실행할 수 있으며, 다양한 하드웨어 환경에서의 활용성을 높여 유연성을 확보하고 있다. 이와 더불어 오픈소스 프로젝트에 기반을 두며 개방성과 확장성을 보장하고 있다.

소프트웨어적 관점에서는 딥러닝 모델의 복잡하고 연산 집약적인 작업을 경량화하기 위해 pruning기법이 주목받고 있다. Pruning은 불필요한 weight를

이 논문은 2023년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.IITP-2017-0-00914, 지능형 IoT 장치용 소프트웨어 프레임워크)

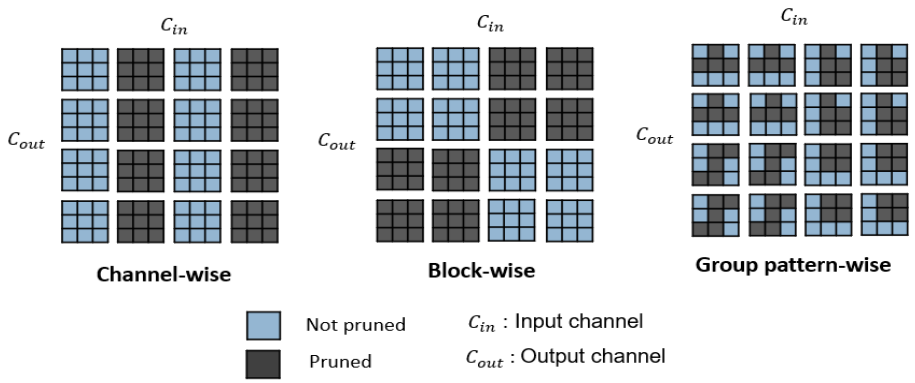

그림 1. 3x3 커널에 대한 Channel-wise, Block-wise 그리고 Group pattern-wise pruning의 비교

제거하여 딥러닝 모델 파라미터 크기와 연산량을 줄이는 방법으로 모델을 효율적으로 압축하는 기법이다.

Pruning은 대표적으로 fine-grained pruning, structured pruning으로 나뉜다. Fine-grained pruning의 경우에는 element-wise로 weight를 제거하는 방식으로, 모델의 weight 중 작은 값이나 중요도가 낮은 값을 0으로 만들어 pruning을 진행한다[8]. 이 경우, 정확도는 유지되지만 추가적인 인덱싱 연산으로 인해 연산 속도를 가속화하기 어렵다. 뿐만 아니라, 희소한 패턴으로 인해 메모리 상에 연속된 데이터를 읽지 못하며 메모리 접근 패턴이 불규칙 해진다. 이에 따라 일부 하드웨어 환경에서는 성능 저하를 가져올 수 있다. 반면, structured pruning은 weight를 더 큰 구조 단위로 제거하는 방식이다[9]. Fine-grained pruning에 비해 모델 압축이나 하드웨어 구현에 있어서 효율적이지만, 모델의 구조를 제한할 수 있으며 중요한 weight를 제거하여 정확도가 fine-grained pruning에 비해 감소한다.

Structured pruning은 대표적으로 channel-wise pruning, block-wise pruning으로 나뉘어진다. Block-wise pruning은 모델의 weight를 블록 단위로 pruning하는 방법으로 모델 크기를 줄이고 성능 유지하는 기법이다[10]. Channel-wise pruning은 모델의 채널 단위로 pruning하는 기법으로, 각 채널에서 중요하지 않은 weight를 제거하여 경량화를 진행한다. Channel-wise pruning은 그림 1과 같이 커널(필터)이 input channel 축과 output channel 축으로 나뉘어져 channel 단위로 weight를 제거된다. Block-wise pruning은 블록 단위로 weight를 제거하여 세분화된 pruning을 통해 channel-wise pruning보다 정확도를 유지할 수 있다.

그러나, 이러한 방식들은 상대적으로 dense한 모델보다 coarse-grained하여 낮은 정확도를 보여준다. 또한, 특정 하드웨어 가속기나 플랫폼에서는 특정한 모델 구조를 지원하지 않을 수 있으며, pruned된 모델을 실행하기 위해서는 추가적인 호환성 작업이 필요하다. 이 문제점들을 해결하기 위해 본 논문에서는 VTA 가속기 GEMM Core에 맞는 최적화된 group pattern-wise pruning 기법을 제안한다. Group pattern-wise pruning은 GEMM core의 최소 연산 단위에 맞춰서 커널(필터)을 그룹화하여 weight를 제거하기 때문에 block-wise pruning보다는 fine-grained하여 정확도를 유지함과 동시에, VTA의 GEMM core 구조를 변경하지 않고 활용할 수 있다. 이는 특정 하드웨어를 설계하지 않아도 pruning된 모델을 가속화할 수 있으며, vector engine 기반 NPU에도 적용시킬 수 있는 확장성을 가졌다.

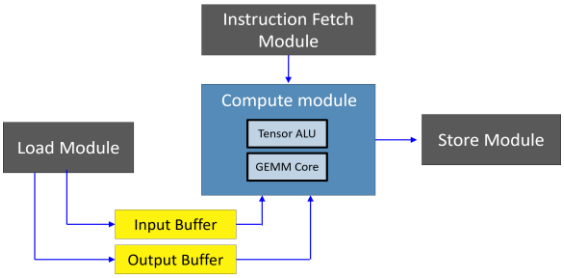

그림 2. VTA 하드웨어 구조도

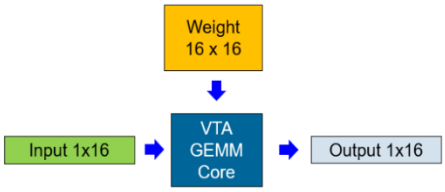

그림 3. VTA GEMM Core 연산 방식

II. 본론

2.1 VTA 가속기

2.1.1. VTA GEMM Core

VTA 하드웨어는 그림 2와 같이 4가지 모듈로 instruction fetch module, load module, compute module, store module로 구성되어 있다. Compute module같은 경우 연산을 담당하며 크게 Tensor ALU, GEMM Core로 구성되어 있다.

GEMM core에서는 그림 3과 같이 input에 대한 연산 단위는 1x16이며 weight에 대한 연산 단위는 16x16이다. 따라서 하나의 1x16의 input에 대해 16x16 weight와 연산을 진행한다. 연산 결과는 output 1x16으로, 본 논문에서는 GEMM Core를 활용하여 컨볼루션 연산을 진행한다.

2.1.2 VTA 컨볼루션 연산

VTA 컨볼루션 연산은 GEMM core에서 진행되며, 그림 4와 같이 하나의 1x16 input이 16개의 커널과 GEMM 연산을 통해 1x16 output을 출력한다.

컨볼루션 연산 시, GEMM 연산으로 가능하도록 input과 커널의 layout을 변경하여 진행한다.

본 논문에서는 VTA GEMM core 연산 방식에 맞게 커널의 1x16 단위로 pruning을 진행하여, block-wise보다는 fine-grained하면서도 VTA 하드웨어 구조 변경없이 정확도를 유지하는 group pattern-wise pruning 기법을 제안한다.

2.2 Group pattern-wise pruning

2.2.1 Weight pruning 단계

Group pattern-wise pruning은 그림 5와 같이 Resnet-20 구조에서 컨볼루션 연산 전 weight pruning 단계를 진행한다. Weight pruning 하기 전 weight 값은 pre-trained된 값을 사용한다.

2.2.3 Group pattern-wise pruning 기법

그림 6은 컨볼루션 커널에서의 group pattern-wise pruning을 나타내고 있다. 그림의 예시로 3x3 커널 16개가 IC (Input channel) 축과 OC (Output channel)축으로 크게 $G_{(0,0,;)}$~$G_{(1,1,;)}$ 4그룹으로 나뉜다. 그리고 그룹 $G_{(IC,OC,;)}$ 내에서 element별 9그룹으로 나뉘게 된다. $G_{(0,0,0)}$~$G_{(0,0,8)}$, $G_{(0,1,9)}$~$G_{(0,1,17)}$, $G_{(1,0,18)}$~$G_{(1,0,26)}$, $G_{(1,1,27)}$~$G_{(1,1,35)}$. 한 그룹 $G_{(IC,OC,;)}$ 당 element별 그룹으로 총 36개의 그룹을 가진다. 각 그룹 $G_{(IC,OC,;)}$ 의 집합 L1 norm을 구해서 top-k 방법으로 sparsity를 정하게 되고 pruning을 진행한다.

C_{in}: Input Channel, C_{out}: Output Channel
IH : Input Height, IW : Input Width,
KH : Kernel Height, KW : Kernel Width,
OH : Output Height, OW : Output Width

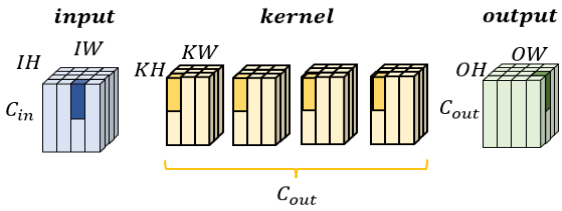

그림 4. VTA 컨볼루션 연산

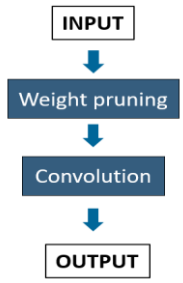

그림 5. Weight pruning 적용한 Resnet-20

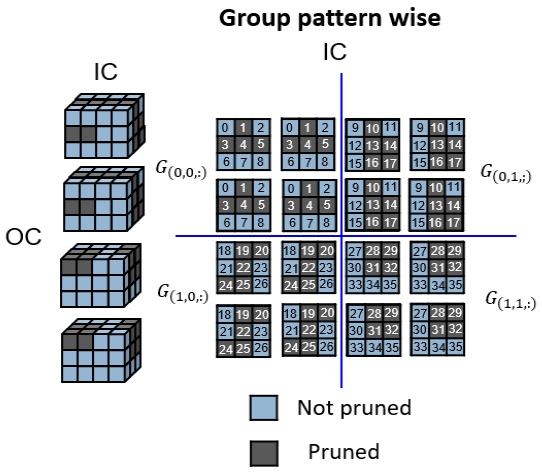

그림 6. 커널의 group pattern-wise pruning

III. 실험 결과 및 분석

3.1 실험 환경

딥러닝 프레임워크는 Pytorch를 사용하였고, pre-

train된 Resnet-20 모델과 CIFAR-10 dataset을 활용하여 group pattern-wise pruning 기법을 구현하였다. (Batch size=128, 200 epoch, 최적화 기법=0.0001의 weight decay & SGD, pre-trained weight 사용)

Pre-train된 Resnet-20의 convolution layer전에 block-wise 단위와 새로 제안하는 group pattern-wise 단위로 weight pruning을 진행하였다. 그리고 sparsity를 50%부터 90%까지 10% 단위로 설정하여 각 패턴들의 정확도를 측정하였다.

3.2 실험 결과

다음 그림 7은 block-wise 단위와 새로 제안하는 group pattern-wise단위로 pruning한 모델의 정확도를 나타내고 있다. Group pattern-wise pruning한 모델이 모든 sparsity에서 block-wise pruning한 모델보다 정확도가 높았으며, 5.73%~23.83% 정도 차이를 보였다. 실험 결과, block-wise pruning보다 group pattern-wise가 더 fine-grained하여 정확도 측면에서 우세한 것을 확인할 수 있다.

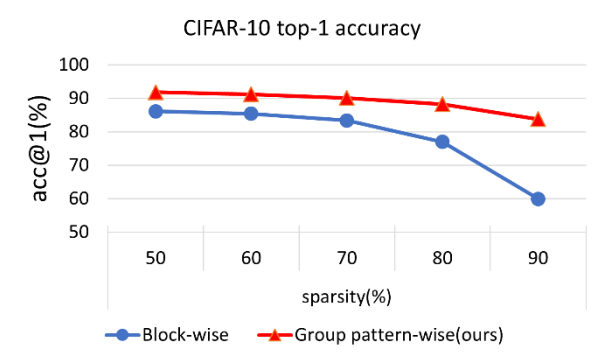

그림 7. Block-wise와 Group pattern-wise pruning sparsity에 따른 정확도

Ⅳ. 결론 및 향후 계획

최근 CNN 모델들은 많은 파라미터와 복잡한 행렬 연산으로 인해 메모리 요구량과 연산 복잡도가 증가하고 있다. 이러한 연산을 처리하기 위해 별도의 NPU 하드웨어가 개발되고 있지만, 특정한 모델에 맞춰 설계해야 한다는 단점이 있다.

본 논문에서는 기존 VTA GEMM core 연산을 기반으로 구조 변경없이 연산량을 줄이는 group pattern-wise pruning 기법을 제시한다. VTA의 기본 1x16 연산 단위로 그룹화하여 block-wise pruning보다는 fine-grained하게 pruning을 진행하여 정확도를 높였다.

실제 VTA 가속기에 group pattern-wise pruning기법을 적용하면 속도가 빠르고 정확도도 유지될 것으로 보인다. 또한, 이 기법을 활용하여 다양한 vector engine 기반 NPU에 적용시킬 수 있는 확장성이 있을 것으로 기대된다.

참고문헌

[1] F. Sijstermans, "The NVIDIA Deep Learning Accelerator," inHot Chips, 2018.

[2] Thierry Moreau, Tianqi Chen, Luis Vega, Jared Roesch, Eddie Yan, Lianmin Zheng, Josh Fromm, Ziheng Jiang, Luis Ceze, Carlos Guestrin, Arvind Krishnamurthy "A Hardware-Software Blueprint for Flexible Deep Learning specialization", IEEE Micro,pp.8-16.

[3] R. Venkatesanet al., "MAGNet: A Modular Accelerator Generator forNeural Networks," in ICCAD, 2019.

[4] H. Sharmaet al., "From High-level Deep Neural Models to FPGAs," in MICRO, 2016.

[5] H. Kwonet al., "MAERI: Enabling Flexible Dataflow Mapping over DNN Accelerators via Programmable Interconnects," in ASPLOS, 2018.

[6] J. Cong et al., "PolySA: polyhedral-based systolic array auto-compilation," in ICCAD, 2018.

[7] X. Zhanget al., "DNNBuilder: An Automated Tool for Building High-performance DNN Hardware Accelerators for FPGAs," in ICCAD, 2018.

[8] Han, Song, et al. "Learning both weights and connections for efficient neural network," Advances in neural information processing systems 28 (2015).

[9] Li, Hao, et al. "Pruning filters for efficient convnets," arXiv preprint arXiv:1608.08710 (2016).

[10] Erich Elsen, Marat Dukhan, Trevor Gale, Karen Simonyan. "Fast Sparse ConvNets," IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 14629-14638,2020.