Single Shot Multibox Detector(SSD)에서의 Tucker Decomposition 효과 분석

한성혜○, 홍경환, 신동군
성균관대학교 소프트웨어대학
hansh0713@gmail.com, redcarrottt@gmail.com, dongkun@skku.edu

Analysis on the Effect of Tucker Decomposition on SSD

Seonghye Han○, Gyeonghwan Hong, Dongkun Shin
College of Software, Sungkyunkwan University

요 약

딥 러닝을 이용한 사물 인식 기술은 다양한 분야에서 사용되고 있다. 최근에는 고성능의 서버 뿐만 아니라 임베디드 장치에서도 딥 러닝을 이용한 사물 인식 기술을 적용하고자 하는 시도가 늘어나고 있다. 하지만 임베디드 장치의 사양을 고려하였을 때 기존 사물 인식 모델을 경량화 하여야 할 필요가 있다. 본 논문에서는 기존 사물 인식 모델 중 하나인 SSD에 Tucker Decomposition을 적용하여 모델을 경량화 하는 방법에 대해 제안한다. SSD의 Layer별 Tucker Decomposition 효과를 분석한 후, Tucker Decomposition을 적용하는 Layer의 개수를 다르게 하여 서로 다른 3가지 방법으로 SSD를 경량화 하였다

1. 서 론

딥 러닝을 이용한 사물 인식 기술은 얼굴 인식, 자율 주행 자동차, 폐쇄 회로 등 다양한 분야에서 사용되고 있다. 최근에는 고성능의 서버 뿐만 아니라 스마트폰, 스마트 워치, 사물 인터넷 기기 등 임베디드 장치에서도 사물 인식 기술을 적용하고자 하는 시도가 늘어나고 있다.

사물 인식 모델 중 하나인 SSD(Single Shot Multibox Detector)는 mAP(mean Average Precision) 74.3%의 정확도를 내고, 서버 상에서는 59 FPS(Frame per Second) 의 빠른 추론 속도를 보여서 실시간 추론이 가능하다. 그러나 컴퓨팅 자원이 부족한 임베디드 장치에서는 실시간 추론이 불가능할 정도로 속도가 줄어들게 된다.

기존에는 딥 러닝 모델을 경량화하는 방법에 대한 연구로, Tensor Decomposition[1], Pruning[2], Quantization[3] 등이 제안된 바가 있다. 그 중 Tensor Decomposition은 큰 Weight Tensor를 작은 크기의 여러 Tensor로 나누어, 하나의 큰 Convolution Layer를 작은 여러 Convolution Layer로 분리하는 경량화 기법이다. Tensor Decomposition을 통해 딥 러닝 모델의 FLOPs(Floating point Operation per Second)와 메모리 사용량을 줄일 수 있어, 컴퓨팅 성능이 모자라는 임베디드 장치에 적합하다. Tensor Decomposition의 대표적인 기법으로는 Tucker[4]와 CP[5]가 있으며, 본 논문에서는 여러 Layer에 동시에 적용 가능한 Tucker Decomposition을 사용하여 SSD를 경량화하였다.

SSD에 Tucker Decomposition을 적용하여 SSD의 특징을 살리고 연산량은 줄이되 정확도를 유지하는 것을 목적으로 하였다. SSD의 모든 Layer에 Tucker Decomposition 적용 시 기존 SSD 대비 mAP가 큰 폭인 16.68% 감소하였다. 이에 따라 SSD의 각 Convolution Layer에 Tucker Decomposition을 적용하며 성능을 측정하여 Tucker Decomposition 적용 시 성능이 좋아지는 Layer들을 모아 여러 조합의 모델을 만들고자 하였다.

임베디드 장치에는 고성능 모바일 CPU와 GPU를 탑재한 스마트폰부터 저전력 목적으로 설계된 스마트 워치, 사물 인터넷 장치까지 다양한 사양의 하드웨어가 있다. 따라서 본 논문에서는 Tucker Decomposition을 적용하는 Layer의 개수를 다르게 하여, 다양한 사양의 임베디드 장치에서 사용할 수 있도록 SSD를 세 가지 방법으로 경량화하였다. 방법 1의 경우 FLOPs 64.38%감소 대비 mAP 16.68%감소, 방법 2의 경우 FLOPs 47.81%감소 대비 mAP 5.81%감소, 방법 3의 경우 FLOPs 25.35%감소 대비 mAP 4.47%감소의 성능을 보였다.

본 논문의 2장에서는 SSD, Tucker Decomposition의 배경 지식을 설명하고, 3장에서는 연구내용, 4장에서는 실험 환경, 실험 결과 및 분석 등 실험 전반에 대해 설명한다. 마지막으로 5장에서는 결론 및 향후 연구에 대해 설명한다.

2. 배 경

2.1. SSD

SSD[6]는 사물 인식을 위한 딥 러닝 모델 중 하나로써 단일 신경망으로 이루어져 있다. 특징으로는 default box 개념을 사용하는 것과 여러 개의 feature map을 사용하여 정확도를 높인다는 점이 있다. SSD의 신경망은 VGG-16의 Pool_5까지 Layer와 그 후 6개의 보조 Layer의 구조로 이루어져 있다. 이 후 검출과정과 여러 검출결과 중 정확한 결과만 걸러내는 NMS(Non-Maximum Suppression)의 과정

"본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 SW컴퓨팅산업원천기술개발사업(SW스타랩)의 연구결과로 수행되었음" (IITP-2017-0-00914)

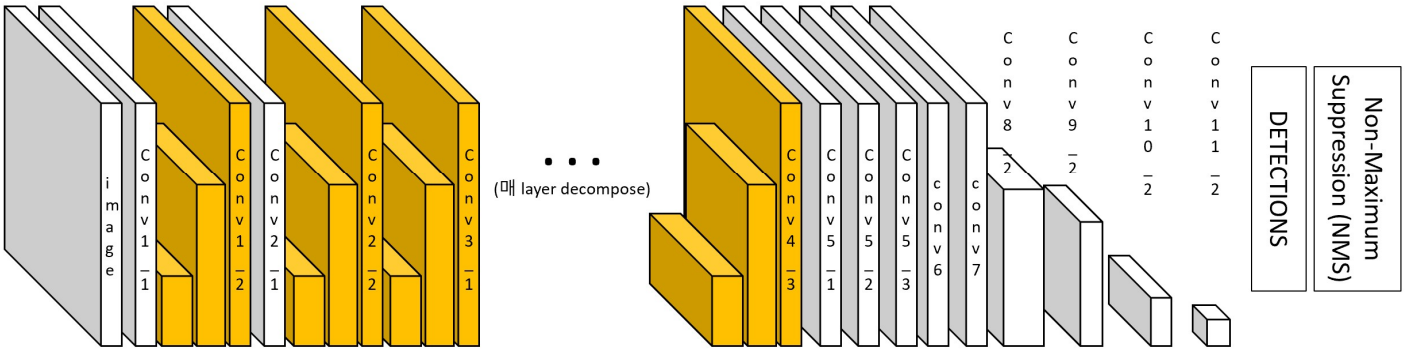

그림 1 기존 SSD 대비 FLOPs 감소율 / mAP 감소율(BCR) > 1인 Layer를 Decompose한 SSD 구조

을 거친다.

다른 사물 인식 모델인 Faster R-CNN이 7FPS의 속도를 내는 반면, SSD는 46FPS로 속도가 향상되면서도 mAP는 73.2%에서 74.3%로 소폭 향상되었다. YOLO는 SSD와 45FPS로 비슷한 속도를 내지만, YOLO는 mAP가 63.4%로 더 저하되었다.

2.2. Tucker Decomposition

Tucker Decomposition이란 하나의 고차원 Tensor를 작은 고차원 중심 Tensor 하나와 여러 개의 저차원 보조 Tensor로 분리하는 기법이다. 딥 러닝 모델에서 Tucker Decomposition을 사용할 경우 하나의 Convolution Layer가 3개의 작은 Layer로 분리된다. 또한 여러 Convolution Layer에 동시에 적용할 수 있다. 딥 러닝 모델에 Decomposition 적용 후에는 Weight의 정보가 손실되면서 정확도가 떨어지기 때문에, 모델의 재학습을 통해 정확도를 회복해 주어야 한다.

3. 선택적 Tucker Decomposition 적용

본 연구에서는 다양한 조합의 Layer에 Tucker Decomposition을 적용하여 SSD를 경량화하고자 하였다. Tucker Decomposition 적용 시 FLOPs가 감소하는 것이 Benefit, mAP가 감소하는 것이 Cost인 점을 기반으로 하여 기존 SSD 대비 FLOPs 감소율 / mAP 감소율을 BCR(Benefit-Cost Ratio)로 설정하였다. 즉, FLOPs의 감소가 크고, mAP의 감소가 작을수록, 즉 BCR이 더 클수록 Decomposition 적용 효과가 크다고 할 수 있다.

첫 번째로 SSD의 모든 Layer에 각각 Tucker Decomposition을 적용하여 경량화하였다. 모든 Layer에 Tucker를 적용한 모델은 연산량이 크게 감소하지만 정확도도 크게 감소한다.

두 번째로 SSD의 각 Layer에 Tucker Decomposition을 적용하며 Decomposition 적용 효과를 측정하였다. 또한 BCR을 바탕으로 Decomposition 적용 시 유의미한 효과를 보이는 Layer들을 분류하였다.

세 번째로 Decomposition 적용 시 유의미한 효과를 보이는 Layer들에 선택적으로 Tucker Decomposition을 적용하여 경량화하였다. BCR이 1 이상인 Layer를 Decompose (그림 1), BCR이 2 이상인 Layer를 Decompose의 두 가지 방법으로 기존 SSD를 경량화하였다.

SSD의 모든 Layer를 Decompose한 경우(방법 1), BCR이 1 이상인 Layer를 Decompose한 경우(방법 2), BCR이 2 이상인 Layer를 Decompose한 경우(방법 3)의 총 세 가지 방법으로 SSD를 경량화하였다. 방법 1에서 3으로 갈수록 Decomposition을 적용하는 Layer의 수가 줄어들어 경량화가 덜 진행된다. 이에 따라 모델 연산량이 늘어나고 모델 정확도는 높아진다. 이렇게 만들어진 세 가지 방법은 다양한 사양의 하드웨어를 갖는 임베디드 장치의 특성을 고려하여 장치의 사양과 필요한 정확도에 따라 골라 사용할 수 있게 하였다.

4. 실 험

4.1. 실험 환경

실험은 pytorch를 이용하여 Nvidia Titan V GPU 상에서 진행하였다. Dataset은 VOC2007, VOC 2012를 사용하여 학습하였으며 VOC2007을 사용하여 추론하였다. Decomposition 후에는 모델의 5,000회 재학습을 진행하여 정확도를 회복하였다.

4.2. 모든 Layer에 Decomposition 적용

SSD에 존재하는 19개의 Convolution Layer에 동시에 Tucker Decomposition을 적용하여 만든 새로운 모델의 성능을 측정하였다. 추론 성능 측정 결과는 그림 2에서 볼 수 있듯이 FLOPs 11.2GMACs, mAP는 64.71%로 기존 SSD 대비 FLOPs는 64.38% 감소하였고 mAP 또한 16.68% 감소하였다. 연산량이 절반 이하로 크게 줄어들었지만 정확도 또한 큰 폭으로 감소하였다.

4.3. 하나의 Layer에 Decomposition 적용

SSD에 존재하는 Convolution Layer 중 하나에만 Tucker Decomposition을 적용하여 성능을 측정하였다. 추론 성능 측정 결과는 그림 2와 같다. 하나의 Layer에만 Decomposition을 적용해서는 기존 SSD의 성능과 큰 차이가 없다.

기존 SSD 대비 FLOPs 감소율 / mAP 감소율을 BCR로 각 Layer의 Decomposition 적용 효과를 측정하였다. BCR이 1 이상인 Layer는 Conv1_2, Conv2_2, Conv3_2, Conv3_3, Conv4_1, Conv4_2, Conv4_3의 7개 layer이고, BCR이 2 이상인 Layer는 Conv3_3, Conv4_2, Conv4_3의 3개 layer이다.

4.4. 여러 Layer에 Decomposition 적용

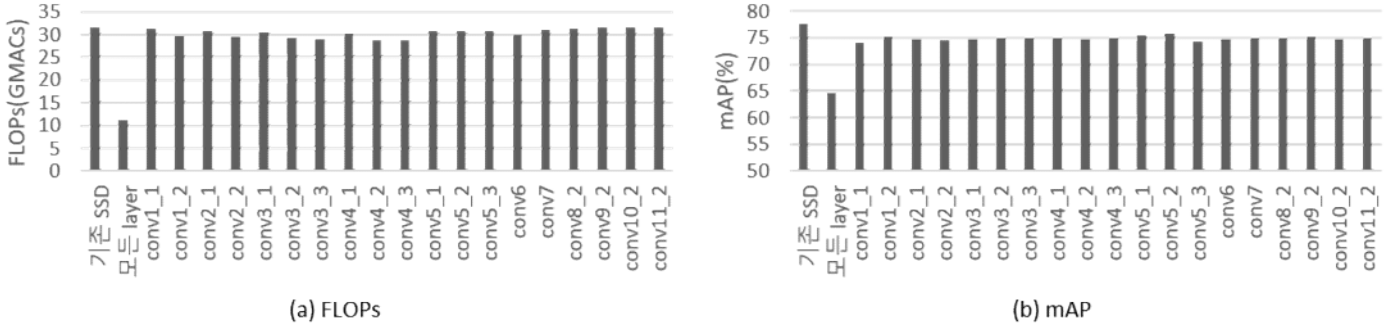

그림 2 하나의 Layer에 Tucker Decomposition 적용 성능

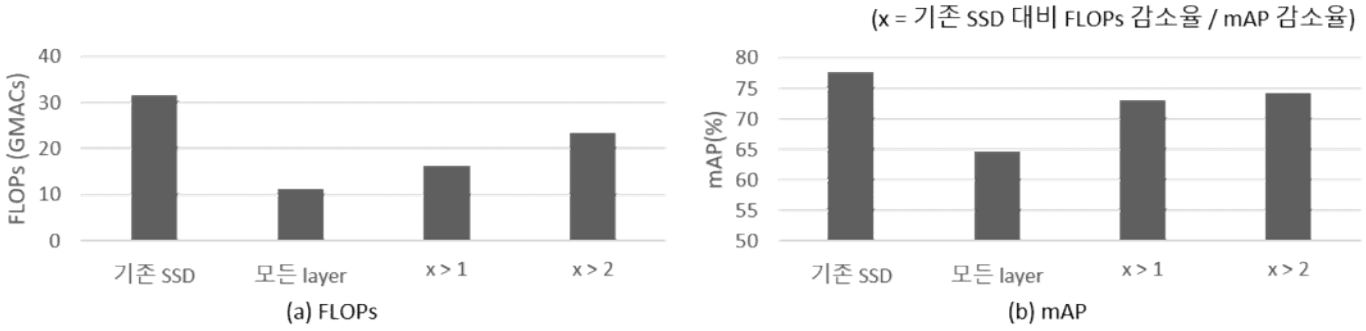

그림 3 기존 SSD와 경량화된 모델의 성능 비교

SSD모델에 존재하는 Convolution Layer 중 Decomposition 적용 시 유의미한 효과를 보이는 Layer들에 선택적으로 Tucker Decomposition을 적용하였다. BCR이 1 이상인 Layer를 Decompose한 경우, BCR이 2 이상인 Layer를 Decompose한 경우의 두 가지 방법으로 기존 SSD를 경량화한 모델의 성능을 측정하였다. 추론 성능 측정 결과는 그림 3과 같다. BCR이 1 이상 Layer Decompose한 경우 기존 SSD모델 대비 FLOPs 47.8%감소, mAP 5.8%감소, BCR이 2 이상인 경우 FLOPs 25.35%감소, mAP 4.47%감소의 결과를 내었다.

5. 결 론

본 논문에서는 딥 러닝을 사용한 사물 인식 모델의 하나인 SSD의 경량화를 위해 Tucker Decomposition을 사용하는 것을 제안하였다. 다양한 사양의 임베디드 장치에서 사용할 수 있도록 SSD를 세 가지 방법으로 경량화하여 FLOPs 64.38%감소 대비 mAP 16.68%감소, FLOPs 47.81% 감소 대비 mAP 5.81%감소, FLOPs 25.35%감소 대비 mAP 4.47%감소의 성능을 갖는 세 가지 새로운 모델을 만들었다. 또한 이 모델들은 사용하는 임베디드 장치의 성능에 맞추어 골라 사용할 수 있게 하였다.

추후 본 논문에 이어 SSD 뿐만 아니라 YOLO, SqueezeDet 등의 다양한 사물 인식 모델에 Tucker Decomposition을 적용한 성능 향상을 연구를 제안한다.

참 고 문 헌

[1] Kolda, Tamara G and Bader, BrettW. Tensor decompositions and applications. SIAM review, 51(3), p. 455-500, 2009.

[2] Han, Song, Mao, Huizi, and Dally,William J. A deep neural network compression pipeline: Pruning, quantization, huffman encoding. arXiv preprint arXiv:1510.00149, 10, 2015.

[3] Gong, Yunchao, Liu, Liu, Yang, Ming, and Bourdev, Lubomir. Compressing deep convolutional networks using vector quantization. arXiv preprint arXiv: 1412.6115, 2014.

[4] Tucker, Ledyard R. Some mathematical notes on three-mode factor analysis. Psychometrika, 31(3), p. 279-311, 1966.

[5] Carroll, J Douglas and Chang, Jih-Jie. Analysis of individual differences in multidimensional scaling via an n-way generalization of eckart-young decomposition. Psychometrika, 35(3), p. 283-319, 1970.

[6] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. SSD: Single shot multibox detector. In European conference on computer vision. Springer, Cham, p. 21-37. 2016.

[7] Jacob Gildenblat. pytorch-tensor-decompositions URL: https://github.com/jacobgil/pytorch-tensor-decompositions.

[8] Alexander Max deGroot. ssd.pytorch URL: https://github.com/amdegroot/ssd.pytorch.