Journal of KIIT. Vol. 16, No. 10, pp. 1-9, Oct. 31, 2018. pISSN 1598-8619, eISSN 2093-7571 1
http://dx.doi.org/10.14801/jkiit.2018.16.10.1

모바일 시스템을 위한 CNN 딥 러닝 가속화 알고리즘

박성우*[1], 한경호*[2], 장우영**

CNN Deep Learning Acceleration Algorithm for Mobile System

Sungwoo Park*[1], Kyong-Ho Han*[2], and Wooyoung Jang**

이 논문은 2017년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (NRF-2017R1D1A1B03036353).

요 약

컴퓨팅 및 저장 용량이 제한된 모바일 시스템은 딥 러닝(Deep Learning) 학습과 추론을 데이터 센터에서 주로 처리한다. 따라서 모바일 시스템은 개인을 위한 특별한 인공지능 서비스를 제공하기 어렵고, 사용자들은 개인 정보를 데이터 센터로 전송하는 것을 꺼려할 수 있다. 따라서, 본 논문에서는 모바일 시스템에서 딥 러닝의 학습(Train)과 추론(Inferencd)를 할 수 있는 컨볼루션 신경망(Convolutional Neural Network) 기반 딥 러닝 가속 알고리즘을 제안한다. 제안 알고리즘은 컨볼루션 신경망에서 낮은 랭크 근사(Low-Rank Approximation)로 신경망 정보를 소수의 가중치로 집중시키고, 중요하지 않은 가중치를 제거하는 가지치기(Pruning) 기법으로 컨볼루션 신경망의 크기를 효율적으로 줄인다. 실험을 통해 제안된 알고리즘은 기존의 가지치기 알고리즘보다 추론의 속도를 1.65배 빠르며, 재학습 횟수를 1.5배 줄이고, 가중치들 저장을 위한 메모리 용량을 2배 줄일 수 있다.

Abstract

A mobile system with limited computing and storage capacity mainly processes the training and inference of deep learning in a data center. Therefore, it is difficult for a mobile system to provide private artificial intelligence services, and users may be reluctant to transfer personal information to data centers. Therefore, this paper proposes a deep learning acceleration algorithm for convolutional neural network where a mobile system enables learning and inference itself. The proposed algorithm efficiently reduces the size of the convolutional neural network by a low-rank approximation method that compacts the information of the neural network into some weights, and a pruning method that removes non-critical weights. Experimental results show that the proposed algorithm achieves the speed of inference 1.65 times faster, requires the number of fine-tune fewer 1.5 times, and reduces the memory capacity for storing weights 2 times less than the conventional prunning algorithm.

Keywords
deep learning, convolutional neural network, pruning, low-rank approximation

* 단국대학교 전자전기공학과
- ORCID[1]: http://orcid.org/0000-0003-4561-8560
- ORCID[2]: http://orcid.org/0000-0002-0260-2107
** 단국대학교 전자전기공학과 교수(교신저자)
- ORCID: http://orcid.org/0000-0002-7262-0028

· Received: Jul. 24, 2018, Revised: Aug. 22, 2018, Accepted: Aug. 25, 2018
· Corresponding Author: Wooyoung Jang
Dept. of Electronics and Electrical Engineering, Dankook University
Yongin-si, Geonggi-do, Korea,
Tel.: +82-31-8005-3620, Email: wyjang@dankook.ac.kr

Ⅰ. 서 론

계속되는 반도체 공정의 미세화와 급격하게 증가하는 데이터로 인공지능이 최근에 많은 주목을 받고 있다[1]. 그 중에서도 컨볼루션 신경망(CNN, Convolutional Neural Network)을 기반으로 하는 딥 러닝(Deep Learning)은 컴퓨터 비전[2][3], 자연어 처리[4], 음성인식[5] 등 다양한 영역에서 발전하고 있다. 컨볼루션 신경망은 1998년 손 글씨를 인지하는 LeNet[6]에 의해 소개되었고, 최근에 손 글씨 인지의 정확도를 향상시키기 위하여 신경망의 복잡도와 가중치 행렬의 계층이 증가되고 있다. 예를 들어, ILSVRC(ImageNet Large Scale Visual Recognition Challenge)에서 우승한 Alexnet[7]과 VGG[8]는 각각 61M개, 138M개의 가중치가 존재한다. 또한, 그 가중치를 학습시키기 위한 많은 데이터 세트와 연산을 필요로 한다. 이러한 이유로 컨볼루션 신경망 기반 딥 러닝을 이용한 인공지능 서비스는 고성능의 컴퓨터 시스템과 대용량 메모리를 요구하고 있다[9][10].

컴퓨팅 및 저장 용량이 부족한 모바일 시스템은 딥 러닝 학습(Train)과 추론(Inference)을 데이터 센터에서 주로 처리한다. 그러한 방식으로는 개인을 위한 특별한 인공지능 서비스를 제공하기 어렵고, 사용자는 개인 정보를 데이터 센터로 전송하는 것을 꺼려할 수 있다. 따라서 모바일 시스템이 인공지능 서비스를 직접 지원하는 것을 하나의 해결책으로 주목받고 있다. 하지만, 계속되는 반도체 집적도 향상에도 불구하고, 최신의 컨볼루션 신경망에 필요한 프로세서와 메모리를 모바일 시스템에 구비하기는 것은 현실적으로 어렵다[11]-[13].

이를 해결하는 방법으로 가중치 가지치기(Pruning) 기법이 있다. 그 기법은 작은 메모리에 컨볼루션 신경망을 저장하기 위하여, 가중치의 개수를 줄이고, 가중치들의 연결을 단순화한다. 그러한 가지치기 기법들로 [14]에서는 ConvNet(Convolution Network)의 크기를 2배 줄이고, [15][16]은 Alexnet의 크기를 9배, VGG-16의 크기를 13배 줄였다. 하지만, 복잡한 신경망에 가지치기를 적용하면, 정확도 손실이 많이 발생하기 때문에, [15][16]에서는 정확도를 다시 올리기 위해 많은 시간동안 재학습(Fine-Tune)을 진행한다. 다른 방법으로는 낮은 차수 근사(Low-Rank Approximation)가 있다. 낮은 차수 근사는 가중치로 구성된 행렬의 크기를 줄인다. 낮은 차수 근사는 원래 가중치 행렬의 특성을 유지하기 때문에 가중치 가지치기를 적용했을 때보다 정확도 하락이 적다. 또한, 행렬의 크기가 작아지기 때문에 딥 러닝 학습과 추론의 연산 속도가 향상된다.

본 논문에서는 모바일 시스템을 위한 컨볼루션 신경망 가속화 알고리즘을 제안한다. 우선 낮은 차수 근사를 통해 신경망의 모든 계층의 행렬의 크기를 줄인다. 다음으로 각 계층에서 가중치의 희소성에 따라 가중치 가지치기를 적용할 계층을 선택한다. 선택된 계층에서 기준값(Threshold)이하의 가중치들을 제거한다. 이를 통해 연산 및 저장 용량이 제한된 모바일 시스템에 적합한 컨볼루션 신경망을 제공할 수 있다.

본 논문의 다음과 같이 구성되어있다. 2장에서는 컨볼루션 신경망와 기본 구조와 동작 그리고 기존의 관련 연구에 대해 논의한다. 3장에서는 낮은 차수 근사와 가중치 가지치기를 적용한 가속화 알고리즘을 제안하고 설명한다. 4장에서는 일반적인 가지치기 알고리즘과 제안한 가속화 알고리즘을 비교분석한다. 5장에서는 본 논문의 결론을 내린다.

Ⅱ. CNN의 기본 동작과 관련 연구

2.1 CNN의 기본 동작

그림 1은 컨볼루션(Convolution) 신경망의 기본 구조이다.

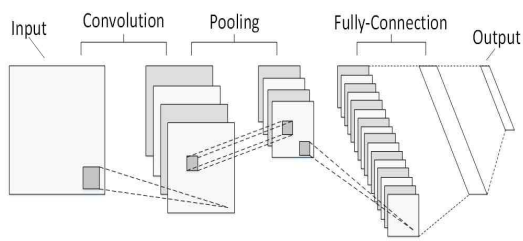


그림 1. CNN의 구조.
Fig. 1. Convolutional neural networks structure

컨볼루션 신경망은 입력의 특징을 추출하는 컨볼루션 계층, 추출된 특징으로 결과를 추론하는 완전결합(Fully Connected) 계층, 컨볼루션 계층의 출력(Feature Map)의 크기를 줄여주는 풀링(Pooling) 계층으로 구성된다. 컨볼루션 계층에서는 그림 2(a)와 같이, 가중치 필터 행렬과 입력 행렬이 컨볼루션 연산이 된다. 즉, 그 필터 행렬과 상응하는 입력을 곱한 후, 그 값을 더한 결과(Partial Sum)를 출력 행렬에 순차적으로 저장한다. 이때, 컨볼루션 출력 행렬의 크기는 다음과 같이 정의 된다.

$$h_o = (h_i - h_f + 2p)/s + 1 \quad (1)$$

$$w_o = (w_i - w_f + 2p)/s + 1 \quad (2)$$

식 (1)에서 h_o, h_i, h_f은 각각 출력 행렬의 높이, 입력 행렬의 높이, 가중치 필터 행렬의 높이를 의미한다. p는 제로 패딩(Zero Padding)을 의미하고, s는 필터의 움직임을 의미한다. 식 (2)에서 w_o, w_i, w_f는 각각 출력 행렬의 넓이, 입력 행렬의 넓이, 가중치 필터 행렬의 넓이를 의미한다. 컨볼루션 계층은 반복적인 곱셈과 덧셈 연산 때문에, 컨볼루션 신경망의 연산시간의 대부분은 컨볼루션 계층에서 소모된다.

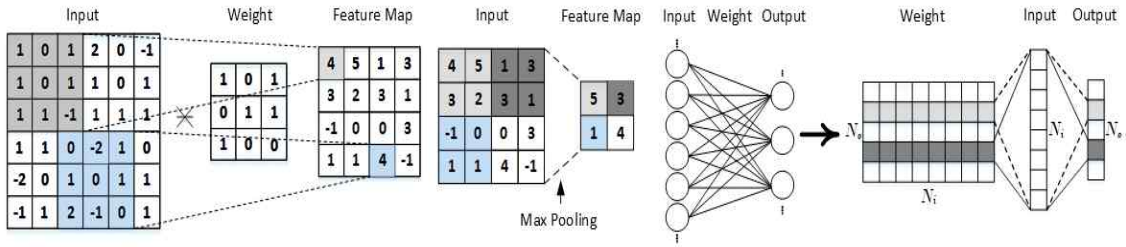
풀링 계층은 컨볼루션 계층에서 생성된 출력 행렬의 크기를 줄인다. 또한, 줄어든 출력 행렬은 오버피팅(Over-Fitting) 문제를 줄여줄 수 있다. 출력 행렬을 줄이는 방법은 그림 2(b)와 같다. 크기가 2×2 혹은 3×3인 가중치 필터가 2 또는 3 만큼 이동하면서 가중치 필터 행렬과 교차하는 입력 행렬의 값 중에서 가장 큰 값을 출력 행렬에 저장한다.

완전연결 계층은 신경망의 마지막 부분에 존재하며 컨볼루션 계층을 통해 얻은 특징들을 종합하여 결과를 추론한다. 완전 연결 계층의 구조는 그림 2의 (c)와 같다. 가중치 필터 행렬의 크기는 입력과 출력 행렬의 크기에 의해 정해진다. 출력의 개수가 N_o, 입력의 개수가 N_i이라면 가중치 행렬의 크기는 $N_o \times N_i$으로 정의될 수 있다. 그리고 완전 연결 계층 연산 방법은 입력 행렬인 $1 \times N_i$과 가중치 필터 행렬의 n 번째 행의 값을 곱하고, 곱한 모든 값을 더하여 출력의 n 번째 값이 된다.

2.2 관련연구

사진을 판별하기 위한 신경망인 AlexNet[7], VGG[8], GoogleNet[17] 등은 많은 계층의 가중치 필터 행렬을 저장하기 위한 대용량의 메모리와 반복적인 곱셈과 덧셈 연산을 위해 고성능 컴퓨터 시스템이 요구된다. 따라서 대부분의 컨볼루션 신경망은 대용량 메모리를 내재하고 멀티-코어로 구성된 GPU(Graphics Processing Unit)에서 주로 구현된다.

다양한 프로세서에서도 컨볼루션 신경망을 활용하기 위한 연구가 진행되고 있다[18]. 메모리 용량이 부족한 MCU(Micro Controller Unit) 또는 CPU (Central Processing Unit)에서 학습이 끝난 가중치 필터 행렬에 낮은 차수 근사로 가중치 필터 행렬의 크기를 줄인다[19]. 또한, 가지치기[15][16]를 통하여 불필요한 가중치와 연결을 줄인다.



(a) 컨볼로션 레이어 (b) 풀링 레이어 (c) 완전 결합 레이어.

그림 2. 계층 연산과정

Fig. 2. Layer operation, (a) Convolution layer, (b) Pooling layer, (c) Fully-connected layer

대부분의 가지치기 기법은 완전 연결 계층에 주로 적용되고[15][16], 최근 연구에서는 컨볼루션 계층에도 적용하고 있다[14][18]. 컨볼루션 계층에 가지치기를 하면, 신경망의 정확도가 크게 낮아진다.

III. 모바일 시스템을 위한 CNN 가속화 알고리즘

Caffe[20]에서 AlexNet에 딥 러닝 학습을 적용하면 Top-1 정확도는 57.4%이다. Top-1 정확도는 추론으로 입력에 대해 정확한 출력이 나올 확률이다. 다음으로, [15]과 같이 가지치기를 하면, 51.02%까지 Top-1 정확도가 낮아진다. 가지치기된 Alexnet에서는 입력 영상의 특징을 찾는 첫 번째 컨볼루션 계층의 가중치 필터 행렬에도 가지치기를 적용하기 때문에, 전체 컨볼루션 신경망의 정확도가 낮아진다. 손실된 Top-1 정확도를 가지치기 기법을 적용하기 전까지 회복하기 위하여 94,000회 재학습이 수행되어야 한다. 재학습은 마스킹(Masking) 계층을 추가하여 진행이 되기 때문에, 1회 학습에 걸리는 시간이 일반적인 학습 시간보다 길다[15]. 이러한 문제를 해결하기 위하여, 본 논문에서는 컨볼루션 계층에 낮은 차수 근사를 먼저 적용하여 다수의 가중치들을 소수의 가중치로 값을 집약한 후에, 가지치기를 적용한다. 그 결과로 신경망의 크기가 작아져서 가중치의 저장과 연산을 줄이고, 정확도 손실도 최소화된다.

3.1 가속화 알고리즘

알고리즘 1은 제안하는 가속화 알고리즘을 보여준다. 먼저, 학습이 완료된 신경망의 모든 계층에 낮은 차수 근사를 적용한다. 완전 결합 계층은 가중치의 절대값이 0에 가까운 값부터 전체 가중치의 90%를 가지치기를 적용한다[15]. 다음으로, 각 계층들의 희소성을 조사한다. 기준값은 첫 번째 컨볼루션 계층에서 가중치의 절대값의 크기가 하위 x%가 되는 값이 된다. x는 모바일 시스템의 메모리 크기에 따라 초기값이 변경될 수 있다. 결정된 기준값 이하의 값은 0으로 간주되고, 희소성은 계층의 가중치 중 0이 차지하는 비율을 의미한다. 기준값에 의해 조사된 각 계층의 희소성을 기준으로 가지치기를 선택된 컨볼루션 계층부터 가지치기를 적용한다. 다음으로, 낮은 차수 근사와 가지치기 기법들에 의한 Top-1 정확도 손실을 복원하기 위해 재학습을 진행한다.

Algorithm 1. CNN Acceleration

1: generate l^r_0, l^r_1,...., $l^r_{n-1} \in L^r$ where trained layers l^t_0, l^t_1,...., $l^t_{n-1} \in L^t$ are low-rank approximated;
2: **for** non-convolution layer in L^r **do**
3: generate l^p_0, l^p_1,...., $l^p_{n-1} \in L^p$ where 90% weights close to zero are prunned in each layer [15];
4 **end for**
5 compute threshold t from the sparsity x% of the first low-rank approximated layer l^r_0;
6 select layer l^r_s which starts to be prunned;
7: **for** convolution layer from l^r_s to l^r_{n-1} in low-rank approximated L^r **do**
8: generate L^p where weights more than t are prunned;
9: **end for**
10: **for** Top-1 accuracy(L^t) - Top-1 accuracy(L^f) is greater than error e **do**
11: generate l^f_0, l^f_1,...., $l^f_{n-1} \in L^f$ where L^p is fine-tuned;
12: **end for**

3.2 낮은 차수 근사

낮은 차수 근사는 Top-1 정확도의 큰 손실 없이 신경망의 크기를 줄여 연산 속도를 향상 시키는 방법이다. 컨볼루션 계층의 가중치 필터 행렬은 작지만, 다수의 가중치 필터 행렬이 존재한다. 따라서 많은 연산이 다층의 컨볼루션 계층에서 발생한다. 완전 연결 계층은 가중치가 집중되어 있어 신경망의 전체적인 크기에 영향을 준다. 이러한 각 계층들에 낮은 차수 근사 기법을 사용하여 신경망의 크기를 줄일 수 있다. 낮은 차수 근사는 특이값 분해(Singular Value Decomposition)를 통해 행렬을 분해하고 본 행렬과 차이가 적게 나는 작은 차수(Rank)를 구하여 행렬의 크기를 줄인다[21].

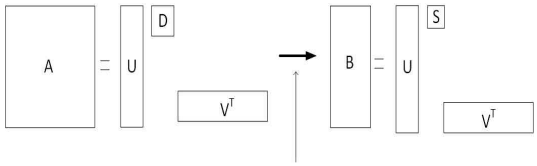


그림 3. 낮은 차수 근사 기법 [23]
Fig. 3. Low-rank approximation method [23]

그림 3은 $m \times n$ 행렬 A의 낮은 차수 근사 기법을 보여준다. 먼저, 특이값 분해를 통해 $A = UDV^T$로 만든다. 여기서, 행렬 D는 행렬 A의 대각 행렬이고, $D = AA^T$는 $m \times n$ 행렬이고, U와 V 행렬은 각각 $m \times m$, $n \times n$인 직교 행렬이 된다. 여기서 대각 행렬 D의 요소들이 행렬 A의 특이값(Singular Value)이 된다. 행렬 U의 열벡터들을 행렬 A의 왼쪽 고유벡터, 행렬 V의 열벡터들을 오른쪽 고유벡터가 된다. 그리고 새로운 행렬 $B = USV^T$ 가 정의되고, 식 (3)에 보듯이, 행렬 A와 오류가 가장 적은 행렬 B를 구해야 한다. 여기서, 행렬 S는 $r \times r$이 되며, r을 행렬 A의 차수라고 한다. r은 식 (4)을 통해 정해진다[22][23].

$$\min_B \|A - B\|_F \tag{3}$$

$$r \leq \frac{mn}{m+n} \tag{4}$$

작아진 행렬 B는 행렬 A의 특이값과 고유 벡터를 갖고 있기 때문에, 최소의 Top-1 정확도 하락으로 컨볼루션 신경망 가중치 개수를 줄인다.

3.3 가지치기 기법

가지치기 기법은 학습이 끝난 컨볼루션 신경망에서 중요하지 않은 가중치를 제거하는 방법이다[15][16][24]. 가지치기 기법에서는 각 계층에서 남겨놓을 가중치 비율이 결정되면, 0에 가장 근접한 가중치부터 제거된다. 각 계층이 Top-1 정확도에 미치는 영향이 다르므로, 동일한 가중치 비율로 계층들이 가지치기 기법이 적용된다면, Top-1 정확도가 크게 낮아질 수 있다. 따라서 제안된 알고리즘은 낮은 차수 근사 후에 각 계층마다 Top-1 정확도에 영향이 작아지도록 다른 비율로 가중치 가지치기를 한다.

가지치기를 적용하기 위한 기준값은 첫 번째 컨볼루션 계층의 가중치 희소성이 10%가 되는 값으로 설정한다. 첫 번째 컨볼루션 계층의 희소성을 바탕으로 기준값을 설정한 이유는 첫 번째 컨볼루션 계층에서는 입력 영상의 경계(Edge)를 인지하기 때문에, 컨볼루션 계층 중에서 가중치 희소성이 가장 낮다. 계속되는 상위 컨볼루션 계층을 통해 연산이 진행되면, 경계 부분의 가중치들이 소수의 가중치들로 더 집중된다.

그림 4는 첫 번째 컨볼루션 계층의 희소성을 10%와 20%가 되는 기준값에 따른 다른 컨볼루션 계층의 희소성을 보여준다. 첫 번째 컨볼루션 계층의 가중치 희소성이 20%가 되는 기준값의 경우[25], 완전 결합 계층에서는 대부분의 가중치를 0에 근접한 값으로 인지하기 때문에 가지치기 기법을 적용하면 Top-1 정확도가 크게 하락한다. 반복적인 실험의 결과로 첫 번째 컨볼루션 계층의 희소성이 10% 이내가 되는 기준값으로 설정한다.

다음으로, 모바일 시스템의 메모리 크기에 따라 가지치기를 수행할 컨볼루션 계층을 선택한다. 그 메모리의 크기가 작다면, 낮은 컨볼루션 계층부터 가지치기를 수행하는 것이 좋다. 하지만, Top-1 정확도는 낮아지고, 재학습의 시간이 늘어날 것이다. 반대로, 그 메모리 크기가 크다면, 높은 컨볼루션 계층부터 가지치기를 수행하는 것이 좋다. 하지만, Top-1 정확도 높아지고, 재학습의 시간은 짧아질 것이다. 가지치기는 첫 번째 계층에서 결정된 기준값에 따라 수행된다.

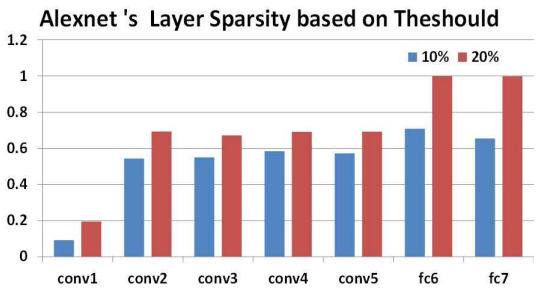


그림 4. Alexnet의 가중치 희소성
Fig. 4. Weights sparsity of Alexnet

그림 4에서 보듯이, 각 계층에서 기준값보다 작은 가중치들의 개수가 다르다. 만약 각 컨볼루션 계층을 동일한 비율로 가지치기 적용한다면, 상대적으로 중요한 가중치들이 제거될 수 있다. 제안된 알고리즘에서는 동일한 기준값을 모든 계층의 가중치들에 공정하게 적용하여 가지치기가 수행하였다.

IV. 실험 결과

4.1 실험 환경

본 논문에서는 컨볼루션 신경망 기반 딥 러닝 가속화 알고리즘을 구현하기 위한 프레임워크로 Caffe를 선택하였고, 제안된 알고리즘을 적용할 컨볼루션 신경망으로 Alexnet 모델을 사용하였다. 영상 추론에 최적화된 딥 러닝 프레임워크인 Caffe는 C++로 신경망의 각 부분이 모듈화 되어있고, 이를 Shell Script, Python, MATLAB 등을 사용하여 신경망의 학습과 추론을 한다. 또한, CPU와 더불어 Nvidia에서 제공하는 CUDA(Compute Unified Device Architecture) 라이브러리를 사용하여 GPU에서도 실험이 가능하다. Alexnet은 57.4%의 Top-1 정확도로 ILSVRC-2012에서 우승한 컨볼루션 신경망으로서, 5개의 컨볼루션 계층, 3개의 풀링 계층, 그리고 2개의 완전 결합 계층으로 구성되어 있다. 본 논문에서는 Caffe에서 미리 학습된 Alexnet을 사용하여 실험을 진행하였다.

제안된 가속화 알고리즘은 원본 Alexnet(oAlexnet)과 가지치기된 AlexNet(pAlexnet)와 비교된다. pAlexNet은 첫 번째 컨볼루션 계층을 38% 나머지 컨볼루션 계층들은 동일하게 65% 가지치기를 적용하고 완전 결합 계층은 90% 가지치기를 적용한다[15]. 제안된 Alexnet(fAlexent) 2, 3, 4는 모든 계층에 낮은 차수 근사가 먼저 적용되고, 각각 컨볼루션 2, 3, 4 계층부터 가중치 가지치기를 적용한 것을 의미한다. 예를들어, fAlexnet 3는 컨볼루션 계층 1과 2는 낮은 차수 근사만 진행하고, 컨볼루션 계층 3부터 낮은 차수 근사 후 가지치기를 수행한다. 또한, 완전 결합 계층은 낮은 차수 근사 기법을 적용한 후에 pAlexnet과 동일하게 90% 가중치 가지치기를 적용하였다. 가속화 알고리즘을 적용하기 위한 낮은 차수 근사와 가지치기, 재학습은 Nvidia GTX 1080Ti 환경에서 진행하였고, 추론 속도를 측정하기 위해 Intel core i7-6700k에서 실험되었다.

4.2 실험 결과 비교

제안된 CNN 가속화 알고리즘은 낮은 차수 근사를 적용한 후에 가지치기를 적용하였다. 표 1은 제안 알고리즘과 일반적인 가지치기 알고리즘이 재학습 전의 Top-1 정확도를 보여준다.

그림 5는 모델의 크기, 추론의 시간, 재학습의 회수를 비교한다. 제안된 fAlexnet 2는 일반적인 가지치기 알고리즘을 적용하는 pAlexnet보다 재학습 전의 Top-1 정확도가 3.21% 더 높다. 또한, 그림 5에서 보는 바와 같이, fAlexnet 2의 모델 크기는 pAlexnet의 모델 크기의 50.2% 정도로 매우 작다. 이것은 낮은 차수 근사로 중요 가중치들을 소수의 가중치로 집중시킨 후에 가지치기를 적용하였기 때문이다. pAlexent는 두 번째부터 네 번째 컨볼루션 계층까지 각 계층마다 동일한 비율로 가중치를 가지치기 수행한다.

표 1. 재학습 수행전의 Top-1 정확도
Table 1. Top-1 accuracy before fine-tune

Network Model	Top-1 Accuracy before Fine-Tune
oAlexNet	57.4%
pAlexNet	51.02%
fAlexnet 2	54.23%
fAlexnet 3	54.37%
fAlexnet 4	54.47%

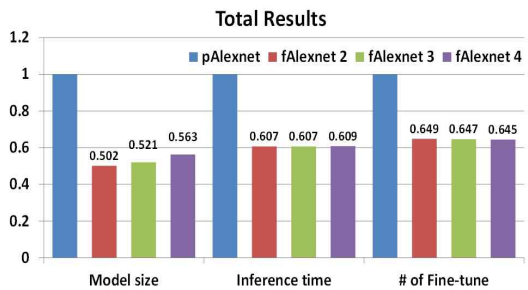


그림 5. 종합적인 결과
Fig. 5. Total results

만약 중요 가중치가 어떤 한 계층에 집중되어 있으면, 각 계층은 동일한 비율로 가중치 가지치기를 진행하기 때문에 중요한 가중치도 제거될 수 있다.

제안 알고리즘에서는 전체 컨볼루션 계층에서 기준값보다 작은 가중치를 제거하기 때문에 중요 가중치가 제거되지 않는다. fAlexnet 4의 경우에는 첫 번째 컨볼루션 계층부터 세 번째 컨볼루션 계층까지 낮은 근사 후 가지치기를 적용하지 않았다. 따라서 재학습 전의 Top-1 정확도는 fAlexnet 1과 fAlexnet 2에 비해 조금 더 향상되지만, 모델의 크기는 pAlexent의 56.3%까지 증가하였다.

낮은 차수 근사 후 가지치기 기법을 통해 가중치 개수를 일반적인 가지치기를 적용하는 pAlexnet보다 많이 줄였기 때문에, 추론 속도는 매우 빨라진다. pAlexnet의 추론 속도로 정규화하면, 제안된 fAlexnet 2, 3, 4의 연산 속도는 각각 0.607, 0.6073, 0.609가 된다. 예를 들어, fAlexnet 2은 pAlexnet보다 1.65배 빠른 추론 속도를 보여준다.

표 1에서 보듯이, 낮은 차수 근사나, 가지치기 기법을 적용하면, Top-1 정확도가 낮아진다. 따라서 Top-1 정확도를 향상시키기 위해 재학습을 수행해야 한다. 그림 5에서 # of fine-tune은 oAlexnet의 Top-1 정확도까지 올리기 위해 수행된 재학습 횟수를 의미한다. 제안된 fAlexnet에서는 Top-1 정확도 하락이 작아서, 기존의 pAlexent보다 약 33,000회 적은 재학습이 필요하다. pAlexnet은 약 94,000회의 재학습이 수행된 반면에, 제안된 fAlexnet 2, 3, 4는 약 61,000회의 재학습으로 원본 oAlexnet의 Top-1 정확도를 회복하였다. 따라서 fAlexnet 2는 재학습 횟수는 pAlexnet보다 약 1.5배 감소한다.

V. 결 론

컨볼루션 신경망 기반 딥 러닝은 고성능 컴퓨팅 시스템과 대용량 메모리를 필요로 한다. 하지만, 모바일 시스템은 제한된 자원을 내재하고 있어서, 딥 러닝 서비스를 데이터 센터에 의지한다. 따라서 제한된 자원에서 동작할 수 있는 컨볼루션 신경망 기반 딥 러닝 가속화 알고리즘이 필요하다. 제안된 가속화 알고리즘은 낮은 차수 근사와 가중치 가지치기 방법을 적용하여 신경망의 크기를 효율적으로 줄인다. 따라서 필요한 메모리가 적아지고, 연산량도 줄어들어 추론의 속도가 향상된다. 또한, 재학습의 시간도 크게 줄일 수 있는 장점이 있다. 제안된 알고리즘은 모바일 시스템에서 인공지능 서비스를 구현할 수 있는 많은 기회를 제공할 것이다. 우리는 추론의 속도를 더욱 향상시키기 위하여 Re-RAM (Resistive Random-Access Memory)을 이용한 하드웨어 가속기를 연구하고 있다.

References

[1] D. Jeong, "Trend on Artificial Intelligence Technology and Its Related Industry", JKIIT, Vol. 15, No. 2, pp. 21-28, Feb. 2017.

[2] K. Kang and D, Kang, "A Study of Face Detection Algorithm Using CNN Based on Symmetry-LGP & Uniform-LGP and the Skin Color", JKIIT, Vol. 15, No. 1, pp. 107-113, Jan. 2017.

[3] R. Girshick, "Fast R-CNN", International Conference on ComputerVision (ICCV), pp. 1440-1448, Sep. 2015.

[4] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning", Proceedings of the 25th international conference on Machine learning(ICML), pp. 160-167, Jul. 2008.

[5] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks", International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4087-4091, May 2014.

[6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition", Proceedings of the IEEE, Vol. 86, No. 11, pp. 2278-2324, Nov. 1998.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", In Advances in neural informationprocessing systems (NIPS), pp. 1097-1105, Jan. 2012.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", ICLR, arXiv preprint arXiv:1409. 1556, pp. 1-14, Apr. 2015.

[9] G. Poli and M. R. Zorzan, "Processing neocognitron of face recognition on high performance environment based on GPU with CUDA architecture", 20th International Symposium on Computer Architecture and High Performance Computing, SBAC-PAD'08. pp. 81-88, Oct. 2008.

[10] M. Rhu, N. Gimelshein, J. Clemons, A. Zulfiqar and S. W. Keckler, "vDNN: Virtualized deep neural networks for scalable, memory-efficient neural network design", In Microarchitecture (MICRO), 2016 49th Annual IEEE/ACM International Symposium, pp. 1-13, Oct. 2016.

[11] Y. D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, "Compression of deep convolutional neural networks for fast and low power mobile applications", Computer Science 2015, arXiv preprint arXiv: 1511.06530, pp. 1-16, Nov. 2015.

[12] J. Wu, C. Leng, Y. Wang, Q. Hu and J. Cheng, "Quantized convolutional neural networks for mobile devices", In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4820-4828, May 2016.

[13] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, and Y. Wang, "Going Deeper with Embedded FPGA platform for Convolutional Neural Network", In Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA), pp. 26-35, Feb. 2016.

[14] S. Anwar, K. Hwang, and W. Sung, "Structured pruning of deep convolutional neural networks", ACM Journal on Emerging Technologies in Computing Systems (JETC), Vol. 13, No. 3, 12pages, May 2017.

[15] S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding", arXiv preprint arXiv:1510.00149, pp. 1-14, Oct. 2015.

[16] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network" Advances in neural information processing systems (NIPS), pp. 1135-1143, Oct. 2015.

[17] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov and A. Rabinovich, "Going deeper with convolutions", In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1-9 Jun. 2015.

[18] J. Yu, A. Lukefahr, D. Palframan, G. Dasika, R. Das and S. Mahlke, "Scalpel: Customizing dnn pruning to the underlying hardware parallelism", In Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA), pp. 548-560, Jun. 2017.

[19] J. Chung and T. Shin, "Simplifying deep neural networks for neuromorphic architectures", Design Automation Conference (DAC), 53nd ACM/ EDAC/IEEE, pp. 1-6, Jun. 2016.

[20] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding", In Proceedings of the 22nd ACM international conference on Multimedia, pp. 675-678, Nov. 2014.

[21] K. Baker, "Singular value decomposition tutorial", The Ohio State University 24, 2005.

[22] J. Ye, "Generalized low rank approximations of matrices", Machine Learning, Vol. 61, No. 1-3, pp. 167-191, Nov. 2005.

[23] T. Roughgarden and G. Valiant, "CS168: The Modern Algorithmic Toolbox Lecture# 9: The Singular Value Decomposition (SVD) and Low-Rank Matrix Approximations", in Stanford University Lecture, pp. 2-7, Apr. 2015.

[24] J. Park, S. Li, W. Wen, P. T. P. Tang, H. Li,

Y. Chen, and P. Dubey, "Faster cnns with direct sparse convolutions and guided pruning", arXiv preprint arXiv:1608.01409, pp. Nov. 2016.

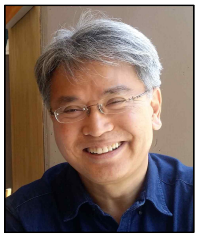
[25] A. Parashar, M. Rhu, A. Mukkara, A. Puglielli, R. Venkatesan, B. Khailany, and W. J. Dally, "Scnn: An accelerator for compressed-sparse convolutional neural networks", In Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA), pp. 27-40, Jun. 2017.

저자소개

박 성 우 (Sungwoo Park)

2015년 8월 : 단국대학교 전자전기공학부(공학사)
2015년 ~ 현재 : 단국대학교 전자전기공학과(공학석사)
관심분야 : 머신러닝, 컴퓨터 구조, 임베디드 시스템, SoC

한 경 호 (Kyong-Ho Han)



1982년 : 서울대학교 (공학사)
1984년 : 서울대학교 전자공학과(공학석사)
1992년 : 텍사스 A&M 대학교 전자공학과(공학박사)
1984년 ~ 1985년 : 삼성 HP사업부 연구원
1985년 ~ 1987년 : 한국통신 품질보증단 전임연구원
1992년 ~ 1993년 : 한국전자통신연구소(ETRI) CDMA 단말기 개발실 선임연구원
1993년 ~ 현재 : 단국대학교 전자전기공학부 교수
관심분야 : 드론, 자율주행, ITS, F/A System, 임베디드 프로세서 응용

장 우 영 (Wooyoung Jang)

1998년 2월 : 경희대학교 전파공학과(공학사)
2000년 2월 : 연세대학교 전기컴퓨터공학과(공학석사)
2011년 5월 : 텍사스 주립 대학교 (Univ. of Texas at Austin) 전기컴퓨터공학과(공학박사)
2000년 ~ 2012년 : 삼성전자 System LSI 사업부 SoC 개발실 책임연구원
2013년 ~ 현재 : 단국대학교 전자전기공학부 조교수
관심분야 : 컴퓨터 구조, 설계 자동화, 머신러닝