(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2023-0030290
(43) 공개일자 2023년03월06일

(51) 국제특허분류(Int. Cl.)
G06N 3/08 (2023.01) *G06N 3/04* (2023.01)
G06N 3/063 (2023.01)
(52) CPC특허분류
G06N 3/082 (2023.01)
G06N 3/04 (2023.01)
(21) 출원번호 10-2021-0112340
(22) 출원일자 2021년08월25일
심사청구일자 2021년08월25일

(71) 출원인
인하대학교 산학협력단
인천광역시 미추홀구 인하로 100(용현동, 인하대학교)
(72) 발명자
최동완
인천광역시 미추홀구 용정공원로 33, 108동 1303호(용현동, 인천 SK Sky VIEW)
이건호
인천광역시 남동구 독곡로16번길 22-19, 304호(서창동, 하은빌라)
김민수
경상북도 포항시 남구 지곡로 294, 221동 1104호(지곡동, 효자그린아파트)
(74) 대리인
양성보

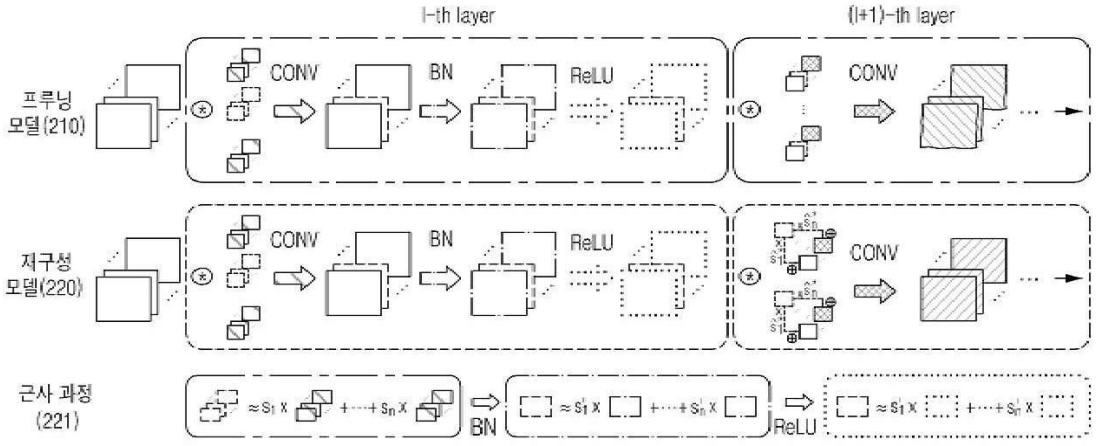
전체 청구항 수 : 총 8 항

(54) 발명의 명칭 **네트워크 프루닝 기법을 통해 경량화 된 신경망에 대한 데이터와 미세 조정 학습이 필요 없는 성능 복구 방법 및 장치**

(57) 요 약

네트워크 프루닝 기법을 통해 경량화 된 신경망에 대한 데이터와 미세 조정 학습이 필요 없는 성능 복구 방법 및 장치가 제시된다. 본 발명에서 제안하는 네트워크 프루닝 기법을 통해 경량화 된 신경망에 대한 데이터와 미세 조정 학습이 필요 없는 성능 복구 방법은 사전학습된 합성곱 신경망의 레이어에서 제거되는 필터와 보존되는 필터간의 선형관계를 이용하여 손상된 필터를 나타내는 단계, 배치 정규화 레이어와 활성화 레이어에 따른 상기 손상된 필터로 인한 손실 함수를 정의하는 단계 및 상기 손실함수를 최소화하는 계수값을 전달행렬의 요소값으로 나타내어 사전학습된 합성곱 신경망의 필터와 프루닝 행렬의 연산을 수행함으로써 프루닝으로 압축된 신경망의 성능을 복구하는 단계를 포함한다.

대 표 도



(52) CPC특허분류
G06N 3/063 (2013.01)

이 발명을 지원한 국가연구개발사업
과제고유번호 1711135511
과제번호 2021R1F1A1060160
부처명 과학기술정보통신부
과제관리(전문)기관명 한국연구재단
연구사업명 기본연구
연구과제명 [Ezbaro] 질의 가능한 인공지능: 상호작용형 XAI 서비스를 실현하기 위한 질의 기반 딥러닝 모델 해석 기술
기 여 율 40/100
과제수행기관명 인하대학교 산학협력단
연구기간 2021.06.01 ~ 2022.02.28
이 발명을 지원한 국가연구개발사업
과제고유번호 1711126439
과제번호 2019-0-00240-003
부처명 과학기술정보통신부
과제관리(전문)기관명 정보통신기획평가원
연구사업명 방송통신산업기술개발사업
연구과제명 [Ezbaro] 자율협력형 스마트 임베디드 기기에서의 실시간 추론을 위한 딥러닝 모델 병합 및 분할 플랫폼 개발(3차년도)
기 여 율 30/100
과제수행기관명 인하대학교 산학협력단
연구기간 2021.01.01 ~ 2021.12.31
이 발명을 지원한 국가연구개발사업
과제고유번호 1711126440
과제번호 2019-0-00064-003
부처명 과학기술정보통신부
과제관리(전문)기관명 정보통신기획평가원
연구사업명 방송통신산업기술개발사업
연구과제명 [Ezbaro] 커넥티드 카를 위한 지능형 모바일 엣지 클라우드 솔루션 개발(3차년도)
기 여 율 20/100
과제수행기관명 인하대학교 산학협력단
연구기간 2021.01.01 ~ 2021.12.31
이 발명을 지원한 국가연구개발사업
과제고유번호 1711126259
과제번호 2020-0-01389-002
부처명 과학기술정보통신부
과제관리(전문)기관명 정보통신기획평가원
연구사업명 인공지능융합연구센터지원사업(국고)
연구과제명 [Ezbaro][정부] 인공지능융합연구센터지원(2차년도)
기 여 율 10/100
과제수행기관명 인하대학교 산학협력단
연구기간 2021.01.01 ~ 2021.12.31

명 세 서

청구범위

청구항 1

사전학습된 합성곱 신경망의 레이어에서 제거되는 필터와 보존되는 필터간의 선형관계를 이용하여 손상된 필터를 나타내는 단계;

배치 정규화 레이어와 활성화 레이어에 따른 상기 손상된 필터로 인한 손실 함수를 정의하는 단계; 및

상기 손실함수를 최소화하는 계수값을 전달행렬의 요소값으로 나타내어 사전학습된 합성곱 신경망의 필터와 프루닝 행렬의 연산을 수행함으로써 프루닝으로 압축된 신경망의 성능을 복구하는 단계

를 포함하는 합성곱 신경망의 성능 복구 방법.

청구항 2

제1항에 있어서,

상기 사전학습된 합성곱 신경망의 레이어에서 제거되는 필터와 보존되는 필터간의 선형관계를 이용하여 손상된 필터를 나타내는 단계는,

상기 사전학습된 합성곱의 필터 프루닝에서 사용되는 프루닝 행렬을 전달 행렬로 변형시킴으로써 제거되는 필터의 정보를 보존되는 필터에게 상속시키고, 제거되는 필터를 보존되는 필터들의 선형 결합으로 나타내는

합성곱 신경망의 성능 복구 방법.

청구항 3

제1항에 있어서,

배치 정규화 레이어와 활성화 레이어에 따른 상기 손상된 필터로 인한 손실 함수를 정의하는 단계는,

손실함수를 최소화하기 위해 활성화맵을 특징맵으로 근사화 한 후, 잔차 오류(Residual Error)에 대하여 배치 정규화 레이어와 활성화 레이어에 따른 활성화맵을 특징맵이 배치 정규화 레이어를 통과한 값으로 나타내어 손실 함수를 정의하고, 상기 손실함수의 잔차 오류, 배치정규화 오류, 활성화 오류를 최소화 하는 계수값을 찾는

합성곱 신경망의 성능 복구 방법.

청구항 4

제1항에 있어서,

상기 손실함수를 최소화하는 계수값을 전달행렬의 요소값으로 나타내어 사전학습된 합성곱 신경망의 필터와 프루닝 행렬의 연산을 수행함으로써 프루닝으로 압축된 신경망의 성능을 복구하는 단계는,

상기 손실함수를 최소화하는 계수값을 상기 제거되는 필터에 곱하고, 상기 보존되는 필터에 더함으로써 사전학습된 합성곱 신경망과 압축된 신경망의 재구성 오류를 최소화하는 전달 행렬을 찾고,

상기 손실함수를 최소화하는 계수값을 전달행렬의 요소값으로 나타내어 사전학습된 합성곱 신경망의 필터와 프루닝 행렬의 2-모드곱(2-mode product) 연산을 수행함으로써 데이터와 학습없이 프루닝으로 압축된 신경망의 성능을 복구하는

합성곱 신경망의 성능 복구 방법.

청구항 5

사전학습된 합성곱 신경망의 레이어에서 제거되는 필터와 보존되는 필터간의 선형관계를 이용하여 손상된 필터를 나타내고, 배치 정규화 레이어와 활성화 레이어에 따른 상기 손상된 필터로 인한 손실 함수를 정의하는 프루닝 모델링부; 및

손실함수를 최소화하는 계수값을 전달행렬의 요소값으로 나타내어 사전학습된 합성곱 신경망의 필터와 프루닝 행렬의 연산을 수행함으로써 프루닝으로 압축된 신경망의 성능을 복구하는 재구성 모델링부

를 포함하는 합성곱 신경망의 성능 복구 장치.

청구항 6

제5항에 있어서,

상기 프루닝 모델링부는,

상기 사전학습된 합성곱의 필터 프루닝에서 사용되는 프루닝 행렬을 전달 행렬로 변형시킴으로써 제거되는 필터의 정보를 보존되는 필터에게 상속시키고, 제거되는 필터를 보존되는 필터들의 선형 결합으로 나타내는

합성곱 신경망의 성능 복구 장치.

청구항 7

제5항에 있어서,

상기 프루닝 모델링부는,

손실함수를 최소화하기 위해 활성화맵을 특징맵으로 근사화 한 후, 잔차 오류(Residual Error)에 대하여 배치 정규화 레이어와 활성화 레이어에 따른 활성화맵을 특징맵이 배치 정규화 레이어를 통과한 값으로 나타내어 손실 함수를 정의하고, 상기 손실함수의 잔차 오류, 배치정규화 오류, 활성화 오류를 최소화 하는 계수값을 찾는

합성곱 신경망의 성능 복구 장치.

청구항 8

제5항에 있어서,

상기 재구성 모델링부는,

상기 손실함수를 최소화하는 계수값을 상기 제거되는 필터에 곱하고, 상기 보존되는 필터에 더함으로써 사전학습된 합성곱 신경망과 압축된 신경망의 재구성 오류를 최소화하는 전달 행렬을 찾고,

상기 손실함수를 최소화하는 계수값을 전달행렬의 요소값으로 나타내어 사전학습된 합성곱 신경망의 필터와 프루닝 행렬의 2-모드곱(2-mode product) 연산을 수행함으로써 데이터와 학습없이 프루닝으로 압축된 신경망의 성능을 복구하는

합성곱 신경망의 성능 복구 장치.

발명의 설명

기 술 분 야

[0001] 본 발명은 네트워크 프루닝 기법을 통해 경량화 된 신경망에 대한 데이터와 미세 조정 학습이 필요 없는 성능 복구 방법 및 장치에 관한 것이다.

배 경 기 술

[0002] 네트워크 프루닝은 과적합된 신경망을 압축하기 위해 활발히 연구가 이루어지고 있는 분야 중 하나이다. 네트워크 프루닝은 가능한 한 사전학습된 합성곱 신경망의 성능을 보존하면서 신경망의 파라미터를 제거함으로써 파라미터의 수를 줄이며 신경망의 추론 속도를 줄이는 것이 목표이다. 네트워크 프루닝은 주로 두 가지 웨이트(unstructured pruning)와 필터 프루닝(filter pruning)으로 나뉜다. 웨이트 프루닝은 신경망의 구조를 변경하지 않고, 파라미터 자체를 '0'으로 대체함으로써 신경망을 희소하게(sparse) 만드는 기법이며, 이 기법은 신경망의 파라미터의 수를 줄이는 데 효과적이나 신경망 구조를 변경하지 않기 때문에 추론 속도를 줄이기 위해서는 특별한 하드웨어와 소프트웨어가 필요하다. 반면, 필터 프루닝은 신경망의 필터를 제거함으로써 신경망의 구조를 변경시키기 때문에 추가적인 하드웨어와 소프트웨어 없이 추론 속도를 줄이는 것이 가능하다.

[0003] 일반적으로, 필터 프루닝은 학습되지 않은 신경망을 학습시키는 과정에서 반복적으로 신경망의 필터를 제거하는

반복적인 프루닝(iterative pruning)과 사전학습된 합성곱 신경망을 프루닝하여 파라미터를 제거한 후 재학습을 통하여 성능을 복구하는 원샷 프루닝(one-shot pruning) 기법이 존재한다. 하지만 반복적인 프루닝과 원샷 프루닝은 학습을 위한 충분한 데이터, 하드웨어가 필요하며, 학습과정을 많은 시간이 소요되는 문제점이 있다. 이러한 문제를 해결하고자 다음과 같은 종래기술들이 제안되었다.

[0004] 종래기술[1]은 적은 학습과정으로 압축된 성능을 복구하기 위해 원본 데이터를 이용하여 신경망의 각 레이어마다 제거할 필터를 재구성 오류를 기반하여 그리디(greedy)하게 선택하는 방법을 제안하였다. 하지만 종래기술[1]은 재구성 오류를 만들어내기 위한 데이터와 재학습을 위한 원본 데이터 전체가 필요하다는 문제점이 여전히 존재한다. 종래기술[2]은 적은 데이터를 이용하여 제거해야 할 필터를 쿨백-라이블러 발산(Kullback-Leibler divergence) 기준(criterion)을 이용하여 신경망 내 전역적으로(globally) 선택하는 방법을 제안하였으며 적은 학습만 압축된 신경망을 학습시키기위해 지식 증류(Knowledge Distilation)을 이용하여 재학습하는 방법을 제안하였다. 하지만 여전히 데이터와 재학습이 필요하다는 문제는 해결하지 못하였다. 데이터와 재학습 없이 사전학습된 합성곱 신경망을 프루닝한 이후 성능을 복구하기 위해, 종래기술에 따른 원투원(one-to-one) 방식[3,4]은 제거되는 필터마다 가장 유사한 보존되는 하나의 필터를 찾아 제거되는 필터에 스케일을 곱하고 이를 보존되는 필터에 더함으로써 제거되는 필터로 인해 발생하는 성능 손실을 복구한다.

발명의 내용

해결하려는 과제

[0005] 본 발명이 이루고자 하는 기술적 과제는, 종래기술의 유사도를 기반하여 압축된 신경망을 복구하는 방법에서 신경망 후반 레이어에서 유사하지 않은 필터를 선택할 수 있는 문제점을 해결하기 위해 제거되는 필터와 가장 유사한 필터 하나를 선택하는 방법이 아닌 제거되는 필터와 보존되는 필터들 사이의 선형결합(linear combination)을 기반으로 압축된 신경망의 성능을 복구하는 방법 및 장치를 제공하는데 있다.

과제의 해결 수단

[0006] 일 측면에 있어서, 본 발명에서 제안하는 네트워크 프루닝 기법을 통해 경량화 된 신경망에 대한 데이터와 미세조정 학습이 필요 없는 성능 복구 방법은 사전학습된 합성곱 신경망의 레이어에서 제거되는 필터와 보존되는 필터간의 선형관계를 이용하여 손상된 필터를 나타내는 단계, 배치 정규화 레이어와 활성화 레이어에 따른 상기 손상된 필터로 인한 손실 함수를 정의하는 단계 및 상기 손실함수를 최소화하는 계수값을 전달행렬의 요소값으로 나타내어 사전학습된 합성곱 신경망의 필터와 프루닝 행렬의 연산을 수행함으로써 프루닝으로 압축된 신경망의 성능을 복구하는 단계를 포함한다.

[0007] 상기 사전학습된 합성곱 신경망의 레이어에서 제거되는 필터와 보존되는 필터간의 선형관계를 이용하여 손상된 필터를 나타내는 단계는 상기 사전학습된 합성곱의 필터 프루닝에서 사용되는 프루닝 행렬을 전달 행렬로 변형시킴으로써 제거되는 필터의 정보를 보존되는 필터에게 상속시키고, 제거되는 필터를 보존되는 필터들의 선형결합으로 나타낸다.

[0008] 배치 정규화 레이어와 활성화 레이어에 따른 상기 손상된 필터로 인한 손실 함수를 정의하는 단계는 손실함수를 최소화하기 위해 활성화맵을 특징맵으로 근사화 한 후, 잔차 오류(Residual Error)에 대하여 배치 정규화 레이어와 활성화 레이어에 따른 활성화맵을 특징맵이 배치 정규화 레이어를 통과한 값으로 나타내어 손실 함수를 정의하고, 상기 손실함수의 잔차 오류, 배치정규화 오류, 활성화 오류를 최소화 하는 계수값을 찾는다.

[0009] 상기 손실함수를 최소화하는 계수값을 전달행렬의 요소값으로 나타내어 사전학습된 합성곱 신경망의 필터와 프루닝 행렬의 연산을 수행함으로써 프루닝으로 압축된 신경망의 성능을 복구하는 단계는 상기 손실함수를 최소화하는 계수값을 상기 제거되는 필터에 곱하고, 상기 보존되는 필터에 더함으로써 사전학습된 합성곱 신경망과 압축된 신경망의 재구성 오류를 최소화하는 전달 행렬을 찾고, 상기 손실함수를 최소화하는 계수값을 전달행렬의 요소값으로 나타내어 사전학습된 합성곱 신경망의 필터와 프루닝 행렬의 2-모드곱(2-mode product) 연산을 수행함으로써 데이터와 학습없이 프루닝으로 압축된 신경망의 성능을 복구한다.

[0010] 또 다른 일 측면에 있어서, 본 발명에서 제안하는 네트워크 프루닝 기법을 통해 경량화 된 신경망에 대한 데이터와 미세 조정 학습이 필요 없는 성능 복구 장치는 사전학습된 합성곱 신경망의 레이어에서 제거되는 필터와 보존되는 필터간의 선형관계를 이용하여 손상된 필터를 나타내고, 배치 정규화 레이어와 활성화 레이어에 따른 상기 손상된 필터로 인한 손실 함수를 정의하는 프루닝 모델링부 및 손실함수를 최소화하는 계수값을 전달행렬의 요소값으로 나타내어 사전학습된 합성곱 신경망의 필터와 프루닝 행렬의 연산을 수행함으로써 프루닝으로 압

축된 신경망의 성능을 복구하는 재구성 모델링부를 포함한다.

발명의 효과

[0011] 본 발명의 실시예들에 따른 필터 프루닝을 이용하여 압축된 신경망을 데이터와 학습없이 원본 신경망의 성능으로 복구하는 방법 및 장치를 통해 제거되는 필터와 가장 유사한 필터 하나를 선택하는 방법이 아닌 제거되는 필터와 보존되는 필터들 사이의 선형결합(linear combination)을 기반으로 압축된 신경망의 성능을 복구하여 종래기술의 유사도를 기반하여 압축된 신경망을 복구하는 방법에서의 신경망 후반 레이어에서 유사하지 않은 필터를 선택할 수 있는 문제점을 해결할 수 있다.

도면의 간단한 설명

[0012] 도 1은 종래기술에 따른 심층 신경망의 레이어를 나타내는 도면이다.

도 2는 본 발명의 일 실시예에 따른 네트워크 프루닝 기법을 통해 경량화 된 신경망에 대한 데이터와 미세 조정 학습이 필요 없는 성능 복구 장치의 구성을 나타내는 도면이다.

도 3은 본 발명의 일 실시예에 따른 네트워크 프루닝 기법을 통해 경량화 된 신경망에 대한 데이터와 미세 조정 학습이 필요 없는 성능 복구 방법을 설명하기 위한 흐름도이다.

도 4는 본 발명의 일 실시예에 따른 프루닝 행렬 및 전달 행렬을 설명하기 위한 도면이다.

도 5는 본 발명의 일 실시예에 따른 데이터와 학습없이 프루닝으로 압축된 신경망의 성능을 복구하기 위한 알고리즘을 나타내는 도면이다.

도 6은 본 발명의 일 실시예에 따른 합성곱 신경망의 성능 복구 방법과 NM과의 잔차 오류(RE), 배치 정규화 오류(BE), 가중화된 평균 재구성 오류(WARE)를 나타낸 도면이다.

발명을 실시하기 위한 구체적인 내용

[0013] 이하, 본 발명의 실시 예를 첨부된 도면을 참조하여 상세하게 설명한다.

[0015] 도 1은 종래기술에 따른 심층 신경망의 레이어를 나타내는 도면이다.

[0016] 데이터와 학습없이 프루닝으로 압축된 신경망의 성능을 복구시키기 위해 종래기술[3,4]에서는 제거되는 필터와 보존되는 필터간의 유사도를 이용하여 성능을 회복시키는 방법을 이용하였다. 하지만 도 1[4]과 같이, 최근 개발된 심층 신경망은 많은 레이어를 가지며, 레이어가 깊어질수록 필터간의 유사도는 점점 떨어진다. 즉, 유사도를 기반하여 압축된 신경망을 복구하는 방법은 신경망의 후반 레이어에서 유사하지 않은 필터를 선택할 수 있는 문제점이 있다[3,4]. 이러한 문제를 해결하고자 NM[4]은 신경망의 배치 정규화(batch normalzation)와 활성화 함수(activation function)를 고려한 거리기반 측정법을 이용하여 유사한 필터를 선택하는 방법을 제안하였다. 하지만 이러한 방법은 여전히 유사도 기반으로, 유사하지 않은 필터를 잘못 선택할 가능성이 존재한다.

[0017] 따라서 본 발명에서는 제거되는 필터와 가장 유사한 필터 하나를 선택하는 방법이 아닌 제거되는 필터와 보존되는 필터들 사이의 선형결합(linear combination)을 기반으로 압축된 신경망의 성능을 복구하고자 한다. 본 발명에서는 필터 프루닝을 이용하여 압축된 신경망을 데이터와 학습없이 원본 신경망의 성능으로 복구하는 방법 및 장치를 제안한다.

[0019] 도 2는 본 발명의 일 실시예에 따른 네트워크 프루닝 기법을 통해 경량화 된 신경망에 대한 데이터와 미세 조정 학습이 필요 없는 성능 복구 장치의 구성을 나타내는 도면이다.

[0020] 제안하는 네트워크 프루닝 기법을 통해 경량화 된 신경망에 대한 데이터와 미세 조정 학습이 필요 없는 성능 복구 장치는 프루닝 모델링부(210) 및 재구성 모델링부(220)를 포함한다.

[0021] 본 발명의 실시예에 따른 네트워크 프루닝 기법을 통해 경량화 된 신경망에 대한 데이터와 미세 조정 학습이 필요 없는 성능 복구 장치를 도 2를 참조하여 간략히 설명하면, 제안된 장치는 데이터와 학습없이 프루닝으로 인해 발생하는 압축된 신경망의 성능하락을 복구할 수 있다. 먼저, 프루닝으로 인한 성능 하락은 도 2에서와

같이, 프루닝 모델링부(다시 말해, 프루닝된 모델(Pruned Model))(210)이 사전학습된 합성곱 신경망의 l번째 레이어에서 제거되는 필터로 인해 l번째 레이어의 출력 채널이 제거되고 이로 인해, 그 다음 레이어인 $l+1$번째 필터의 채널이 연쇄적으로 제거됨으로써 출력의 손상이 발생된다. 이러한 오류(error)는 각 레이어 마다 누적되어 최종적으로 압축된 신경망의 성능하락이 발생한다. 이러한 성능 하락을 복구하기 위해, 본 발명에서는 제거되는 필터와 보존되는 필터간의 선형관계를 이용하고 배치 정규화 레이어와 활성화 레이어를 고려한 손실함수를 정의한다. 그리고 정의된 손실함수를 최소화하는 $\hat{s}_k^* \hat{s}_k^*$를 찾아, 제거되는 필터에 곱하여 보존되는 필터들에게 더함으로써 사전학습된 합성곱 신경망과 압축된 신경망의 재구성 오류를 최소화할 수 있도록 한다. 도 2에서 재구성 모델링부(다시 말해, 재구성된 모델(Restored Model))(220)에 관하여 더욱 상세히 설명하면, 본 발명은 먼저 합성곱 신경망에서 필터 프루닝을 N-mode product[5]을 이용하여 나타내고 원본 신경망과 프루닝된 신경망 사이의 재구성 오류를 최소화하는 문제로 정의하여 데이터와 학습없이 프루닝으로 압축된 신경망을 회복 방법을 제안한다.

[0022] 본 발명의 실시예에 따른 프루닝 모델링부(210)는 사전학습된 합성곱 신경망의 레이어에서 제거되는 필터와 보존되는 필터간의 선형관계를 이용하여 손상된 필터를 나타내고, 배치 정규화 레이어와 활성화 레이어에 따른 상기 손상된 필터로 인한 손실 함수를 정의한다. 프루닝 모델링부(210)는 상기 사전학습된 합성곱의 필터 프루닝에서 사용되는 프루닝 행렬을 전달 행렬로 변형시킴으로써 제거되는 필터의 정보를 보존되는 필터에게 상속시키고, 제거되는 필터를 보존되는 필터들의 선형 결합으로 나타낼 수 있다.

[0023] 본 발명의 실시예에 따른 프루닝 모델링부(210)는 손실함수를 최소화하기 위해 활성화맵을 특징맵으로 근사화한 후(221), 잔차 오류(Residual Error)에 대하여 배치 정규화 레이어와 활성화 레이어에 따른 활성화맵을 특징맵이 배치 정규화 레이어를 통과한 값으로 나타내어 손실 함수를 정의하고, 상기 손실함수의 잔차 오류, 배치정규화 오류, 활성화 오류를 최소화 하는 계수값을 찾을 수 있다.

[0024] 재구성 모델링부(220)는 손실함수를 최소화하는 계수값을 전달행렬의 요소값으로 나타내어 사전학습된 합성곱 신경망의 필터와 프루닝 행렬의 연산을 수행함으로써 프루닝으로 압축된 신경망의 성능을 복구한다.

[0025] 본 발명의 실시예에 따른 재구성 모델링부(220)는 손실함수를 최소화하는 계수값을 상기 제거되는 필터에 곱하고, 상기 보존되는 필터에 더함으로써 사전학습된 합성곱 신경망과 압축된 신경망의 재구성 오류를 최소화하는 전달 행렬을 찾는다. 이후, 상기 손실함수를 최소화하는 계수값을 전달행렬의 요소값으로 나타내어 사전학습된 합성곱 신경망의 필터와 프루닝 행렬의 2-모드곱(2-mode product) 연산을 수행함으로써 데이터와 학습없이 프루닝으로 압축된 신경망의 성능을 복구할 수 있다.

[0027] 도 3은 본 발명의 일 실시예에 따른 네트워크 프루닝 기법을 통해 경량화 된 신경망에 대한 데이터와 미세 조정 학습이 필요 없는 성능 복구 방법을 설명하기 위한 흐름도이다.

[0028] 제안하는 네트워크 프루닝 기법을 통해 경량화 된 신경망에 대한 데이터와 미세 조정 학습이 필요 없는 성능 복구 방법은 사전학습된 합성곱 신경망의 레이어에서 제거되는 필터와 보존되는 필터간의 선형관계를 이용하여 손상된 필터를 나타내는 단계(310), 배치 정규화 레이어와 활성화 레이어에 따른 상기 손상된 필터로 인한 손실함수를 정의하는 단계(320) 및 상기 손실함수를 최소화하는 계수값을 전달행렬의 요소값으로 나타내어 사전학습된 합성곱 신경망의 필터와 프루닝 행렬의 연산을 수행함으로써 프루닝으로 압축된 신경망의 성능을 복구하는 단계(330)를 포함한다.

[0029] 단계(310)에서, 사전학습된 합성곱 신경망의 레이어에서 제거되는 필터와 보존되는 필터간의 선형관계를 이용하여 손상된 필터를 나타낸다.

[0030] 본 발명의 실시예에 따른 단계(310)에서, 프루닝 모델링부는 프루닝 모델링부는 상기 사전학습된 합성곱의 필터 프루닝에서 사용되는 프루닝 행렬을 전달 행렬로 변형시킴으로써 제거되는 필터의 정보를 보존되는 필터에게 상속시키고, 제거되는 필터를 보존되는 필터들의 선형 결합으로 나타낼 수 있다.

[0031] 단계(320)에서, 배치 정규화 레이어와 활성화 레이어에 따른 상기 손상된 필터로 인한 손실 함수를 정의한다.

[0032] 본 발명의 실시예에 따른 단계(320)에서, 프루닝 모델링부는 손실함수를 최소화하기 위해 활성화맵을 특징맵으로 근사화 한 후, 잔차 오류(Residual Error)에 대하여 배치 정규화 레이어와 활성화 레이어에 따른 활성화맵을 특징맵이 배치 정규화 레이어를 통과한 값으로 나타내어 손실 함수를 정의하고, 상기 손실함수의 잔차 오류, 배

치정규화 오류, 활성화 오류를 최소화 하는 계수값을 찾을 수 있다.

[0033] 단계(330)에서, 손실함수를 최소화하는 계수값을 전달행렬의 요소값으로 나타내어 사전학습된 합성곱 신경망의 필터와 프루닝 행렬의 연산을 수행함으로써 프루닝으로 압축된 신경망의 성능을 복구한다.

[0034] 본 발명의 실시예에 따른 단계(330)에서, 재구성 모델링부는 손실함수를 최소화하는 계수값을 상기 제거되는 필터에 곱하고, 상기 보존되는 필터에 더함으로써 사전학습된 합성곱 신경망과 압축된 신경망의 재구성 오류를 최소화하는 전달 행렬을 찾는다. 이후, 상기 손실함수를 최소화하는 계수값을 전달행렬의 요소값으로 나타내어 사전학습된 합성곱 신경망의 필터와 프루닝 행렬의 2-모드곱(2-mode product) 연산을 수행함으로써 데이터와 학습 없이 프루닝으로 압축된 신경망의 성능을 복구할 수 있다.

[0036] 도 4는 본 발명의 일 실시예에 따른 프루닝 행렬 및 전달 행렬을 설명하기 위한 도면이다.

[0037] 도 4(a)는 본 발명의 실시예에 따른 프루닝 행렬, 도 4(b)는 LBYL에 대한 전달 행렬 및 도 4(c)는 원투원에 대한 전달행렬을 나타낸다.

[0038] 본 발명의 실시예에 따른 사전학습된 합성곱 신경망에서의 필터 프루닝을 설명하기 앞서, L개의 레이어를 가지는 합성곱 신경망의 l번째 연산에서 얻은 결과는 다음과 같다:

[0039]
$$\mathbf{Z}^{(\ell)} = \mathbf{A}^{(\ell-1)} \circledast \mathbf{W}^{(\ell)}$$
$$\mathbf{A}^{(\ell)} = \mathcal{F}(\mathcal{N}(\mathbf{Z}^{(\ell)}))$$

[0040] 여기서, $\circledast$는 합성곱 연산을 의미하며, $A^{(l-1)} \in \mathbb{R}^{n\times w\times h}$은 $l-1$번째 레이어의 출력이면서 l번째 레이어의 입력 활성화맵이며, $W^{(l)} \in \mathbb{R}^{m\times n\times k\times k}$은 합성곱 필터를 의미한다. $Z^{(l)}, A^{(l)} \in \mathbb{R}^{m\times w\times h}$은 각각 특징맵, 활성화맵을 나타낸다. 이때 m은 필터의 수를 의미하며 n은 이전 레이어의 활성화맵의 채널 수이며, w×h는 활성화맵의 크기이며, k×k는 필터의 크기를 의미한다. 그리고 F는 활성화 레이어이며, N은 배치 정규화 레이어이다.

[0041] l번째 레이어에서 필터 프루닝을 수행하면 사전학습된 합성곱 신경망의 구조가 변경되어, 합성곱 필터의 차원이 변경되기 때문에 손상된 필터$\tilde{W}^{(l)} \in \mathbb{R}^{t\times n\times k\times k}$, 특징맵$\tilde{Z}^{(l)} \in \mathbb{R}^{t\times w\times h}$, 활성화맵$\tilde{A}^{(l)} \in \mathbb{R}^{t\times w\times h}$으로 변경된다. 여기서 t는 m보다 작은 남아있는 필터의 수를 의미한다. 이를 사전학습된 합성곱 신경망의 필터 $W^{(l)}$와 프루닝 행렬(pruning matrix) $\boldsymbol{S} \in \mathbb{R}^{m\times t}$의 1-모드곱(1-mode product)를 이용하여 도 4(a)와 같이 나타낼 수 있다.

[0042] 여기서, 각각의 i_k'번째 필터는 제거되지 않으며, (m-t)개의 필터들은 완전히 제거된다.

[0043] 신경망의 l번째 레이어에서 제거되는 필터로 인해 l번째 레이어의 출력 채널이 제거되고 이로 인해 그 다음 레이어인 $l+1$번째 필터의 채널이 연쇄적으로 제거됨으로써 출력의 손상이 발생된다. 즉, $l+1$번째 필터 $W^{(l+1)} \in \mathbb{R}^{m'\times m\times k'\times k'}$ 는 $\tilde{W}^{(l+1)} \in \mathbb{R}^{m'\times t\times k'\times k'}$로 변경된다. 여기서 m'은 $l+1$번째 레이어의 필터의 수이며, $k'\times k'$는 필터의 크기이다. 필터의 차원이 변경되지만 필터의 수는 변경되지 않기 때문에 $l+1$번째 레이어의 출력의 차원은 사전학습된 합성곱 신경망의 출력 차원과 동일하다. 하지만 제거된 필터의 채널로 인해 발생되는 오류(error)는 각 레이어마다 누적되어 최종적으로 압축된 신경망의 성능하락이 발생한다. 손상된 필터 $\tilde{W}^{(l+1)}$는 사전학습된 필터$W^{(l+1)}$와 프루닝 행렬S^T의 2-모드곱(2-mode product)으로 나타낼 수 있다:

[0044]
$$\tilde{\mathbf{Z}}^{(\ell+1)} = \tilde{\mathbf{A}}^{(\ell)} \circledast \tilde{\mathbf{W}}^{(\ell+1)} \not\approx \mathbf{Z}^{(\ell+1)}$$
$$\tilde{\mathbf{W}}^{(\ell+1)} = \mathbf{W}^{(\ell+1)} \times_2 \boldsymbol{\mathcal{S}}^T$$

[0045] 여기서, $\tilde{Z}^{(l+1)}$은 손상된 특징맵을 의미하며 $Z^{(l+1)}$는 사전학습된 합성곱 신경망의 특징맵이다.

[0046] 본 발명의 실시예에 따른 필터 프루닝에서 사용되는 도 4(a)와 같은 프루닝 행렬 $\boldsymbol{S}$를 도 4(b)와 같이 전달 행렬 $\boldsymbol{S}^*$로 변형시킴으로써 제거되는 필터의 정보를 보존되는 필터에게 상속시키고자 한다. 즉, i번째 레이어의 필터가 제거되어 연쇄적으로 $l+1$번째 레이어 필터의 채널이 제거되었을 때, 압축된 신경망의 출력값 $\hat{Z}^{(l+1)}$이 사전학습된 합성곱 신경망의 출력값 $Z^{(l+1)}$에 근사할 수 있는 전달 행렬 $\boldsymbol{S}^*$을 찾아 손상된 성능을 복구하고자 한다.

[0047] 따라서, 본 발명에서는 다음과 같이 사전학습된 합성곱 신경망과 압축된 신경망의 $l+1$번째 특징맵 사이의 재구성 오류를 최소화하는 전달 행렬을 찾는 것을 목적으로 하고, 목적식은 다음과 같이 나타낼 수 있다:

[0048] $$\sum_{i=1}^{m'} \|\mathbf{Z}_i^{(\ell+1)} - \hat{\mathbf{Z}}_i^{(\ell+1)}\|_1$$

[0049] 하지만 특징맵은 데이터가 있어야 얻을 수 있는 출력값이다. 따라서 본 발명에서는 데이터 없이 특징맵 사이의 재구성 오류를 최소화하는 방법을 제안한다. 이를 위해, 제거되는 하나의 필터는 보존되는 필터들의 선형 결합으로 나타낼 수 있다는 가정한다. 이때 얻은 계수값을 제거되는 필터의 가중치와 계수값 $s_{i'_k}$을 곱하여 보존되는 필터들 각각에게 더한다. j번째 필터 하나가 제거되었을 때, 전달 행렬 $\boldsymbol{S}^* \in \mathbb{R}^{m\times(m-1)}$은 다음과 같이 정의된다:

[0050] $$\boldsymbol{S}^*_{i,k} = \begin{cases} 1 & \text{if } i = i'_k \\ s_{i'_k} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \quad \text{s.t. } i, i'_k \in [1, m] \text{ and } k \in [1, m-1]$$

[0051] 여기서 $s_{i'_k}$는 스칼라로서, 제거되는 j번째 필터와 얼마나 관련있는지 정도를 나타낸다. 도 4(b)는 제거되는 필터의 수가 2개인 경우의 예시이다.

[0052] 합성곱 신경망의 하나의 레이어는 합성곱 필터와 배치 정규화 레이어, 활성화 레이어로 구성된다. 하지만, 먼저 배치 정규화와 활성화 레이어를 고려하지 않고 재구성 오류가 어떻게 이루어져 있는지 살펴보고자 한다. 위의 목적식에서 단순함을 위해 $\hat{Z}^{(l+1)}$의 i번째 채널만을 고려하게되는 경우 압축된 신경망의 특징맵 $\hat{Z}_i^{(l+1)}$은 다음과 같이 나타낼 수 있다:

[0053] $$\hat{\mathbf{Z}}_i^{(\ell+1)} = \sum_{k=1,k\neq j}^{m} \mathbf{A}_k^{(\ell)} \circledast (\mathbf{W}_{i,k}^{(\ell+1)} + s_k \mathbf{W}_{i,j}^{(\ell+1)})$$

[0054] 그러면, $l+1$번째 레이어에서의 i번째 특징맵에 대한 사전학습된 합성곱 신경망과 압축된 신경망의 재구성 오류는 다음과 같이 나타낼 수 있다:

[0055] $$\mathbf{Z}_i^{(\ell+1)} - \hat{\mathbf{Z}}_i^{(\ell+1)} = (\mathbf{A}_j^{(\ell)} - \sum_{k=1,k\neq j}^{m} s_k \mathbf{A}_k^{(\ell)}) \circledast \mathbf{W}_{i,j}^{(\ell+1)}$$

[0056] 위의 식으로부터 l번째 레이어에서 j 번째 필터가 제거되었을 때, $l+1$번째 레이어의 재구성 오류는 다음이 나타낼 수 있다:

[0057] $$\sum_{i=1}^{m'} \|\mathbf{Z}_i^{(\ell+1)} - \hat{\mathbf{Z}}_i^{(\ell+1)}\|_1 = \sum_{i=1}^{m'} \|(\mathbf{A}_j^{(\ell)} - \sum_{k=1,k\neq j}^{m} s_k \mathbf{A}_k^{(\ell)}) \circledast \mathbf{W}_{i,j}^{(\ell+1)}\|_1$$

[0058] 상기 식을 바탕으로 하여, 목적식을 최소화하는 것은 하기식을 최소화하는 것으로 문제를 줄일 수 있다:

[0059] $$\|(\mathbf{A}_j^{(\ell)} - \sum_{k=1,k\neq j}^{m} s_k \mathbf{A}_k^{(\ell)})\|_1$$

[0060] 여기서 $A_j^{(l)} \approx \sum_{k=1,k\neq j}^{m} s_k A_k^{(l)}$ 를 만족하는 s_k를 찾아야 한다. 앞서 배치 정규화와 활성화 레이어를 고려하지 않았기 때문에 쉽게, 활성화맵을 특징맵으로 근사할 수 있게 된다$(A^{(l)} \approx Z^{(l)})$. 따라서 위의 식은 하기식으로 나타낼 수 있다:

[0061] $$\|\mathbf{A}_j^{(\ell)} - \sum_{k=1,k\neq j}^{m} s_k \mathbf{A}_k^{(\ell)}\|_1 = \|(\mathbf{A}^{(\ell-1)} \circledast \mathbf{W}_j^{(\ell)}) - \sum_{k=1,k\neq j}^{m} s_k \times (\mathbf{A}^{(\ell-1)} \circledast \mathbf{W}_k^{(\ell)})\|_1$$
$$= \|(\mathbf{A}^{(\ell-1)} \circledast (\mathbf{W}_j^{(\ell)} - \sum_{k=1,k\neq j}^{m} s_k \times \mathbf{W}_k^{(\ell)})\|_1.$$

[0062] 여기서 $A^{(l-1)}$는 데이터없이 구할 수 없는 값이므로, 나머지 부분$W_j^{(l)} - \sum_{k=1,k\neq j}^{m} s_k W_k^{(l)}$ 을 최소화하는 s_k를 찾아야 하며, 이 부분을 잔차 오류 ε(Residual Error)라 한다.

[0063] 여기서 ε는 사전학습된 합성곱 신경망의 네트워크의 파라미터만으로 최소화된다.

[0064] 배치 정규화 레이어를 고려하게되면, 활성화맵은 특징맵이 배치 정규화 레이어 N를 통과한 값으로 나타낼 수 있다$A^{(l)} \approx N(Z^{(l)})$. 그리고 $\|A_j^{(l)} \approx \sum_{k=1,k\neq j}^{m} s_k A_k^{(l)}\|_2$ 은 다음과 같이 표현할 수 있다:

[0065] 정리1.

[0066] $$\|\frac{\gamma_j}{\sigma_j}(\mathbf{A}^{(\ell-1)} \circledast \boldsymbol{\varepsilon}) + \frac{\gamma_j}{\sigma_j}\{\sum_{k=1,k\neq j}^{m} s_k \frac{\sigma_j}{\gamma_j}\frac{\gamma_k}{\sigma_k}(\mu_k - \frac{\sigma_k}{\gamma_k}\beta_k) - \mu_j + \frac{\sigma_j}{\gamma_j}\beta_j\}\|_1$$

[0067] 여기서, μ, σ, β은 배치 정규화 파라미터 이다.

[0068] 배치정규화를 고려한 s_k는 $s_k\frac{\sigma_j}{\gamma_j}\frac{\sigma_k}{\gamma_k}$로 대체될 수 있다. 그리고 상기 식의 두 번째 항을 배치 정규화 오류 $\mathcal{B}$(Batch Normalization Error)로 나타낸다.

[0069] 정의2.

[0070] $$\mathcal{B} = \frac{\gamma_j}{\sigma_j}\{\sum_{k=1,k\neq j}^{m} s_k \frac{\sigma_j}{\gamma_j}\frac{\gamma_k}{\sigma_k}(\mu_k - \frac{\sigma_k}{\gamma_k}\beta_k) - \mu_j + \frac{\sigma_j}{\gamma_j}\beta_j\}$$

[0071] 마지막으로, 활성화 레이어F를 고려하면, 활성화맵은 특징맵이 배치 정규화와 활성화 레이어를 통과한 값으로 나타낼 수 있다$A^{(l)} \approx F(N(Z^{(l)}))$. 그리고 $\|A_j^{(l)} \approx \sum_{k=1,k\neq j}^{m} s_k A_k^{(l)}\|_2$ 는 다음과 같이 나타낼 수 있다:

[0072] 이론1.

[0073] $$\|\frac{\gamma_j}{\sigma_j}(\mathbf{A}^{(\ell-1)} \circledast \boldsymbol{\varepsilon}) + \mathcal{B} + \mathcal{R}\|_1$$

[0074] 여기서, $\mathcal{R} = \sum_{k=1,k\neq j}^{m} s_k \min(0, N(Z_k^{(l)})) - \min(0, N(Z_k^{(l)}))$는 활성화 오류(Activation Error)로 나타낸다.

[0075] 배치 정규화와 활성화 레이어를 고려하는 확장을 통해서, 최종적으로 재구성 오류는 다음 위의 식과 같이 정의할 수 있다. 위 식으로부터 s_k는 잔차 오류($\boldsymbol{RE},\boldsymbol{\mathcal{E}}$), 배치정규화 오류($\boldsymbol{BE},\boldsymbol{\mathcal{B}}$), 활성화 오류($\boldsymbol{AE}.\boldsymbol{\mathcal{R}}$)를 최소화하는 값으로 정해져야 한다. 하지만 활성화 오류를 줄이기 위해서는 특징맵의 모든 요소에 대해 모든 경우를 고려해야 하고, 이는 데이터없이는 불가능하다. 따라서, 다음 아래의 손실함수를 최소화하는 s_k를 찾고자 한다:

[0076] $$\mathcal{L}_{re} = \|\boldsymbol{\mathcal{E}}\|_2^2 + \lambda_1\|\boldsymbol{\mathcal{B}}\|_2^2 + \lambda_2\|\mathbf{s}\|_2^2$$

[0077] 여기서, $\boldsymbol{s} = [s_1 \cdots s_{j-1} s_{j+1} \cdots s_m]$.

[0078] 위 손실함수를 만족하는 s_k는 다음과 같이 닫힌 형태(closed form)을 얻어 낼 수 있다.

[0079] 이론2.

[0080] $$\boldsymbol{s} = [X^T X + \lambda_1 \frac{\gamma_j^2}{\sigma_j^2}\boldsymbol{p}\boldsymbol{p}^T + \lambda_2 I]^{-1}[X^T \boldsymbol{y} + \frac{\lambda_1 \gamma_j}{\sigma_j}(\frac{\mu_j \gamma_j}{\sigma_j} - \beta_j)\boldsymbol{p}],$$

[0081] 여기서, $X = [\frac{\sigma_j}{\gamma_j}\frac{\gamma_1}{\sigma_1}\boldsymbol{f_1} \cdots \frac{\sigma_j}{\gamma_j}\frac{\gamma_{j-1}}{\sigma_{j-1}}\boldsymbol{f_{j-1}}\ \frac{\sigma_j}{\gamma_j}\frac{\gamma_{j+1}}{\sigma_{j+1}}\boldsymbol{f_{j+1}} \cdots \frac{\sigma_j}{\gamma_j}\frac{\gamma_m}{\sigma_m}\boldsymbol{f_m}]$, $\boldsymbol{f_i}$는 l번째레이어의 i번째 필터의 벡터, $\boldsymbol{y}$는 l번째 레이어의 j번째 필터의 벡터, $\boldsymbol{p} = [p_1 \cdots p_{j-1}\ p_{j+1} \cdots p_m]^T$ 및 $p_i = \frac{\sigma_j}{\gamma_j}\frac{\gamma_i}{\sigma_i}(\mu_i - \frac{\sigma_i}{\gamma_i}\beta_i)$이다.

[0083] 도 5는 본 발명의 일 실시예에 따른 데이터와 학습없이 프루닝으로 압축된 신경망의 성능을 복구하기 위한 알고리즘을 나타내는 도면이다.

[0084] 상기 손실 함수를 최소화하는 s_k는 제거되는 필터에 대해 각각 위의 닫힌 형태로부터 얻게 되고, 이를 전달 행렬의 요소값으로 정하고 이를 사전학습된 합성곱 신경망의 필터와 2-모드곱(2-mode product) 연산을 수행하게 되면, 데이터와 학습없이 프루닝으로 압축된 신경망의 성능을 복구할 수 있다. 이에 대한 알고리즘을 도 5에 도시하였다.

[0085] 합성곱 신경망을 프루닝하는 레이어에 대해서, 레이어 각각의 전달 행렬 $\boldsymbol{S}$를 초기화한다(2번째 줄). 그리고 각 레이어에서 보존되는 필터에 대해서는 전달행렬에 대해서는 보존되는 필터에 해당하는 인덱스만 1이고 나머지가 0인 원핫 벡터(one-hot vector)를 입력한다(4-5번째 줄). 그 이후 제거되는 필터에 대해서는 손실함수를 최소화는 s_k를 찾아 전달 행렬의 요소로 입력한다(6-7번째 줄). 하나의 레이어에 대해서 전달 행렬을 구하게 되면 사전학습된 합성곱 신경망의 파라미터와 전달 행렬과의 2-모드곱 를 수행하여 프루닝으로 손실된 성능을 복구시킨다. 이러한 과정을 프루닝을 수행하는 레이어에 대해서 모두 수행한다.

[0086] 표 1 내지 표 4는 본 발명에서 제안한 데이터와 학습없이 프루닝기법으로 압축된 사전학습된 합성곱 신경망의 성능을 복구한 기법을 적용한 것에 대한 CIFAR와 ImageNet 데이터에 대한 결과이다.

[0087] <표 1>

[0088]

VGG-16 (Acc. 93.7)												
Criterion	L2-norm			L2-GM			L1-norm			Random		
Pruning Ratio	Ours	NM	Prune	Ours	NM	Prune	Ours	NM	Prune	Ours	NM	Prune
10%	**92.04**	91.93	89.43	**92.15**	91.82	88.83	91.85	**91.97**	89.85	**88.05**	86.52	74.87
20%	**87.84**	87.24	71.77	**87.97**	86.42	72.04	**89.51**	86.93	73.64	**75.24**	63.33	37.19
30%	**83.25**	76.91	56.95	**82.83**	75.58	52.40	**81.80**	75.01	49.79	45.67	**46.86**	12.81
40%	**66.81**	54.32	31.74	**67.19**	52.24	30.46	**56.63**	52.54	17.78	**31.34**	26.79	10.00
50%	**45.71**	32.58	12.37	**49.46**	32.13	11.71	**43.32**	27.87	10.43	**21.56**	18.08	10.00
Average Acc.	**75.13**	68.60	52.45	**75.92**	67.64	51.09	**72.62**	66.86	48.30	**52.37**	48.32	28.97

[0089] <표 2>

ResNet-50 (Acc. 78.82)												
Criterion	L2-norm			L2-GM			L1-norm			Random		
Pruning Ratio	Ours	NM	prune	Ours	NM	prune	Ours	NM	prune	Ours	NM	prune
10%	**78.14**	77.28	75.14	**78.01**	76.92	74.49	**78.25**	77.21	75.07	**76.53**	72.46	59.32
20%	**76.15**	72.73	63.39	**76.36**	72.32	63.54	**75.71**	72.24	61.84	**73.29**	59.44	19.27
30%	**73.29**	64.47	39.96	**72.97**	64.01	39.01	**72.07**	63.07	35.77	**69.05**	40.42	3.25
40%	**65.21**	46.40	15.32	**65.78**	46.17	13.14	**62.64**	45.98	12.59	**59.49**	20.89	2.59
50%	**52.61**	25.98	5.22	**54.04**	23.44	4.32	**49.07**	21.98	4.25	**43.51**	8.77	2.77
Average Acc.	**69.08**	57.37	39.81	**69.43**	56.57	38.90	**67.55**	56.10	37.90	**64.37**	40.40	17.44

[0090]

[0091] <표 3>

ResNet34 - ImageNet (Acc. 73.27)												
Criterion	L2-norm			L2-GM			L1-norm			Random		
Pruning Ratio	Ours	NM	Prune	Ours	NM	Prune	Ours	NM	Prune	Ours	NM	Prune
10%	**69.22**	66.96	63.74	**69.12**	66.59	61.76	**69.07**	66.30	62.05	**65.90**	64.75	52.97
20%	**62.49**	55.70	42.81	**62.27**	55.39	43.45	**61.18**	53.95	40.61	**50.64**	48.40	18.62
30%	**47.59**	39.22	17.02	**49.43**	37.97	15.74	**46.88**	35.56	12.58	22.92	**23.69**	1.35
Average Acc.	**59.77**	53.96	41.19	**60.27**	53.32	40.32	**59.04**	51.94	38.41	**46.49**	45.61	19.10

[0092]

[0093] <표 4>

ResNet101 - ImageNet (Acc. 77.31)												
Criterion	L2-norm			L2-GM			L1-norm			Random		
Pruning Ratio	Ours	NM	Prune	Ours	NM	Prune	Ours	NM	Prune	Ours	NM	Prune
10%	**74.59**	72.36	68.90	**74.54**	72.38	68.38	**74.19**	72.05	68.33	**59.78**	57.13	46.55
20%	**68.47**	61.42	45.78	**69.03**	61.02	46.25	**68.52**	60.57	44.57	**34.85**	15.78	6.83
30%	**55.51**	37.38	10.32	**56.65**	33.74	7.65	**56.11**	36.43	9.32	**13.27**	2.31	0.60
Average Acc.	**66.19**	57.05	41.67	**66.74**	55.71	40.76	**66.27**	56.35	40.74	**35.97**	25.07	17.99

[0094]

[0096] 실험에 사용된 네트워크 프루닝 기준은 합성곱 필터의 L1-norm[6], L2-norm[7], L2-GM[8]를 이용하였으며 제안된 방법이 무작위로 프루닝된 경우에도 강건함(robustness)를 보이기 위해 무작위(Random) 기준을 이용하여 실험하였다. 각 표에서의 가장 위에는 사전학습된 합성곱 신경망과 데이터에 대한 성능을 나타낸다. 실험 비교군으로 NM은 원투원 보상 방법이며, Prune은 프루닝 이후 재학습을 하지 않은 압축된 신경망의 성능을 나타낸다. 표 1 내지 표 4를 통해, 제안된 방법이 다른 비교 실험군보다 다양한 데이터와 사전학습된 합성곱 신경망에서 평균적으로 높은 성능을 보임을 확인할 수 있다.

[0098] 도 6은 본 발명의 일 실시예에 따른 합성곱 신경망의 성능 복구 방법과 NM과의 잔차 오류(RE), 배치 정규화 오류(BE), 가중화된 평균 재구성 오류(WARE)를 나타낸 도면이다.

[0099] 도 4는 제안된 방법과 NM과의 잔차 오류(RE), 배치 정규화 오류(BE), 가중화된 평균 재구성 오류(WARE)를 나타낸 것으로, 가중화된 평균 재구성 오류는 학습 데이터 1000개를 이용하여 제안된 방법으로 복구된 신경망과 NM에서 제안한 방법과의 재구성 오류의 평균과의 비교를 하였다. 도 4에서 보이듯이, 잔차 오류는 제안된 방법이 낮은 오류값을 가진다. 이러한 결과가 나타난 이유는 NM은 제거되는 필터에 대해 보존되는 하나의 필터에 제거되는 필터의 정보를 전달하는 반면, 제안된 방법은 제거되는 필터에 대해 보존되는 필터들의 간의 관계 고려하기 때문에 더 낮은 잔차 오류를 가질 수 밖에 없다. 그리고 배치 정규화 오류와 가중화된 평균 재구성 오류가 신경망의 초반 레이어에서는 비슷한 값을 가지는 이유는 도 1에서와 같이 유사한 필터가 초반 레이어에서 더 많이 존재하기 때문이다. 하지만 후반 레이어에 갈수록 NM의 오류는 제안한 방법보다 더 높은 것을 확인할 수 있다. 이를 통해, 제안된 방법은 레이어 간의 변동이 NM에 비해 작으며 3가지 오류에 대해 더 낮은 값을 갖도록 유지시키는 것을 확인할 수 있었다.

[0101] 이상에서 설명된 장치는 하드웨어 구성요소, 소프트웨어 구성요소, 및/또는 하드웨어 구성요소 및 소프트웨어 구성요소의 조합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치 및 구성요소는, 예를 들어, 프로세서, 콘트롤러, ALU(arithmetic logic unit), 디지털 신호 프로세서(digital signal processor), 마이크로컴퓨터, FPA(field programmable array), PLU(programmable logic unit), 마이크로프로세서, 또는 명령(instruction)을 실행하고 응답할 수 있는 다른 어떠한 장치와 같이, 하나 이상의 범용 컴퓨터 또는 특수 목적 컴퓨터를 이용하여 구현될 수 있다. 처리 장치는 운영 체제(OS) 및 상기 운영 체제 상에서 수행되는 하나 이상의 소프트웨어 애플리케이션을 수행할 수 있다. 또한, 처리 장치는 소프트웨어의 실행에 응답하여, 데이터를

접근, 저장, 조작, 처리 및 생성할 수도 있다. 이해의 편의를 위하여, 처리 장치는 하나가 사용되는 것으로 설명된 경우도 있지만, 해당 기술분야에서 통상의 지식을 가진 자는, 처리 장치가 복수 개의 처리 요소(processing element) 및/또는 복수 유형의 처리 요소를 포함할 수 있음을 알 수 있다. 예를 들어, 처리 장치는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 콘트롤러를 포함할 수 있다. 또한, 병렬 프로세서(parallel processor)와 같은, 다른 처리 구성(processing configuration)도 가능하다.

[0102] 소프트웨어는 컴퓨터 프로그램(computer program), 코드(code), 명령(instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성요소(component), 물리적 장치, 가상 장치(virtual equipment), 컴퓨터 저장 매체 또는 장치에 구체화(embody)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 하나 이상의 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.

[0103] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(magnetic media), CD-ROM, DVD와 같은 광기록 매체(optical media), 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical media), 및 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다.

[0104] 이상과 같이 실시예들이 비록 한정된 실시예와 도면에 의해 설명되었으나, 해당 기술분야에서 통상의 지식을 가진 자라면 상기의 기재로부터 다양한 수정 및 변형이 가능하다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다.

[0105] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한다.

[0107] <참고자료>

[0108] [1] J. Luo, J. Wu, and W. Lin. Thinet: A filter level pruning method for deep neural network compression. In IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017, pages 5068-5076. IEEE Computer Society, 2017

[0109] [2] J. Luo and J. Wu. Neural network pruning with residual-connections and limited-data. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA,USA, June 13-19, 2020, pages 1455-1464. IEEE, 2020.

[0110] [3] S. Srinivas and R. V. Babu. Data-free parameter pruning for deep neural networks. In X. Xie,M. W. Jones, and G. K. L. Tam, editors, Proceedings of the British Machine Vision Conference 2015, BMVC 2015, Swansea, UK, September 7-10, 2015, pages 31.1-31.12. BMVA Press, 2015.

[0111] [4] W. Kim, S. Kim, M. Park, and G. Jeon. Neuron merging: Compensating for pruned neurons. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020.

[0112] [5] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. SIAM Rev., 51(3):455- 337 500, 2009

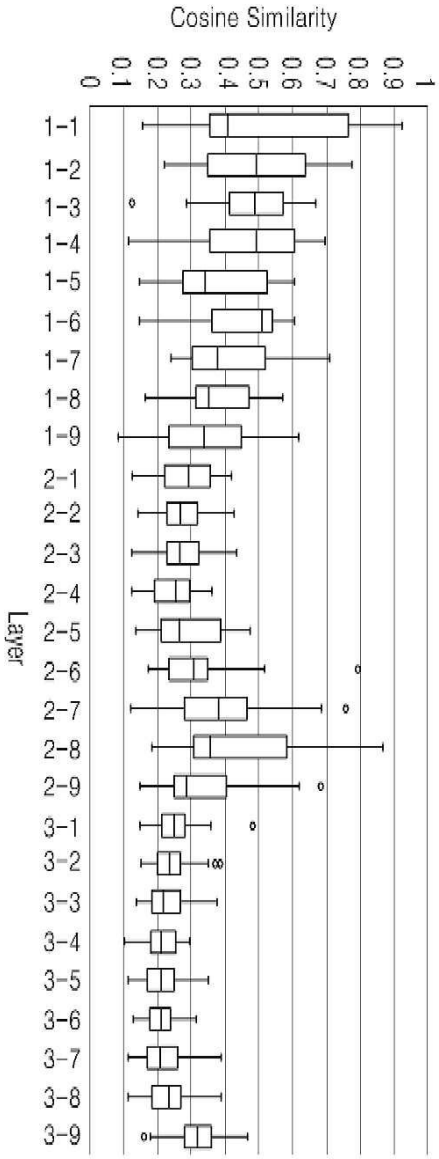
[0113] [6] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. In 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net, 2017.

[0114] [7] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang. Soft filter pruning for accelerating deep convo315 lutional neural networks. In J. Lang, editor, Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden, pages 2234-2240. ijcai.org, 2018

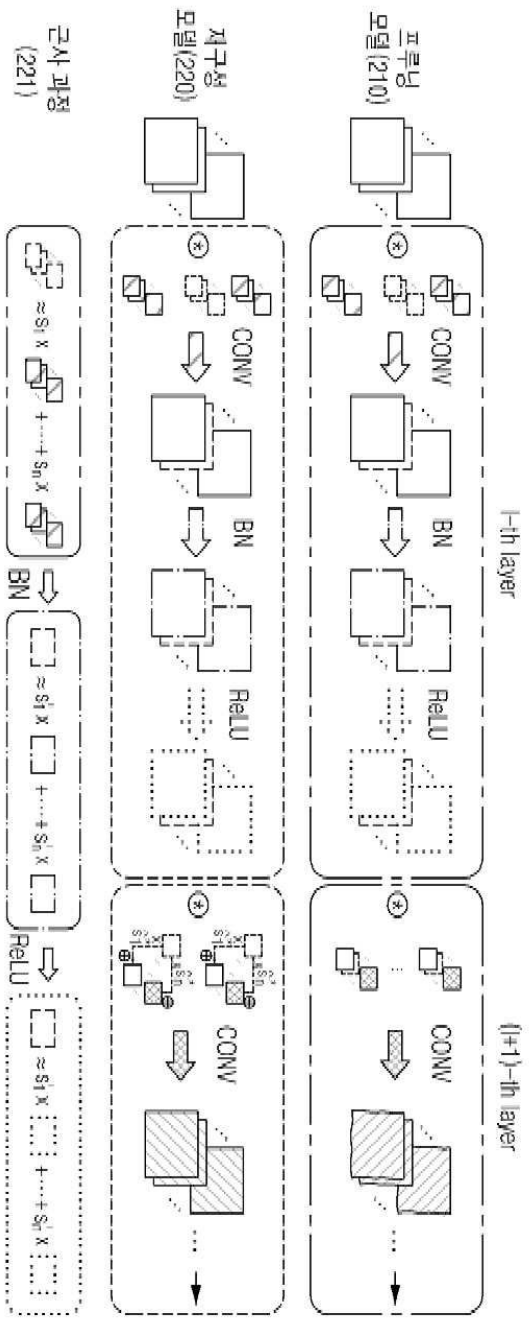
[0115] [8] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019, pages 4340-4349. Computer Vision Foundation / IEEE, 2019.

도면

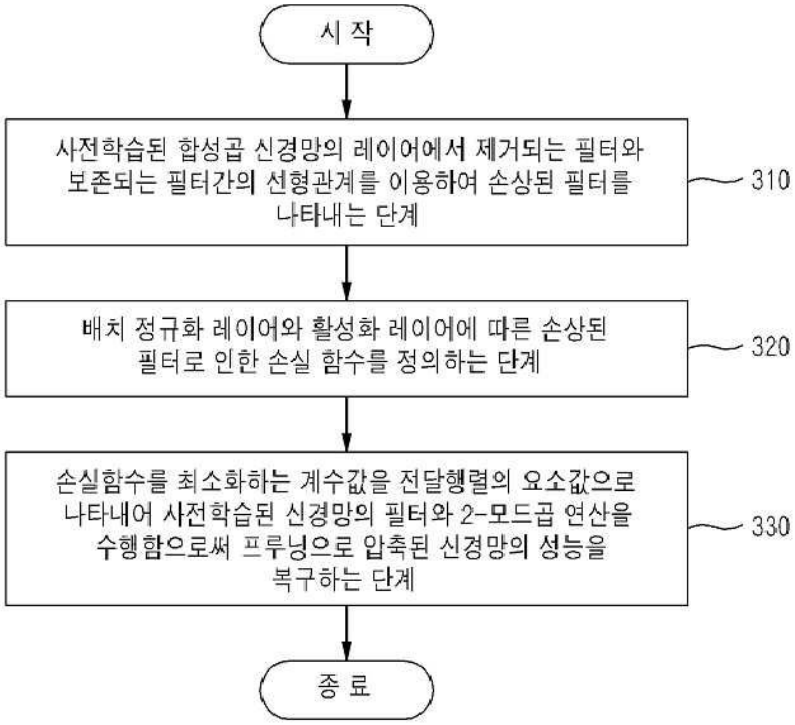
도면1



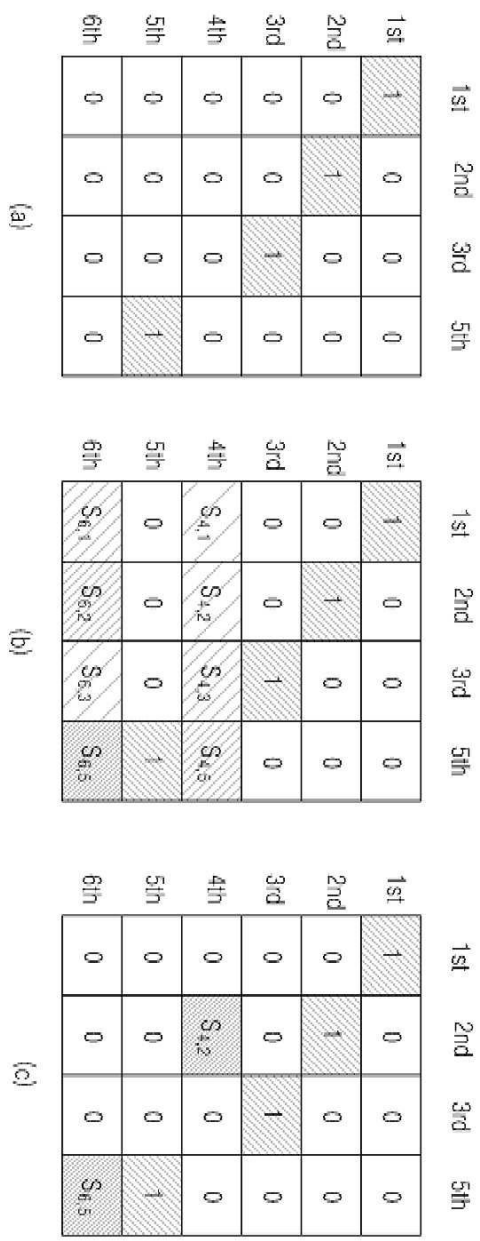
도면2


프루닝 모델(210)
재구성 모델(220)
근사 과정 (221)
CONV
BN
ReLU
l-th layer
(l+1)-th layer

도면3


시 작
사전학습된 합성곱 신경망의 레이어에서 제거되는 필터와 보존되는 필터간의 선형관계를 이용하여 손상된 필터를 나타내는 단계
310
배치 정규화 레이어와 활성화 레이어에 따른 손상된 필터로 인한 손실 함수를 정의하는 단계
320
손실함수를 최소화하는 계수값을 전달행렬의 요소값으로 나타내어 사전학습된 신경망의 필터와 2-모드곱 연산을 수행함으로써 프루닝으로 압축된 신경망의 성능을 복구하는 단계
330
종 료

도면4


(a)
1st 2nd 3rd 5th
1st 1 0 0 0
2nd 0 1 0 0
3rd 0 0 1 0
4th 0 0 0 0
5th 0 0 0 1
6th 0 0 0 0
(b)
1st 2nd 3rd 5th
1st 1 0 0 0
2nd 0 1 0 0
3rd 0 0 1 0
4th S4,1 S4,2 S4,3 S4,5
5th 0 0 0 1
6th S6,1 S6,2 S6,3 S6,5
(c)
1st 2nd 3rd 5th
1st 1 0 0 0
2nd 0 1 0 0
3rd 0 0 1 0
4th 0 S4,2 0 0
5th 0 0 0 1
6th 0 0 0 S6,5

도면5

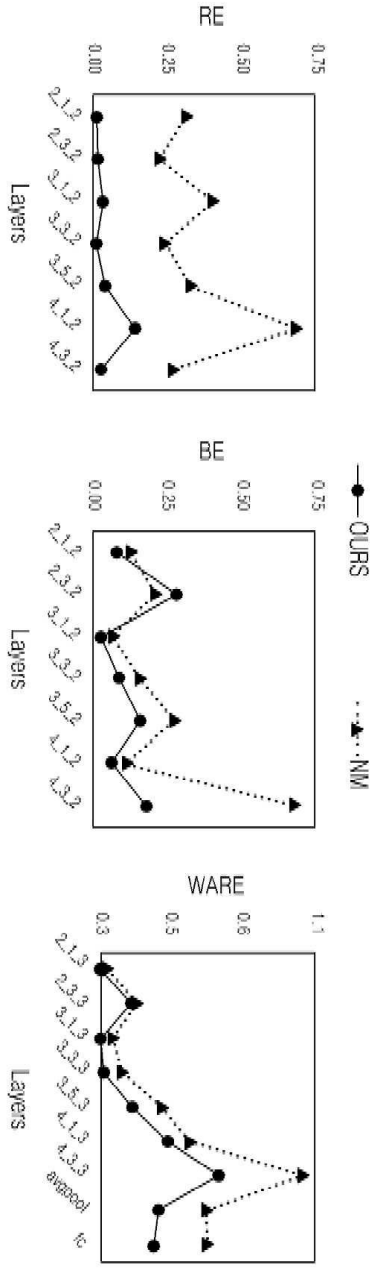
Algorithm 1 LBYL method

Input: $\Theta = [\mathbf{W}^{(1)}, ..., \mathbf{W}^{(L)}]$:= Original model

Output: $\hat{\Theta} = [\hat{\mathbf{W}}^{(1)}, ..., \hat{\mathbf{W}}^{(L)}]$:= Pruned & restored model

for *each layer* $\ell \in [1, L]$ *that has been pruned* **do**
  $\mathcal{S}^{(\ell)} \leftarrow$ a zero-valued matrix
  for *each filter* $\mathbf{W}_i^{(\ell)}$ *s.t.* $i \in [1, m]$ **do**
    if $\mathbf{W}_i^{(\ell)}$ *is not pruned* **then**
      $\mathcal{S}_{i,k}^{(\ell)} = \begin{cases} 1 & \text{if } i = i'_k \\ 0 & \text{otherwise} \end{cases}$
    else
      $\mathcal{S}_{i,:}^{(\ell)} \leftarrow \arg\min_{\mathbf{s}} \|\mathcal{E}\|_2^2 + \lambda_1 \|\mathcal{B}\|_2^2 + \lambda_2 \|\mathbf{s}\|_2^2$ `// obtained by Theorem 2`
  $\hat{\mathbf{W}}^{(\ell+1)} = \mathbf{W}^{(\ell+1)} \times_2 \mathcal{S}^{(\ell)T}$
return $\hat{\Theta}$

도면6


RE
0.00
0.25
0.50
0.75
2,1,2
2,3,2
3,1,2
3,3,2
3,5,2
4,1,2
4,3,2
Layers
BE
0.00
0.25
0.50
0.75
2,1,2
2,3,2
3,1,2
3,3,2
3,5,2
4,1,2
4,3,2
Layers
OURS
NM
WARE
0.3
0.5
0.6
1.1
2,1,3
2,3,3
3,1,3
3,3,3
3,5,3
4,1,3
4,3,3
average
Layers