Journal of Convergence for Information Technology
Vol. 10. No. 8, pp. 23-34, 2020
e-ISSN 2586-4440
DOI : https://doi.org/10.22156/CS4SMB.2020.10.08.023

심층신경망의 더블 프루닝 기법의 적용 및 성능 분석에 관한 연구

이선우[1], 양호준[2], 오승연[2], 이문형[2], 권장우[3*]
[1]인하대학교 전기컴퓨터공학과 학생, [2]인하대학교 컴퓨터공학과 학생, [3]인하대학교 컴퓨터공학과 교수

Application and Performance Analysis of Double Pruning Method for Deep Neural Networks

Seon-Woo Lee[1], Ho-Jun Yang[2], Seung-Yeon Oh[2], Mun-Hyung Lee[2], Jang-Woo Kwon[3*]
[1]Student, Electric Computer Engineering, Inha University
[2]Student, Computer Engineering, Inha University
[3]Professor, Computer Engineering, Inha University

요 약 최근 인공지능 딥러닝 분야는 컴퓨팅 자원의 높은 연산량과 가격문제로 인해 상용화에 어려움이 존재했다. 본 논문은 더블 프루닝 기법을 적용하여 심층신경망 모델들과 다수의 데이터셋에서의 성능을 평가하고자 한다. 더블 프루닝은 기본의 네트워크 간소화(Network-Slimming)과 파라미터 프루닝(Parameter-Pruning)을 결합한다. 이는 기존의 학습에 중요하지 않는 매개변수를 절감하여 학습 정확도를 저해하지 않고 속도를 향상시킬 수 있다는 장점이 있다. 다양한 데이터셋 학습 이후에 프루닝 비율을 증가시켜, 모델의 사이즈를 감소시켰다. NetScore 성능 분석 결과 MobileNet-V3가 가장 성능이 높게 나타났다. 프루닝 이후의 성능은 Cifar 10 데이터셋에서 깊이 우선 합성곱 신경망으로 구성된 MobileNet-V3이 가장 성능이 높았고, 전통적인 합성곱 신경망으로 이루어진 VGGNet, ResNet또한 높은 폭으로 성능이 증가함을 확인하였다.

주제어 : 모델압축, 모델 경량화, 딥러닝, 프루닝, 네트워크 간소화

Abstract Recently, the artificial intelligence deep learning field has been hard to commercialize due to the high computing power and the price problem of computing resources. In this paper, we apply a double pruning techniques to evaluate the performance of the in-depth neural network and various datasets. Double pruning combines basic Network-slimming and Parameter-prunning. Our proposed technique has the advantage of reducing the parameters that are not important to the existing learning and improving the speed without compromising the learning accuracy. After training various datasets, the pruning ratio was increased to reduce the size of the model.We confirmed that MobileNet-V3 showed the highest performance as a result of NetScore performance analysis. We confirmed that the performance after pruning was the highest in MobileNet-V3 consisting of depthwise seperable convolution neural networks in the Cifar 10 dataset, and VGGNet and ResNet in traditional convolutional neural networks also increased significantly.

Key Words : Model Compression, Model Light Weight, Deep Learning, Pruning, Network-Slimming

*This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2017-0-01642) supervised by the IITP(Institute for Information & communications Technology Promotion)
*Corresponding Author : Jnag-Woo Kwon(jwkwon@inha.ac.kr)
Received June 10, 2020 Revised August 4, 2020
Accepted August 20, 2020 Published August 28, 2020

1. 서론

최근의 딥러닝에 대한 연구는 소형 스마트폰이나 소형 디바이스 들에 적용이 가능한 딥러닝 모델들의 최적화에 대한 연구가 많이 되고 있으나 여전히 많은 문제점이 있다. 기존의 뉴럴네트워크 모델들은, 정확도에 초점[1-4]을 맞춘다거나, 속도 측면[5-10]에서 많은 이점을 가져오고자 했다면, 최근의 딥러닝은 정확도와 속도에 대한 이점을 모두 가져와 이러한 문제점을 해결하고자 하였다[5, 11]. 이와 프루닝(Pruning) 기법은 기존의 네트워크의 성능을 거의 유지하면서도 연산량을 줄이는 후처리 기법으로 제안되었다[12-18]. 프루닝은, 기존의 네트워크의 구조에 어떠한 네트워크를 추가하지 않고, 후처리를 통하여 연산량, 저장공간, 에너지 감소 등의 효과를 내는 다양한 이점이 있다[14].

본 논문에서는 기존의 매개변수 프루닝[13]과 채널 간소화 프루닝 기법[17]을 각각 적용 할때와 동시에 적용할때의 성능결과를 보고 데이터셋에 따른 성능을 비교하였다. 이와 더불어 속도를 우선시 하는 깊이별 분리 합성곱 신경망[5-7] 3종과 일반적으로 사용되는 합성곱 신경망 기반의 모델 2종[8, 10]종에 각각 적용하였다. 마지막으로 모델의 정확도와 속도측면을 동시에 고려하는 평가지표를 입력크기와 분류 레이블의 수가 각기다른 4종의 데이터셋에 도입하여 비교분석을 하였다.

본 논문은 다음과 같이 구성된다. 2장에서는 딥러닝 경량화를 위한 대표적인 합성곱 신경망의 경량화 모델에 대해 설명하고, 다양한 프루닝 기법을 소개하고자 한다. 3장에서는 본 논문에서 사용된 데이터셋과 사용되어지는 딥러닝 모델, 프루닝 기법 등을 설명하고 본 학습에서 사용되는 방법을 제안하고자 한다. 4장에서는 본 논문에서 제안하는 방법으로 실험하여 나온 결과와 모델 성능 측정방법에 대하여 서술 및 성능을 비교하고자 한다. 5장에서는 결론을 맺으며 향후 연구에 대하여 제안하고자 한다.

2. 관련연구

2.1 경량 딥러닝 모델

심층신경망 최적 구조에 관한 연구는 그동안 많이 이루어져 왔다. 일반적으로 심층신경망은 정확도와 속도에서의 거래(trade-off)가 있는데 정확도를 최대한 보전하면서 속도를 효율적으로 올리기 위한 연구가 상용화에 있어서 중요한 이슈 중 하나이다.심층신경망의 합성곱 신경망은 기본적으로 행렬 연산이기 때문에 많은 연산이 필요하다. 초기에 이를 극복하기 위하고자 다양한 시도들이 이루어졌다.

초기에 1×1 합성곱 신경망을 이용하여 서로비슷한 은닉층(Hidden Layer)을 줄임으로써, 효과적으로 연산량을 줄이는 연구가 있었다.[8],그 이후 합성곱 신견망의 기본적인 연산 구조를 변환시키는 깊이별 분리 합성곱 신경망(Depthwise Seperable Convolution) 등과 같은 연산을 통하여 연산량을 줄이는 방법이 성공적으로 적용되었다[5-7, 10]. 특징지도(FeatureMap)을 효율적으로 뽑아낼 수 있도록 하나의 입력에 대해서 Group Convolution을 적용하여 특징점을 효율적으로 추출하려는 시도도 있었다[9]. 이와 더불어 기존 모델의 최적의 해상도(Resolution scaling), 깊이(Depth scaling), 핵의 크기(Kernel Size), 필터의 개수(width scaling)에 따른 최적 구조를 제안[19, 20]을 하였다.

2.2 프루닝 기법

프루닝은 경량화 기법의 하나로, 학습 이후 각 심층신경망 층의 상대적으로 불필요한 매개변수를 제거함으로써, 딥러닝 성능의 저하를 최소화하면서 성공적인 정확도를 가져오는 방법으로 사용되어왔다. Fig. 1는 프루닝의 기본적인 방법으로써, 학습 이후 프루닝을 한 뒤, 다시 미세조정(Fine-tuning)을 거치면서 학습이 된다.

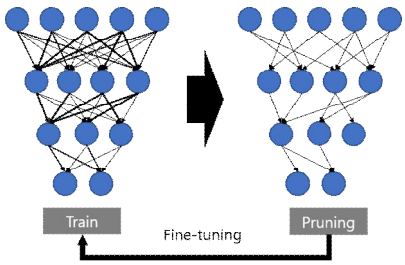


Fig. 1. Basic Pruning Principle Structure in Deep Learning

프루닝은 기본적으로 모델의 은닉층 필터를 제거하는 방법[13]과, 은닉층 안의 값들을 0으로 만드는 방법[17,18,21]이 있다. 최근의 프루닝 기법은, 학습하기n 전에 프루닝 이후의 매개변수들을 그대로 적용시키면 모델의 성능이 저하되는 경우가 있는데, 이는 딥러닝 모델이 아직 오류가 전역 최소(Global Minimum)로

도달하지 못한 상태에서 학습을 시작하면, 데이터의 허용 용량(Capacity)이 부족하여 발생하는 상태이다. 이런 문제를 극복하고자 프루닝을 학습이 다 끝나지 않은 상태에서 약간의 에폭(Epoch) 이후 프루닝을 일정부분 해주고, 다시 몇 번의 에폭을 반복하는 방법[23]과 수식을 통한 중요 뉴런을 근사 계산하여 해당중요 뉴런만 남기고 프루닝 하는 방법[16]이 있다.

3. 실험 데이터 및 제안하는 방법

3.1 실험데이터

실험을 적용하고자 하는 학습 및 테스트 데이터로는 Cifar10, Cifar100[24], FER2013[25], FER+[26]라는 총 4가지 데이터로 학습을 시켰으며, 각 데이터셋 마다 훈련, 테스트, 레이블에 대한 정보는 Table 1과 같이 구성된다.

Table 1. Information about the size of the dataset to be applied in this paper, and the number of training and test data

	Train	Test	Label	Input Size
Cifar10	50,000	10,000	10	$32\times32\times3$
Cifar100	50,000	10,000	100	$32\times32\times3$
Fer2013	28,709	7,178	7	$48\times48\times1$
Fer+	28,561	7,153	8	$64\times64\times1$

3.1.1 Cifar(Canadian Institute For Advanced Research) Dataset

Cifar 10, Cifar 100 은 학습데이터와 테스트 데이터가 동일하지만, 레이블이 다른 데이터로 학습 레이블이 프루닝 결과에 영향을 미치는지 알아보기 위하여 선택 되었다. Cifar 10 데이터셋은, 비행기(airplane), 자동차(automobile), 새(bird), 고양이(cat), 사슴(deer), 개(dog), 개구리(frog), 말(horse), 배(ship), 트럭(truck)으로 이루어진 데이터로 총 60,000개의 데이터로 이루어져 있고, 각 클래스마다 5,000개의 학습 데이터와 1,000개의 테트 데이터로 이루어져 있으며, 클래스별 데이터예시는 Fig. 2와 같다. Cifar 100 데이터셋은 Cifar 10보다 좀 더 세분화된 클래스로 이루어져 있으며, 상위 개념의 슈퍼클래스(Superclass)와 하위의 클래스로 이루어져 있다. 각 클래스별 500개의 학습 이미지와 100개의 테스트 이미지로 이루어져 있다.

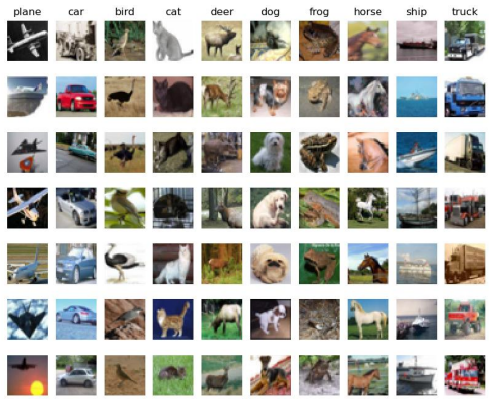


Fig. 3. Cifar 10 dataset example

3.1.2 FER(Facial Emotion Recognition) Dataset

FER2013(Facial Emotion Recognition 2013)과 FER+ 데이터는 학습 데이터와 클래스는 비슷하지만, 입력 크기가 차이가 있는 데이터로 입력 크기에 따른 프루닝 결과를 비교하기 위하여 적용되었다. 또한, FER 데이터셋은 학습 및 테스트 데이터의 분포가 균일하지 않은 상태이기 때문에 프루닝의 영향도 살펴보고자한다. FER2013 데이터셋 은 $48\times48\times1$ 크기로 구성되었으며, 사람의 정확도는 약 $65\pm5\%$로, 35,887개의 이미지와 총 7개의 레이블(화남(Angry), 역겨움(Disgust), 공포(Fear), 행복(Happy), 슬픔(Sad), 놀람(Surprise), 보통(Neutral)로 이루어져 있다. 이 중 28,709개의 훈련(Train) 데이터와 3,589개의 검증(Validation) 데이터 3,589개의 시험(Test) 데이터로 구성되어 있으며 데이터 레이블 개수 분포는 Fig. 3.으로 구성된다.

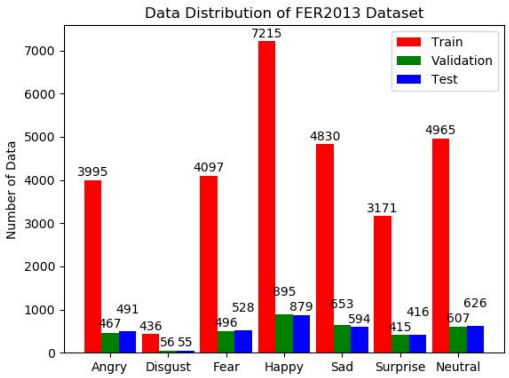


Fig. 4. Data Distribution of FER2013 Dataset

FER+ 데이터셋은 FER2013 데이터셋의 Fig. 4과

같은 부적합 레이블(Noisy Label)을 보안 하고자 10명의 인원이 다시 판단한 뒤에 각자가 생각하는 표정의 종류를 투표하여 표기하였다.

Fig. 5. Noisy Data of FER2013 dataset

기존의 FER2013 데이터 레이블에서 경멸(Contempt) 클래스까지 추가하여 총 8종류의 데이터를 구성하여 다수 투표(Majority Voting), 다중 레이블 학습(Multi-Label Learning), 확률적 표현(Probabilistic Label Drawing), 교차-엔트로피 오차(Cross-entropy loss) 총 4가지 방법으로 레이블링 기법을 적용하여 최적의 레이블을 찾는 데이터 셋을 구성하여 기존 정확도보다 향상되었다[26].

본 논문에서는 Fer+는 4가지 평가방법 중, 교차 엔트로피 손실(Cross Entropy Loss)이 성능대비 표준편차가 적어 [26]에서 우수하여 이를 채용하였고, 그 방법은 아래의 식 (1) 과 같은 방법으로 표현이 가능하며 데이터셋의 분포는 Fig. 5과 같다.

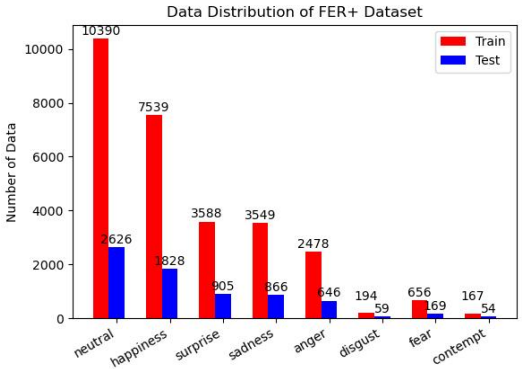


Fig. 6. Diata Distribution of Fer+ Dataset (Cross-Entropy- Loss)

$$L=-\sum_{i=1}^{N}\sum_{k=1}^{8} p_k^i \log q_k^i \quad \cdots\cdots (1)$$

여기서 p는 FER+에서 다시 레이블한 10명의 투표 확률값이고, N은 데이터 개수, q는 제안하는 네트워크에서 심층신경망의 결과값을 의미한다.

3.2 적용 딥러닝 모델

제안하는 프루닝 방법은 3.1 장에서 제안된 학습 데이터에 다수의 딥러닝 모델을 학습 후 매개변수 프루닝과 네트워크 간소화를 차례로 거친 모델로써, 적용하여 실험하고자 하는 대상 딥러닝 신경망은 경량화 목적의 깊이별-분리 합성곱(Depthwise-Separable Convolution) 신경망을 사용한 신경망 모델 3종과 일반적으로 합성곱 신경망이 사용되는 신경망 모델 2종을 적용하였다. 깊이별-분리 합성곱 신경망 기반의 모델 3종은 MobileNet[6], MobileNet-V2[5], MobileNet-V3[4]를 적용하였고, 일반적으로 사용되는 합성곱 신경망 모델은 VGGNet19[8], ResNet18[10] 모델을 적용하였다.

3.2.1 VGGNet

VGGNet모델은 ILSVRC2014에서 오직 3×3크기의 합성곱 신경망을 이용하여 제안한 네트워크로, 구현이 쉽다는 장점있고, 어느정도 성능이 보장된다는 점이 특징이다. 본 논문에서는 VGG19의 합성곱신경망의 매개변수만을 비교를 위하여 마지막 3개의 완전 결합 층(Fully Connected Layer)을 전역 평균 풀링(Global Average Pooling)으로 전환하여 매개변수의 개수를 줄였다.

3.2.2 ResNet

ResNet 모델은 잔차모듈(Residual module)이라는 Skip Connection 기법을 적용하여 은닉층의 개수를 크게하면서도 기울기 소실/폭발 문제를(Gradient Vanishing/Exploding Problem) 해결한 방법이다. 본 논문에서는 VGG와 깊이가 비슷한 ResNet18 네트워크를 적용하였다.

3.2.3 MobileNet-V1

MobileNet-V1은, 깊이별 합성곱(Depth-wise Convolution)[10]에 포인트별 합성곱(Point-wise Convolution)을 결합한 깊이별 분리 합성곱(Depth-wise Separable Convolution)을 제안하여 정확도 대비 속도가 좋은 네트워크로 많이 사용되고 있다. MobileNet은 width multiplier의 비율(α)에 따라서 성능과 속도간의 거래(trade-off)[7]가 있는데, α는

1.0으로 채택하여 실험에 적용하였다.

3.2.4 MobileNet-V2

MobileNet-V2는 ResNet 모델에서 제안하는 잔차 블록(Residual block)에서 병목 구조(Bottle-Neck Structure)을 차용하여 역-잔차블록(Inverted Residual Block)[6]을 제안하여 우수한 성능을 가져온 모델이다. 본 실험에서도 MobileNetV1의 모델에서와 같이 너비 승수(width multiplier)의 비율(α)와 해상도 승수(resolution multiplier) 비율(ρ)에 따라서 성능이 나뉘는데 본 실험에서는 1.0을 선택하여 실험하였다.

3.2.5 MobileNet V3

MobileNet-V3모델은 최근 MNasNet[26]모델과 같은 모델의 구조의 학습모델을 기반으로하여 나온 모델을 수정하고, 모바일 디바이스에 맞도록 swish[27] 손실함수로 수정하여 제안한 모델을 사용하였다. 본 실험에서는, MobileNetV3 모델이 제안한 모델중 큰(Large)모델과 작은(Small) 모델에서 작은 모델을 차용하여 적용하였다. 또한 학습 입력 크기가 작은 관계로 마지막의 드랍아웃(Dropout) 비율을 0.8에서 0.0으로 없앴다.

3.3 하이퍼 파라미터(Hyper Parameter)

본 논문에서 제안하는 네트워크들의 학습 매개변수는 모두 동일하게 배치사이즈는 128로 진행을 하였다. 에폭(Epoch)은 200으로 선정하였고, 학습속도(Learning Rate)는 초기에 0.1로 설정하였다. 학습속도는 각 에폭마다 다른 학습속도를 코사인 어닐링(Cosine Annealing) 주기적으로 반복시켜주는 SGDR스케줄러[30]를 사용하였다. 이를 적용하면, SGD(Stochastic Gradient Descent)를 수행하고 나서 계속해서 학습속도가 0부터 초기에 설정해둔 학습속도로 반복하게 된다. 여기에 학습 속도 감소(Learning Rate Decay)을 0.005를 적용하여 초기 학습속도보다는 조금 낮게 설정하여 마지막에는 0이 되게끔 설정하였고, 학습 결과는 Table 2와 같은 결과를 보였다.

Table 2. Basic performance results for data sets and deep learning networks to be applied before experiment

Dataset	Network	Parameter(Million)	Top-1 Error	Top-5 Error
Cifar10	MobileNet-v1	3.41	9.53	0.30
Cifar10	MobileNet-v2	4.44	9.16	0.33
Cifar10	MobileNet-v3	2.862	23.86	1.61
Cifar10	VGG19	40.06	6.31	0.34
Cifar10	ResNet18	22.165	**5.05**	**0.18**
Cifar100	MobileNet-v1	6.5	32.5	0.10
Cifar100	MobileNet-v2	4.67	31.9	8.92
Cifar100	MobileNet-v3	2.97	57.03	25.71
Cifar100	VGG19	40.11	27.85	10.53
Cifar100	ResNet18	22.21	**24.04**	**7.38**
Fer2013	MobileNet-v1	6.41	31.5	1.55
Fer2013	MobileNet-v2	4.43	31.43	1.0
Fer2013	MobileNet-v3	2.85	39.96	1.76
Fer2013	VGG19	40.06	29.95	2.35
Fer2013	ResNet18	22.16	**28.29**	**2.33**
FerPlus	MobileNet-v1	6.41	17.24	0.01
FerPlus	MobileNet-v2	4.43	17.63	0.81
FerPlus	MobileNet-v3	2.85	21.60	0.75
FerPlus	VGG19	40.06	16.36	0.91
FerPlus	ResNet18	22.17	**16.52**	**1.08**

Table 2.에서 보듯이 실험결과는 모델별로 정확도 측면에서는 ResNet18이 가장 우수한 성능을 보였다. 이는 ResNet18이 Skip-Connection과 같은 기법과 전통적으로 사용되는 합성곱 신경망이 연산량은 많지만 우수한 깊이별-분리 합성곱 신경망보다 우수한 성능을 보이기 때문이다. 또한 연산량이 적은 깊이별-분리 합성곱 신경망 기반의 모델들(MobileNet-v1, v2, v3)은 일반적인 합성곱 신경망에 비하여 성능이 매개변수 수 측면에서는 연산량 대비 오차율이 적은 현상을 보인다.

데이터셋 측면에서는 Cifar100 데이터셋에서는 MobileNet-V3같은 경우는 오차율이 50%가 넘는 현상을 보이는데 이는 분류할 레이블에 비해서 매개변수의 수가 작아 생기는 현상으로 보인다.

3.4 프루닝(Pruning)

본 논문에서 학습에 사용된 방법은 학습이 끝난 신경망 모델에, 매개변수 프루닝(Parameter Pruning)을 하여 프루닝을 선행하여 최적화 작업 후, 네트워크 간소화(Network Slimming)을 점차 높여 나가면서, 신경망의 연산량 절감과 합성곱 필터의 수를 줄임으로써 정확도를 최소화 함으로써 속도를 향상시키고자 하였다.일반적으로 심층신경망에서는 학습이 끝난뒤, 합성곱 필터에서의 뉴런 값들은, 상대적으로 특징을 잘 골라내고, 강도가 강한 뉴런이 있는반면, 특징을 골라내는데 기여하거나 오히려 방해되는 뉴런들이 존재하게 된다. 이러한 필터나 뉴런등의 값들을 0으로 바꿔줌으로써, 연산량을 줄이고 0으로 된 뉴런들은, 이후 연산을 고려하지 않아도 되는 Zero-Skipping[31] 과 같은 효과를 얻을 수 있다.

3.4.1 네트워크 간소화(Network Slimming)

네트워크 간소화은 합성곱 필터의 수를 줄이는 방법으로, Fig. 6과 같이 학습이 진행된 이후 크기조정 계수(scaling factor) 혹은 프루닝 비율에 따라서 필터의 수를 줄이는 방법으로 진행이 된다.

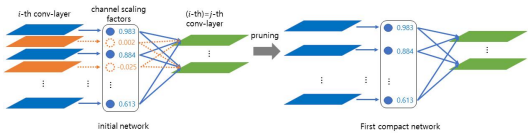


Fig. 7. Example of Network Slimming

크기조정 계수 혹은 프루닝 비율(γ)을 각 은닉층 마다 정하여 진행하고 그 식은 (2)와 같다.(x,y)는 입력값과 예측값을 나타내고, W는 학습 웨이트를 의미하며, 첫 번째, 합은 합성곱 손실값의 오차 함수(Loss Function)을 나타낸다. $g(\bullet)$은 크기조정 계수의 간극성 유도 패널티(sparsity-induced penalty)를 의미[17]한며, λ을 통하여 두 항간의 균형을 맞추는데 본 실험에서는 1로 정하였다. 본 실험에서 $g(\bullet)$은 L1-norm[32]을 선택하였고, 일반적으로 널리 사용되기에 채용하였다.

$$L=\sum_{(x,y)} l(f(x,W),y)+\lambda\sum_{\gamma\in\Gamma} g(\gamma) \quad \cdots\cdots (2)$$

3.4.2 매개변수 프루닝(Parameter Pruning)

본 실험에서 매개변수 프루닝은 각 은닉층에서의 값들을 프루닝 비율에 따라서 마스크 레이어(m)를 두어 그 값을 0으로 만들어 주는 방식을 이용하여 구현하였고, 식 (3)과 같이 표현할 수 있다.

$$f(x, m\odot W) \quad \cdots\cdots (3)$$
$$m\in\{0,1\}$$

이런 매개변수 프루닝을 이용하면, 네트워크 내의 연결이 강한 값들만 남기 때문에, 불필요한 실수 연산을 하지 않으므로 신경망의 매개변수의 수를 줄일 수 있고, 연산량 절감의 효과로 이어져 속도 향상을 할 수 있는 원인이 될 수 있다.

3.4.3 더블 프루닝(Double Pruning)

본 논문에서 제안하고자 하는 더블 프루닝의 방법은 Fig. 7와 같이 나타낼 수 있다.

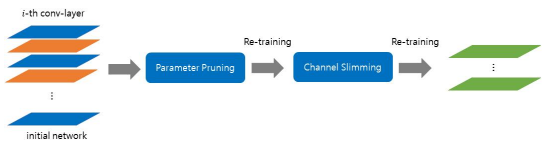


Fig. 8. The architecture of double pruning

제안하는 방법은 학습이 된 신경망 모델에 매개변수 프루닝을 진행한 후에 10 에폭(Epoch) 재학습(re-training)을 실시 후 네트워크 간소화를 진행하여 모델을 재구축

하여 재학습하여 네트워크를 압축하는 방식이다.

본 논문에서 네트워크 간소화 선행하고 매개 변수프루닝을 먼저 수행하였으나, 네트워크 슬라이밍에서의 제거 필터 선정을 할 때, L1-norm의 결과에 상대적으로 필요없는 뉴런들이 영향을 미치므로 매개변수 프루닝을 먼저 선행하여 네트워크 간소화를 수행하였다.

4. 실험 결과

4.1 모델별 성능 평가방법

본 논문에서 성능평가 방법은 프루닝 비율에 따라 변하는 속도 모두를 측정하여 이를 기반으로 심층신경망 모델의 속도와 연산량을 기반으로 모델을 평가하는 검증방법 중 하나인, Net-Score[27]를 측정하고자 하였다. Net-Score는 식 (4)와 같이 표현할 수 있다.

$$ohm(N) = 20\log\left(\frac{\alpha(N)^{\alpha}}{p(N)^{\beta}m(N)^{\gamma}}\right) \cdots\cdots\cdots\cdots (4)$$

네트워크 N의 NetScore $\Omega(N)$는 네트워크의 정확도$a(N)$와네트워크의 파라미터로 단위는 %값을 의미한다. $p(N)$, 곱셈-덧셈의 수(number of multiply-accumulate, MAC)으로 단위는 10억개 단위로 측정한다. $m(N)$은 매개변수의 수로써, 백만개 단위의 개수를 의미하여 이 세 변수를 을 조합하여 나타낸 것이다. 이때, α, β, γ는 상수로써, [27]에서 제안한 값인 $\alpha = 2$, $\beta = 0.5$, $\gamma = 0.5$로 설정하여 평가하였다.

4.2 매개변수 프루닝만 했을 경우

매개변수 프루닝만 적용했을 경우와 이에의한 결과는 Table 3.에 와 같이 나왔다. 전체적으로 정확도 대비 성능이 좋은 모델은 MobileNet-v3였다. 프루닝을 통하여 NetScore가 가장 높은 모델은 MobileNet-V3가 가장 높은 NetScore를 나타내었다. 하지만, 효율성(Efficiency) 측면에서는 MobileNet-V3가 최악의 효율성을 보였다.

데이터셋에 따른 기존 모델 성능대비 프루닝 후의 모델 성능의 향상폭이 가장 큰 모델은 Cifar10, VGGNet19 모델이 133% 성능향상이 있었고, Cifar100 데이터셋은 MobileNet-V1이 159% 성능향상이 있었다. Fer2013, ResNet18모델이 성능 향상이 134%로 가장 높았다. FerPlus 데이터셋에서는 MobileNet-V1이 성능향상 비율이 높았다.

Table 3. NetScore and Effciency result in case of parameter pruning step-by-step (80%, 90%, 95%) after training MobileNet-v1, v2, v3, VGGNet19, ResNet18 models for Cifar10, Cifar100, Fer2013, and Ferplus datasets

Dataset	Network	Non-Prunning	pruned rate: 0.8	pruned rate: 0.9	pruned rate: 0.95	Efficiency
Cifar10	MobileNet-v1	106.86	122.47	128.83	**136.45**	128%
	MobileNet-v2	110.68	126.11	**132.07**	117.07	119%
	MobileNet-v3	**134.33**	142.65	144.41	**145.43**	108%
	VGGNet19	68.64	84.72	**91.48**	11.13	**133%**
	ResNet18	84.70	100.37	106.48	**111.65**	132%
Cifar100	MobileNet-v1	71.65	109.17	**113.62**	95.66	**159%**
	MobileNet-v2	75.57	113.71	**111.48**	26.54	150%
	MobileNet-v3	**109.97**	117.49	119.14	**120.02**	109%
	VGGNet19	58.17	73.93	65.17	-41.06	**127%**
	ResNet18	73.55	87.89	92.11	90.53	125%
Fer2013	MobileNet-v1	87.48	103.13	**109.27**	101.74	125%
	MobileNet-v2	91.30	106.67	108.35	78.49	119%
	MobileNet-v3	**119.41**	127.91	129.71	**130.87**	110%
	VGGNet19	48.97	**64.72**	62.97	24.34	132%
	ResNet18	65.61	81.11	86.36	**87.80**	**134%**
Ferplus	MobileNet-v1	66.30	105.02	**111.36**	102.26	**168%**
	MobileNet-v2	69.86	108.55	**112.77**	92.82	161%
	MobileNet-v3	**127.43**	135.94	**137.49**	134.16	108%
	VGGNet19	50.25	66.03	**69.67**	49.21	139%
	ResNet18	66.49	82.03	87.95	**92.92**	140%

눈에띄는 점으로는 VGGNet19 네트워크도 매개변수 프루닝 작업을 거치게 되면 저연산 목적으로 이루어진 깊이별 우선 합성곱 신경망 기반의 MobileNet류의 모델보다 높은 NetScore를 얻을 수 있다는 점이다. Fer2013 데이터셋의 경우 사람의 정확도는 약 65±5% 정도였는데, 프루닝 작업을 거쳐도 MobileNet류의 성능에 도달하지 못했는데, 데이터셋의 안정화가 되어있지 않은상태에서 프루닝을 진행할 경우 상대적으로 값은 작지만, 중요한 매개변수를 제거한 것이 원인으로 보인다.

4.3 네트워크 간소화만 했을 경우

네트워크 간소화만 적용했을 경우의 결과는 Table 4.과 같은 결과를 보였다. 최소 10%부터 간소화비율(Slimming Rate)는 최대 80%까지 진행하다. 이때특정 딥러닝 모델에서는 일정 비율이후 재결합이 안되었는대 최대한 진행하였다. 모델이 재결합이 안되는 이유는 간소화 구조 특성상 식 (2)를 통하여 추출한 필터별 임계치를 기준으로 네트워크 간소화를 하게 되는데 현재 계산한 은닉층의 값과 이전에 계산한 은닉층의 값의 연결이 없기때문에 일어나는 현상으로 보인다.

네트워크 간소화 기법을 통하여 가장 성능향상의 폭이 높은 모델과 데이터셋은, Cifar10에서는 VGGNet19가 135%로 가장 높은 성능향상의 폭이 있었다.

Cifar100에서는 MobileNet-V1의 성능이 가장 뛰어났다. 그 이유는 Table 4.에서 보듯이 MobileNet-V1이 매개변수가 가장 많아 네트워크 간소화를 통하여 매개변수의 수를 가장 많이 줄일 수 있기 때문으로 보인다.

Fer2013에서는 VGGNet19가 가장 성능향상의 폭이 높았다. 그 이유로는 FER2013은 최대 정확도가 낮기 때문에 뉴런들의 값이 크기않아서 중요한 필터들을 제거할 가능성이 커지고 Skip-Connection[10]과 같은 연산을 한 결과를 제거할 수 있기 때문으로 보인다.

Ferplus에서의 결과는 MobileNet-V1이 가장 높았다. 이러한 이유 역시 Cifar100의 네트워크 간소화를 할 때와 같은 결과로 보인다. 가장 저조한 성능을 보이는 모델은 VGGNet19이었는데, VGGNet모델의 경우 단순히 3×3합성곱 신경망으로만 모델을 구성했기 때문에 프루닝에 대한 성능향상이 부족한 것으로 보인다. Cifar100데이터셋에서 VGGNet19모델의 경우는 95% 프루닝 할 경우 정확도가 약 2.75%가 되는데 학습 매개변수의 부족으로 나타난 현상으로 보인다. 또한 학습 레이블이 많을수록 프루닝에 대한 효과가 적다는 것을 판단할 수 있었다.

전체적으로 네트워크 슬라이밍은 MobileNet V3가 가장 성능향상으로 인한 NetScore가 점수가 가장 높았지만, 모델 자체가 가지고 있는 연산 매개변수의 수가 적기 때문에 프루닝의 결과를 받기는 어려웠다.

Table 4. NetScore and Effciency results when channel slimming (30 to 80%) is carried out step by step for the MobileNet-v1, v2, v3, VGGNet19, and ResNet18 models for the Cifar10, Cifar100, Fer2013, and Ferplus datasets

Dataset	network	Non Prunning	pruned rate: 0.3	pruned rate: 0.4	pruned rate: 0.5	pruned rate: 0.6	pruned rate: 0.7	pruned rate: 0.8	Efficiency
Cifar10	MobileNet-v1	106.86	118.13	**123.05**	90.4	90.59	-	-	115%
	MobileNet-v2	110.68	112.17	**117.56**	90.85	-	-	-	106%
	MobileNet-v3	134.33	138.91	**139.94**	76.12	-	-	-	104%
	VGGNet19	68.64	78.51	82.40	**93.18**	83.43	-	-	**135%**
	ResNet18	84.70	91.59	94.21	93.16	**109.97**	-	-	129%
Cifar100	MobileNet-v1	71.65	105.86	**111.21**	58.59	8.2	-	-	**155%**
	MobileNet-v2	75.57	105.46	**107.55**	68.29	-	-	-	142%
	MobileNet-v3	109.97	114.21	**115.12**	42.62	-	-	-	104%
	VGGNet19	58.17	67.66	**69.85**	14.55	-	-	-	120%
	ResNet18	73.55	**78.18**	-	-	-	-	-	106%
Fer2013	MobileNet-v1	87.48	97.81	**102.54**	68.45	61.71	59.07	-	117%
	MobileNet-v2	91.30	98.31	**100.51**	0.6858	-	-	-	110%
	MobileNet-v3	119.41	123.80	**124.57**	-	-	-	-	104%
	VGGNet19	48.97	58.23	**61.27**	60.36	52	-	-	**125%**
	ResNet18	65.61	71.44	**73.80**	-	-	-	-	112%
Ferplus	MobileNet-v1	66.30	99.98	104.77	82.73	82.35	**119.93**	117.77	**180%**
	MobileNet-v2	69.86	97.75	99.59	82.23	82.27	-	-	142%
	MobileNet-v3	127.43	130.99	**130.99**	76.65	-	-	-	102%
	VGGNet19	50.25	60.01	63.49	**72.93**	50.67	51.06	-	145%
	ResNet18	66.49	73.02	75.50	**79.39**	-	-	-	119%

4.4 매개변수 프루닝과 네트워크 간소화 기법을 함께 적용하였을 경우

매개변수 프루닝과 네트워크 간소화를 함께 진행한 결과는 Table 5와 같은 결과를 보였다. 네트워크 간소화와 매개변수 프루닝 중 먼저 선택된 방법은 매개변수 프루닝이었다. 그 이유는 네트워크 간소화를 먼저 하게되면, 필터 안의 특정 중요 필터를 먼저 없애기 때문에 매개변수 프루닝 이후에 네트워크 간소화를 진행하였다. 네트워크 간소화는, 최대 60%까지 진행되었으며 그 이상을 진행할 경우 네트워크 재구성이 안되는 결과를 보였다.

Table 5.의 모델별 성능을 보면, MobileNet-V1, V2의 경우는 은 매개변수 프루닝과 네트워크 간소화를 통하여 성능이 기존 모델 대비 최대 166%성능 향상이 있었다. MobileNet-V3같은 경우는, 기존 모델 대비 116% 성능 향상이 있었다. VGGNet19과 ResNet18은 각각 최대 142%, 133% 성능 향상이 있었다.

MobileNet-V3는 모델 자체의 매개변수가 적기 때문에 기대하는 감소되는 매개변수의 수가 작고, 네트워크 필터의 구조를 조금만 제거해버려도 남은 필터의 수가 없어 다시 모델 재구성이 안되는 결과를 가져왔다.

5. 결론

본 논문에서는 딥러닝 모델에서 대표적인 프루닝 방법인 매개변수 프루닝과 네트워크 간소화 방법을 구현하였다. 또한 각 방법을 각기 적용하였을 경우와 함께 적용하였을 경우를, 데이터 크기와 학습 레이블이 다른 네가지 데이터셋 3종류에 대하여 적용하였고 각기다른 신경망 모델 5가지와 MobileNet-V1, V2, V3와 VGGNet19, ResNet18 모델에 학습하여 실험하여 비교하였다.

제안하는 더블 푸루닝 방법은, 매개변수 프루닝을 먼저(10~95%) 한 뒤에 네트워크 간소화 비율을 점차 높여가는 방법으로 구성하였다. 이를 통하여 신경망의 정확도, 연산량의 균형을 측정하는 방법 중 하나인 NetScore를 적용하여 신경망을 평가하고자 하였다.

초기에 가장 신경망의 NetScore가 높은 모델은 MobileNet-v3 였으나 가장 성능 향상의 폭이 적은 모델 또한 MobileNet-V3였다. 이는 신경망 모델 자체에 가지고 있는 매개변수의 수가 적은 결과를 보였다.

본 실험을 통하여 매개변수 프루닝을 통하여 깊이별 분리 합성곱 신경망도 프루닝 효과가 있다는 점이였다.

Table 5. NetScore results when channel sliming(S: channel sliming rate) is applied after parameter pruning(P: parameter pruning rate) of the MobileNet-v1, v2, v3, VGGNet19, and ResNet18 models for the Cifar10, Cifar100, Fer2013, and Ferplus datasets

Dataset	Network	Non-Prunning	Parameter-Prunning	Channel-Slimming	Parameter-Prunning + Channel-Slimming
Cifar10	MobileNet-v1	106.86	136.45(P:0.95)	123.05(P:0.4)	151.09(s:0.6 P:0.8)
	MobileNet-v2	106.86	132.07(P:0.9)	117.56(P:0.4)	137.69(S:0.4, P:0.9)
	MobileNet-v3	134.33	145.43(P:0.95)	139.94(P:0.4)	153.96(S:0.4, P:0.95)
	VGGNet19	68.64	91.48(P:0.9)	82.40(P:0.4)	96.76(S:0.2, P:0.9)
	ResNet18	84.7	111.65(P:0.95)	109.97(P: 0.6)	113.41(S:0.2, P:0.95)
Cifar100	MobileNet-v1	71.65	113.62(P:0.9)	111.21(P:0.4)	118.84(S:0.4, P:0.7)
	MobileNet-v2	75.57	111.48(P:0.9)	107.55(P:0.4)	118.64(S:0.6, p:0.7)
	MobileNet-v3	109.97	120.02(P:0.95)	115.12(P:0.4)	127.82(S:0.4, P:0.9)
	VGGNet19	58.17	73.93(P:0.8)	69.85(P:0.4)	73.94(S:0.2, P:0.8)
	ResNet18	73.55	92.11(P:0.9)	78.18(P:0.3)	92.77(S:0.2, P:0.9)
Fer2013	MobileNet-v1	87.48	109.27(P:0.9)	102.54(P:0.4)	118.17(S:0.6,p:0.7)
	MobileNet-v2	91.3	108.35(P:0.9)	100.51(P:0.4)	136.47(S:0.4, P:0.95)
	MobileNet-v3	119.41	130.87(P:0.95)	124.57(P:0.4)	136.47(S:0.4, P:0.95)
	VGGNet19	48.97	64.72(P:0.8)	61.27(P:0.4)	65.50(S:0.6, P:0.5)
	ResNet18	65.61	87.80(P:0.95)	73.80(P:0.4)	81.63(S:0.2, P:0.9)
Ferplus	MobileNet-v1	66.3	111.36(P:0.9)	119.93(P:0.7)	125.58(S:0.6, P:0.7)
	MobileNet-v2	69.86	112.77(P:0.9)	99.59(P:0.4)	116.13(S:0.6, P:0.7)
	MobileNet-v3	127.43	137.49(P:0.9)	130.99(P:0.4)	145.57(S:0.4, P:0.9)
	VGGNet19	50.25	69.67(P:0.9)	72.93(P:0.5)	71.45(S:0.6, P:0.95)
	ResNet18	66.49	92.92(P:0.95)	79.39(P:0.4)	83.25(S:0.4, P:0.6)

또한, MobileNet 계열의 깊이별-우선 합성곱 신경망도 전통적인 합성곱 신경망에 비하여 연산량이 적어 프루닝의 효과가 없을 줄 알았으나, 우수한 성능을 보였다. VGGNet19 과 같은 간단한 모델도, 매개변수 프루닝과 네트워크 간소화 방법과 같은 프루닝 기법을 적용한다면, 프루닝을 안한 저연산 목적의 모델인 MobileNet류보다 높은 NetScore를 보일 수 있다. 실험을 통하여 제안하는 두 가지 방법을 혼합한 더블 푸르닝을 하는것이 단일방법을 사용하는 것보다 효율성이 우수하다는 것을 실험을 통해 알 수 있었다.

본 논문에서 제안하는 더블 프루닝방법은, 모델별, 데이터셋 별로 최적의 프루닝 비율과 네트워크 간소화 비율이 정해져 있지 않기 때문에 실험적으로 해야하기 때문에 최적화 시간이 오래 걸린다는 단점이 있다.

본 논문에서 제안하는 방법은 컴퓨터 자원이 다소 부족한 디바이스나 최적화 등이 요구되는 독립형(Standalone) 타입의 인공지능 디바이스 등에 적용이 가능할 수 있다.

향후 연구로는 매개변수 프루닝과 네트워크 간소화과 같은 연산량 절감과 상대적으로 불필요한 매개변수를 은닉층별로 찾는 방법을 제안하여 프루닝에 걸리는 시간을 단축해보고자 한다.

REFERENCES

[1] E. Real, A. Aggarwal, Y. Huang & Q. V. Le. (2019, July). Regularized evolution for image classifier architecture search. *In Proceedings of the aaai conference on artificial intelligence, 33,* 4780-4789. DOI : 10.1609/aaai.v33i01.33014780

[2] S. Karen & Z. Andrew. (2014), Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556.*

[3] K. He, X. Zhang, S. Ren & J. Sun. (2016). Deep residual learning for image recognition. *In Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[4] C. Szegedy et al. (2015). Going deeper with convolutions. *In Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).

[5] A. Howard. et al. (2019). Searching for mobilenetv3. *In Proceedings of the IEEE International Conference on Computer Vision* (pp. 1314-1324).

[6] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov & L. C. Chen. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. *In Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4510-4520).

[7] G. H. Andrew, Z. Menglong, C. Bo, K. Dmitry, W. Weijun, W. Tobias, A. Marco & A. Hartwig. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861.*

[8] N. I, Forrest. H. Song, W. M. Matthew, A. Khalid, J. D. William & K. Kurt. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. *arXiv preprint arXiv:1602.07360.*

[9] X. Zhang, X. Zhou, M. Lin & J. Sun. (2018). Shufflenet: An extremely efficient convolutional neural network for mobile devices. I*n Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 6848-6856).

[10] F. Chollet. (2017). Xception: Deep learning with depthwise separable convolutions. *In Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1251-1258).

[11] Gholami, A., Kwon K., Wu B., Tai Z., Yue X., Jin P., Zhao S., Keutzer K., (2018. June). Squeezenext: Hardware-aware neural network design. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (pp. 1638-1647).

[12] L. Yann, S. D. John & A. S. Sara. (1990). Optimal brain damage. *In Advances in neural information processing systems* (pp. 598-605).

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

[14] H. Song, M. Huizi & J. D. William (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149.*

[15] R. Reed, (1993). Pruning algorithms-a survey. *IEEE transactions on Neural Networks, 4(5),* 740-747. DOI : 10.1109/72.248452

[16] N. Lee, T. Ajanthan & P. H. Torr. (2018). Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340.*

[17] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan & C. Zhang, (2017). Learning efficient convolutional networks through network slimming. *In*

Proceedings of the IEEE International Conference on Computer Vision (pp. 2736-2744).

[18] Y. He, X. Zhang & J. Sun, (2017). Channel pruning for accelerating very deep neural networks. *In Proceedings of the IEEE International Conference on Computer Vision* (pp. 1389-1397).

[19] M. Tan & Q. V. Le. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946.*

[20] M. Tan & Q. V. Le. (2019). Mixconv: Mixed depthwise convolutional kernels. *CoRR, abs/1907.09595*

[21] J. H. Luo, J. Wu & W. Lin. (2017). Thinet: A filter level pruning method for deep neural network compression. *The IEEE International Conference on Computer Vision* (ICCV) (pp. 5058-5066)

[22] Z. Liu, M. Sun, T. Zhou, G. Huang, T. Darrell. (2019). Rethinking the Value of Network Pruning, *International Conference on Learning Representations (ICLR) Seq*

[23] A. Morcos, H. Yu, M. Paganini & Y. Tian. (2019). One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. *In Advances in Neural Information Processing Systems* (pp. 4932-4942).

[24] A. Krizhevsky & Hinton, G. (2009). Learning multiple layers of features from tiny images.

[25] I. J. Goodfellow et al. (2013, Nov). Challenges in Representation Learning: A report on three machine learning contests. *In International Conference on Neural Information Processing* (pp. 117-124). Springer, Berlin, Heidelberg.

[26] E. Barsoum, C. Zhang, C. C. Ferrer & Z. Zhang. (2016, October). Training deep networks for facial expression recognition with crowd-sourced label distribution. *In Proceedings of the 18th ACM International Conference on Multimodal Interaction* (pp. 279-283).

[27] A. Wong. (2019, August). NetScore: Towards universal metrics for large-scale performance analysis of deep neural networks for practical on-device edge usage. *In International Conference on Image Analysis and Recognition* (pp. 15-26). Springer, Cham.

[28] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard & Q. V. Le. (2019). Mnasnet: Platform-aware neural architecture search for mobile. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2820-2828).

[29] P. Ramachandran, B. Zoph & Q. V. Le. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941.*

[30] I. Loshchilov & F. Hutter. (2016). Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983.*

[31] A. Aimar et al. (2018). Nullhop: A flexible convolutional neural network accelerator based on sparse representations of feature maps. I*EEE transactions on neural networks and learning systems, 30(3),* 644-656.
DOI : 10.1109/TNNLS.2018.2852335

[32] M. Schmidt, G. Fung & R. Rosales. (2007, September). Fast optimization methods for l1 regularization: A comparative study and two new approaches. *In European Conference on Machine Learning* (pp. 286-297). Springer, Berlin, Heidelberg.
DOI : 10.1007/978-3-540-74958-5_28

이 선 우(Seon-Woo Lee) [정회원]

· 2017년 2월 : 인하대학교 컴퓨터정보학과(공학사)
· 2019년 2월 : 인하대학교 컴퓨터공학과 (공학석사)
· 2019년 3월 ~ 현재 : 인하대학교 컴퓨터공학과 박사과정

· 관심분야 : 머신러닝, 인공지능, 데이터분석, HCI
· E-Mail : x21999@inha.edu

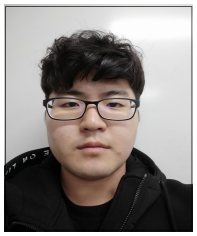
양 호 준(Ho-Jun Yang) [학생회원]

· 2015년 2월 ~ 현재 : 인하대학교 컴퓨터공학과 재학
· 관심분야 : 인공지능
· E-Mail : 12151415@inha.edu

오 승 연(Seung-Yeon Oh) [학생회원]

· 2015년 2월 ~ 현재 : 인하대학교 컴퓨터공학과 재학
· 관심분야 : 인공지능
· E-Mail : 12151415@inha.edu

이 문 형(Mun-Hyung Lee) [학생회원]



· 2016년 2월 ~ 현재 : 인하대학교 컴퓨터공학과 재학
· 관심분야 : 인공지능
· E-Mail : mun0659@gmail.com

권 장 우(Jang-Woo Kwon) [정회원]

· 1990년 2월 인하대학교 전자공학과 졸업
· 1992년 2월 인하대학교 전자공학과석사
· 1996년 8월 인하대학교 전자공학과박사
· 1996년 10월~ 1998년 2월: 특허청 사무관
· 1998년 4월 ~ 2009년 12월: 동명대학교 컴퓨터공학과 부교수
· 2006년 12월 ~ 2012년 2월: 정보통신산업진흥원 인재양성단장
· 2012년 3월 ~ 현재 : 인하대학교 컴퓨터공학과 교수
· 관심분야 : 인공지능, 인간과 컴퓨터 상호작용, 딥러닝
· E-Mail : jwkwon@inha.ac.kr