특집논문 (Special Paper)
방송공학회논문지 제25권 제2호, 2020년 3월 (JBE Vol. 25, No. 2, March 2020)
https://doi.org/10.5909/JBE.2020.25.2.200
ISSN 2287-9137 (Online) ISSN 1226-7953 (Print)

임베디드 보드에서의 CNN 모델 압축 및 성능 검증

문 현 철[a)], 이 호 영[a)], 김 재 곤[a)‡]

Compression and Performance Evaluation of CNN Models on Embedded Board

Hyeon-Cheol Moon[a)], Ho-Young Lee[a)], and Jae-Gon Kim[a)‡]

요 약

CNN 기반 인공신경망은 영상 분류, 객체 인식, 화질 개선 등 다양한 분야에서 뛰어난 성능을 보이고 있다. 그러나, 많은 응용에서 딥러닝(Deep Learning) 모델의 복잡도 및 연산량이 방대해짐에 따라 IoT 기기 및 모바일 환경에 적용하기에는 제한이 따른다. 따라서 기존 딥러닝 모델의 성능을 유지하면서 모델 크기를 줄이는 인공신경망 압축 기법이 연구되고 있다. 본 논문에서는 인공신경망 압축 기법을 통하여 원본 CNN 모델을 압축하고, 압축된 모델을 임베디드 시스템 환경에서 그 성능을 검증한다. 성능 검증을 위해 인공지능 지원 맞춤형 칩인 QCS605를 내장한 임베디드 보드에서 카메라로 입력한 영상에 대해서 원 CNN 모델과 압축 CNN 모델의 분류 성능과 추론시간을 비교 분석한다. 본 논문에서는 이미지 분류 CNN 모델인 MobileNetV2, ResNet50 및 VGG-16에 가지치기(pruning) 및 행렬분해의 인공신경망 압축 기법을 적용하였고, 실험결과에서 압축된 모델이 원본 모델 분류 성능 대비 2% 미만의 손실에서 모델의 크기를 1.3 ~ 11.2배로 압축했을 뿐만 아니라 보드에서 추론시간과 메모리 소모량을 각각 1.2 ~ 2.1배, 1.2 ~ 3.8배 감소함을 확인했다.

Abstract

Recently, deep neural networks such as CNN are showing excellent performance in various fields such as image classification, object recognition, visual quality enhancement, etc. However, as the model size and computational complexity of deep learning models for most applications increases, it is hard to apply neural networks to IoT and mobile environments. Therefore, neural network compression algorithms for reducing the model size while keeping the performance have been being studied. In this paper, we apply few compression methods to CNN models and evaluate their performances in the embedded environment. For evaluate the performance, the classification performance and inference time of the original CNN models and the compressed CNN models on the image inputted by the camera are evaluated in the embedded board equipped with QCS605, which is a customized AI chip. In this paper, a few CNN models of MobileNetV2, ResNet50, and VGG-16 are compressed by applying the methods of pruning and matrix decomposition. The experimental results show that the compressed models give not only the model size reduction of 1.3~11.2 times at a classification performance loss of less than 2% compared to the original model, but also the inference time reduction of 1.2~2.21 times, and the memory reduction of 1.2~3.8 times in the embedded board.

Keyword : CNN, Neural Network Compression, Pruning, Matrix Decomposition, Embedded Board

Ⅰ. 서 론

CNN(Convolutional Neural Network) 기반 인공신경망은 영상 분류, 객체 인식, 화질 개선 등 다양한 분야에서 뛰어난 성능을 보이고 있다. 하지만 보다 나은 성능을 얻기 위해 전체 계층 수 및 학습할 파라미터 수가 크게 증가하여 연산량 및 모델 크기가 또한 크게 증가하였고, 이에 따라 모바일 및 IoT 기기, 그리고 임베디드 시스템과 같이 연산 속도나 메모리가 제한된 환경에서 인공신경망을 적용하기에는 제한이 따른다. 예를 들어, 이미지 분류 모델인 VGG-16의 모델 크기는 약 528MB로써 실제 모바일 및 하드웨어 기기에서 추론을 하기 위해서는 기기 내에 계속 해당 모델을 저장하고 있거나 중앙 서버로부터 전송받아야 한다. 따라서, 이 경우 500MB가 넘는 모델의 파라미터를 추론할 때마다 로딩해야 하고, 이는 저전력 기기에서 부담이 매우 크다. 따라서 기존의 학습된 인공신경망 모델의 성능을 최대한 유지하면서 모델 크기 및 연산량을 줄이는 연구가 진행되고 있다[1].

인공신경망 모델의 복잡도를 줄이는 연구는 크게 연산에 효율적인 인공신경망 구조를 설계하는 인공신경망 경량화 기법과 학습된 파라미터 등을 압축하는 인공신경망 압축 기법으로 나눌 수 있다. 먼저, 인공신경망 경량화 기법은 잔여 블록(Residual Block) 이나 병목 블록(Bottle- neck Block)과 같은 신경망 구조를 이용하여 파라미터 축소 및 모델 성능을 개선하는 구조적 연구와 점별(Pointwise) 및 채널별(Depthwise) 합성곱(Convolution) 필터로 대표되는 합성곱의 연산을 효율적으로 줄이는 연구 등이 있다[2,3,4]. 반면에, 인공신경망 압축 기법은 인공신경망 구조적 설계가 아닌 인공신경망 모델을 표현하는 파라미터의 크기를 줄이는데 주목적을 가지고 있다. 대표적인 인공신경망 압축 기법으로는 모델 내에서 매우 작은 값을 가지는 가중치를 모델의 성능에 내성을 가지고 있다고 간주하고 연결을 끊는(해당 가중치를 0으로 설정) 가지치기(Pruning) 기법과 일반적으로 부동소수점으로 표현되는 가중치를 특정 비트수로 줄여 실제 모델의 저장 크기를 줄이는 양자화 기법[5], 그리고 각 계층의 2차원 이상의 가중치 행렬을 2개 이상의 행렬로 분해하여 가중치의 수를 줄이는 행렬분해기법[6,7] 등이 있다. 따라서, 인공신경망 압축 기법은 이미 학습된 딥러닝 모델에 적용함으로써 기존의 모델 대비 모델의 크기를 줄일 수 있을 뿐만 아니라 실제 추론(Inference) 과정에서의 연산량을 줄일 수 있다[8]. 본 논문에서는 인공지능 지원 맞춤형 칩인 QCS605 칩을 내장한 임베디드 보드에서 이미지 분류 CNN 모델인 VGG-16, ResNet50, 그리고 MobileNetV2 모델에 대하여 인공신경망 압축 기법인 가지치기 및 행렬분해 기법을 적용하여 기존 모델 대비 모델의 크기(용량) 및 추론 과정에서의 추론시간과 메모리 소모량, 그리고 이미지 분류 성능인 Top-5 정확도를 비교하여 성능을 검증한다. 실제 실험은 보드 내의 내장된 카메라로 입력된 영상에 대하여 ImageNet의 1000개 클래스로 영상을 분류하는 방식으로 진행하였는데, 이 경우 정확한 Top-5 분류 성능을 획득하기 어렵기 때문에 오프라인(데스크탑 PC)에서 ImageNet 검증(validation) 데이터셋에 대한 Top-5 분류 성능을 측정하였다.

본 논문의 구성은 다음과 같다. 2장에서는 실험에 적용된 인공신경망 압축 기법인 가지치기 및 행렬분해 기법에 대해 설명하고, 3장에서는 임베디드 보드 실험환경과 제안된 기법의 상세 실험내용 및 각 딥러닝 모델에서의 압축 기법의 실험결과에 대하여 분석하고, 마지막으로 4장에서 결론을 맺는다.

Ⅱ. 인공신경망 압축 기법

인공신경망 압축 기법은 학습된 인공신경망 모델 파라미터를 압축하여 모델을 저장하는 메모리 및 추론 과정에서의 연산량 및 메모리 접근량을 줄이는 기법이다. 본 논문에

a) 한국항공대학교 항공전자정보공학부(School of Electronics and Information Engineering, Korea Aerospace University)
‡ Corresponding Author: 김재곤(Jae-Gon Kim)
E-mail: jgkim@kau.ac.kr
Tel: +82-2-300-0414
ORCID: https://orcid.org/0000-0003-3686-4786
※ 이 논문의 연구결과 중 일부는 한국방송·미디어공학회 “2019년 추계학술대회”에서 발표한 바 있음.
※ This work was supported by National Standards Technology Promotion Program of KATS grant funded by Korea Government (MOTIE) (No. 10084981).
· Manuscript received January 15, 2020; Revised March 2, 2020; Accepted March 4, 2020.

서는 불필요한 가중치를 0으로 변경하여 노드와의 연결을 끊는 가지치기 기법과 각 계층에서 가지고 있는 가중치 행렬을 2개 이상의 행렬로 분해하여 가중치의 수를 줄이는 행렬분해 기법을 사용한다.

1. 가지치기(Pruning)

가지치기 기법은 딥러닝 모델에서 가지고 있는 각각의 학습된 가중치 중에서 모델의 성능에 미치는 중요도가 떨어져 상대적으로 불필요하다고 판단되는 가중치와 노드와의 연결을 끊어서 모델의 크기를 줄이는 기법이다. 수식 (1)은 본 논문에서 실험한 대표적인 가지치기 기법을 수식으로 나타낸 것이다. 여기서 w_i는 딥러닝 모델 내의 i번째 가중치를 의미하며, ϵ는 가중치의 대소 관계를 판단하는 임계값이다. 본 논문에서의 가지치기 기법은 일반적으로 수식 (1)과 같이 모델 내의 각각의 가중치의 절대 값이 매우 작을 경우 모델의 성능에 내성을 가지고 있다고 간주하고 값을 0으로 변경한다[1]. 본 논문에서는 가중치의 대소 관계를 판단하기 위해 각 모델에서 목표로 하는 희소도(sparsity)를 충족하는 임계 값을 구하고, 임계 값보다 낮은 값을 갖는 각각의 가중치들에 대하여 해당 가중치를 0으로 만든다. 여기서 희소도는 전체 가중치의 수 대비 0의 값을 가진 가중치의 비율을 의미한다.

$$w_i = \begin{cases} 0, & \text{if} \left| w_i \right| < \epsilon \\ w_i, & otherwise \end{cases} \tag{1}$$

2. 행렬분해 기법(Matrix Decomposition)

행렬분해 기법은 2차원 이상의 각 계층의 가중치 행렬 1개에 대하여 2개 이상의 행렬로 분해함으로써 가중치의 수 및 연산량을 줄이는 기법이다. 대표적인 행렬분해 기법으로는 2차원 행렬을 SVD(Singular Value Decom- position) 분해 기법을 이용하여 2개의 행렬로 분해하는 낮은 순위 근사(Low-rank Approximation) 기법[6] 과 3차원 이상의 행렬을 다수의 rank-1 텐서(tensor)의 선형결합 형태로 분해하는 CP(Canonical Polyadic) 분해 기법[7] 등이 있다. 본 논문에서는 2차원 가중치 행렬로 구성되어 있는 완전연결 층(Fully-Connected layer: FC layer)에는 낮은 순위 근사 기법을 사용하여 2개의 층으로 분해하고, 4차원 가중치 행렬로 구성되어 있는 합성곱 층에는 CP 분해 기법을 사용하여 3개의 층으로 분해한다.

낮은 순위 근사 방법은 식 (2)와 같이 2차원 행렬을 SVD 분해 기법을 이용하여 2차원 행렬 2개로 분해한다.

$$W_i = U_i V_i^T \tag{2}$$

여기서 W_i, U_i, V_i^T는 각각 $M \times N$, $M \times R$, $R \times N$ 의 크기를 가지며, W_i는 각 CNN 모델의 i번째 층의 가중치 행렬을 의미하며, U_i, V_i^T는 W_i로부터 해당 기법으로 분해되는 2개의 행렬을 의미한다. 또한 M, N은 각각 i번째 층의 입력과 출력의 노드 수를 의미하며, R은 분해의 계수인 rank 값이다.

CP 분해 기법은 일반적으로는 3차원 이상의 N차원 텐서들을 N개의 rank-1 텐서들의 선형결합으로 분해한다. 예를 들어, 식 (3)과 같이 3차원 텐서를 R개로 이루어진 3개의 rank-1텐서로 분해할 수 있다. 여기서 R은 랭크(rank) 값이고, 해당 기법을 적용하기 위해 입력되는 파라미터이며, R 값에 따라 압축율이 상대적으로 결정된다.

$$X \equiv \sum_{r=1}^{R} a_R \otimes b_R \otimes c_R \tag{3}$$

본 논문에서는 4차원으로 표현되는 합성곱 가중치 텐서들을 CP 분해 기법을 적용하여 압축한다[3]. 일반적인 2D 합성곱 층의 가중치들은 4D(필터 가로 크기, 필터 세로 크기, 입력채널 수, 출력채널 수)로 구성되어 있다. 따라서, 합성곱 가중치 텐서 K에 대한 CP 분해는 다음 식 (4)과 같이 표현할 수 있으며, 본 논문에서는 각각의 합성곱 층의 4차원 행렬을 3개의 층으로 분해하는 기법을 제시한다.

$$K_{t,x,j,i} \equiv \sum_{r=1}^{R} U_{r,s}^{(1)} U_{r,j,i}^{(2)} U_{t,r}^{(3)} \tag{4}$$

여기서 $U_{r,s}^{(1)}$, $U_{r,j,i}^{(2)}$, $U_{t,r}^{(3)}$는 각각 $R\times S$, $R\times D\times D$, $R\times T$ 크기를 가진다. 이 때, S와 T는 각각 입력과 출력의 채널 수, $D\times D$는 합성곱의 필터 크기를 의미한다. 따라서, $U_{r,s}^{(1)}$, $U_{t,r}^{(3)}$은 필터 크기가 1인 점별 합성곱 층의 형태로 구성될 수 있으며, $U_{r,j,i}^{(2)}$는 필터크기가 $D\times D$인 채널별 합성곱 층으로 구성되며, 결국 하나의 합성곱 층을 2개의 점별, 1개의 채널별 합성곱 층으로 분해하게 된다. 그림 1은 위의 수식 (4)에 대하여 그림으로 나타낸 것이다.

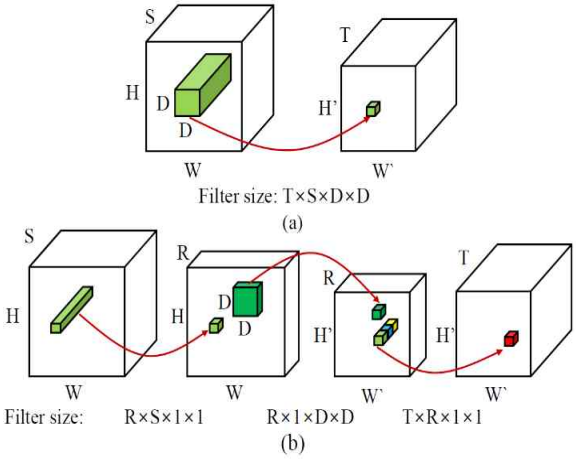

그림 1. CP 분해 기법 (a) 합성곱 층 (b) CP 분해된 합성곱[5]
Fig. 1. CP decomposition (a) Convolutional layer (b) CP decomposed convolutional layer[5]

III. 실험결과 및 분석

1. 임베디드 보드

본 논문의 실험에서는 학습된 인공신경망 모델들을 QCS605 SoC(System on Chip) 칩이 내장된 임베디드 보드에서 실험을 진행하였다. 실험환경 구성을 위해 안드로이드 스튜디오(Android Studio)에서 인공신경망 모델을 이용한 영상 분류 어플(apk 파일)과 학습된 인공신경망 모델을 임베디드 보드에 업로드하고, 보드에 내장된 카메라를 이용하여 입력된 영상에 대한 각각의 CNN 모델의 분류 성능 및 추론 과정에서의 복잡도 등을 측정하였다. 다음은 실험환경의 기반이 되는 하드웨어의 규격이다. QCS605 칩에 대한 규격은 표 1과 같다[9]. 표 2는 QCS605 칩을 내장한 임베디드 보드 QCS605 EV-R.2 보드의의 규격을 나타낸 것이다.

표 1. QCS605 칩의 규격
Table 1. Specification of QCS605 Chip

	Specification
CPU	Qualcomm® Kryo™ 300 CPU, Octa-core CPU / Up to 2.5 GHz / 64-bit Architecture
GPU	Qualcomm® Adreno™ 615 GPU PI Support: OpenGL® ES 3.2, Vulkan® 1.1, OpenCL
Memory	eMCP / LPDDR4x 4GB, 1866MHz

표 2. QCS605 EV-R.2 보드의 규격
Table 2. Specification of QCS605 EV-R.2 Board

	Specification
OS	Android 9.x / Kernel 4.x
Memory	64GB
Support I/O	2 MIPI CSI, 2 MIPI DSI / USB3.1 and DP / Audio input / output / SD Card I/F 1 port

2. 실험 개요

그림 2는 본 논문에서의 실험을 위한 환경 구성을 나타내었다. 성능을 검증하기 위해 임베디드 보드에 내장되어 있는 카메라로 입력된 영상에 대하여 각각의 학습된 인경신경망 모델로 영상 분류에 대한 추론(inference)을 진행하였다.

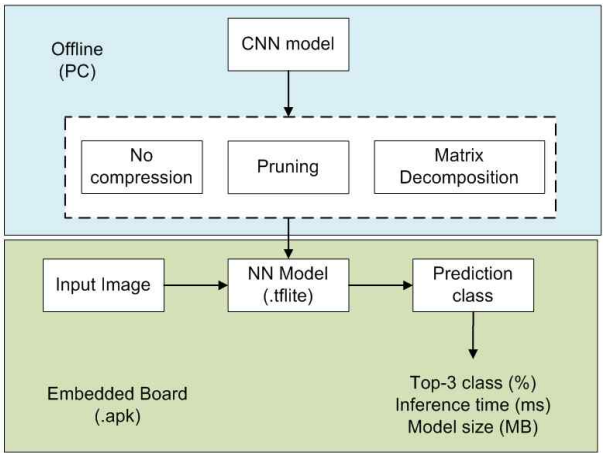

그림 2. 실험환경 구성
Fig. 2. Experimental Configuration

추론된 결과를 화면에 예측된 클래스 중 상위 3개의 확률 값과 추론시간, 그리고 업로드한 인공신경망 모델의 크기도 표시하였으며, 그림 3은 실험결과 화면의 예이다. 또한 실험 비교를 위해 PC 환경에서 인공신경망 압축 기법을 적용한 모델을 임베디드 보드로 업로드하여 원본 모델과 압축된 모델의 성능을 검증하였다. 여기서 추론시간은 해당 인공신경망 모델을 추론하기 위해 모델을 로딩하는 시간과 더불어 입력된 이미지의 분류를 위한 추론시간을 합친 시간이다. 또한 본 논문에서는 공정한 실험비교를 위해 실험에 사용한 모든 모델을 Tensorflow의 tflite (Tensorflow Lite) 모델 파일 포맷을 사용하였다[10].

그림 3. 실험결과 실제화면 예
Fig. 3. An example of experimental result

3. 실험조건

본 논문에서의 실험은 분류 CNN 모델인 VGG-16, ResNet50, 그리고 MobielNetV2에 대하여 원본 및 인공신경망 압축 기법이 적용된 모델들의 성능을 비교하였다. 표 3은 별도의 압축 알고리즘이 적용되지 않은 각각의 원본 CNN 모델의 성능이다. 표 3에서의 Top-5 Accuracy는 모델로부터 예측된 클래스 중 상위 5개에 정답이 있을 경우의 정확도를 의미하여, 분류 성능을 측정하기 위해 ILSVRC (ImageNet Large Scale Visual Recogni- tion Competition) 2012의 검증(validation) 데이터 셋의 50,000개의 영상에 대해 실험하였고, 입력되는 영상의 크기는 가로 및 세로의 크기가 224이다[11].

표 3. 실험에 사용된 CNN 원본 모델의 성능
Table 3. Performance of the CNN original model used in the experiment

Model	Model Size (MB)	Top-5 Accuracy (%)	Inference time (ms)
VGG-16	527.0	90.05	2,000
ResNet50	98.2	91.93	800
MobileNetV2	13.8	90.06	118

4. 가지치기 실험결과

표 4는 가지치기 기법을 적용한 실험결과이다. 표 4에서의 Top-5 Accuracy는 앞서 설명한 ILSVRC의 이미지들에 대한 성능이며(즉, 오프라인에서 측정한 결과), 임베디드 보드에서 측정한 실험결과는 모델 크기와 추론시간이다. 가지치기 기법만 적용할 경우 일부 모델에서 성능 손실이 심한 경우가 있어 모든 모델에 대하여 가지치기 한번을 수행한 뒤에 재학습(re-training)을 한 번 적용하고 다시 한번 가지치기 기법을 적용하였다[12]. 가지치기 기법은 각 모델별로 지정된 희소도 에서의 실험결과만을 포함하고 있어, 각각의 모델에서의 희소도에 따른 성능에 대한 내성을 비교하기 위하여 기존[12]보다 추가 실험하여 3가지 모델들에 동일한 희소도를 적용하여 비교하였다. 표 4에서 알 수 있듯이, 각각의 딥러닝 모델들이 희소도에 따라 민감도가 모두 다르다는 것을 알 수가 있으며, 그 중에 MobileNetV2가 희소도에 따른 민감도가 가장 높은 것을 알 수 있다. 예를 들어, 80% 희소도에서는 VGG-16에서는 2.15%, ResNet50에서는 2.07%성능 손실이 있는 반면에 MobileNetV2에서는 47.94%의 성능 손실이 있었다. 한편, 원본 모델과 가지치기 기법으로 압축된 모델의 추론 속도가 실제 보드에서 거의 동일함을 보여주는 데, 그 이유는 해당 인공신경망 모델 포맷의 한계로 가지치기 기법이 단순히 가중치를 0으로만 만들고 해당 가중치를 제거하지 않았기 때문에 실질적인 연산량의 감소가 거의 없었기 때문이다.

표 4. 가지치기 기법을 적용한 실험결과
Table 4. Experimental results of pruning

	VGG-16		ResNet50		MobileNetV2	
Sparsity (%)	Top-5 Accuracy (%)	Inference time (ms)	Top-5 Accuracy (%)	Inference time (ms)	Top-5 Accuracy (%)	Inference time (ms)
0 (Original)	90.05	2,000	91.93	800	90.06	118
20	90.01	1,950	91.78	780	89.80	115
40	89.95	1,920	91.35	760	86.85	114
60	89.52	1,920	89.94	760	72.21	112
80	87.90	1,900	89.86	750	52.12	110

5. 행렬분해 실험결과

행렬분해 기법은 앞선 가지치기법과 다르게 실제 파라미터 수를 줄이는 기법이므로 모델 압축의 지표를 희소도 대신 실제 압축율로 표기하였으며, 해당 기법은 완전연결 및 합성곱 층에만 적용하였다[13]. 표 5는 행렬분해 기법의 실험을 위한 층의 종류에 따른 세부적인 압축율 지표를 나타낸 것이다. 표 5에서 압축율(%)이 100일 경우의 비율 값은 실제 원본 모델의 합성곱 층 및 완전연결 층의 전체 가중치의 수 대비 절대적인 비율을 나타낸다. ResNet50의 합성곱 층은 일반적인 합성곱 층 대비 점별/채널별 합성곱 층이 비율이 더 높아, VGG-16 같이 동일한 압축율(10~20%)에서의 비교를 할 수 없기 때문에 해당 압축율은 실험에서 제외하였다. 또한, MobileNet의 경우는 대부분의 합성곱 층이 점별/채널별 같은 간결한 합성곱 층으로 구성되어 있어 해당 기법의 적용으로 인한 압축 효과를 기대할 수 없기 때문에 실험에서 제외하였다. 또한, 행렬분해 기법만을 적용할 때 성능 손실이 발생할 수 있으므로, 이를 보상하기 위해 별도의 재학습(re-training)을 적용하였으며, 재학습을 위해 사용한 데이터셋은 ILSVRC(ImageNet Large Scale Visual Recognition Competition) 2012의 학습(training) 데이터 셋의 1,000,000개의 영상이며, 손실함수는 기존 모델과 동일한 cross-entropy 함수이다.

표 5. 행렬분해 기법의 실험을 위한 세부적인 압축율 지표
Table 5. Details of compression ratios for experiments on matrix decomposition

	VGG-16		ResNet50	
Compression ratio (%)	Conv (%)	FC (%)	Conv (%)	FC (%)
100	11	89	93	7
80	9	71	73	7
60	7	53	55	6
20	5	15	-	-
10	4	6	-	-

표 6은 행렬분해 기법을 적용한 실험결과를 나타내었다. 여기서 메모리 접근량은 보드 내의 실제 추론 과정에서 발생하는 특징 맵(Feature Map) 과 모델 내의 가중치에 대한 메모리의 합이다. 표 6에서 알 수 있듯이, VGG-16/ResNet50 모두 행렬분해 기법을 통한 모델 크기 압축과 실제 추론시간 가속화 및 메모리 감소가 있음을 확인하였다. 특히, VGG-16에서는 압축율 60%에서는 원본 모델 대비 추론시간과 메모리 소모량을 1.5배 감소함과 함께 Top-5 분류 성능이 오히려 약 0.11% 향상됨을 확인하였다. 그러나, 압축율과 정비례하게 추론시간 및 메모리 소모량이 감소되지는 않았는데, 그 이유는 각 모델에서 각각의 층이 차지하는 파라미터 비율과 그 층에서 계산되는 연산량 및 메모리가 비례하지 않기 때문이다. 예를 들어, VGG-16의 경우는 완전연결 층의 가중치에 대한 메모리가 딥러닝 모델 전체 메모리의 89%를 차지하지만, 실제 추론에서는 완전연결 층 내의 추론되는데 걸리는 시간 및 특징 맵(Feature Map)이 차지하는 메모리가 약 10%만을 차지한다. 따라서 완전연결 층에서의 가중치에 대한 압축율은 실제 추론시간 및 메모리 감소효과 대비 약 1/8에 불과하다. 반면에, Reset50은 합성곱 층과 완전연결 층에서의 연산량과 파

표 6. 행렬분해 기법을 적용한 실험결과
Table 6. Experimental results of matrix decomposition

	VGG-16			ResNet50		
Compression ratio (%)	Top-5 Accuracy (%)	Memory Access (MB)	Inference time (ms)	Top-5 Accuracy (%)	Memory Access (MB)	Inference time (ms)
100 (Original)	90.05	614.22	2,000	91.93	180.86	800
80	90.30	524.42	1,720	90.87	171.02	660
60	90.16	413.37	1,300	87.78	154.72	570
20	89.31	211.41	980	-	-	-
10	88.10	164.55	890	-	-	-

라미터 비율이 거의 비슷하여 모델 크기와 비례하는 추론 시간 가속화를 보여주지만, 동일한 압축율에서의 VGG-16과 비교하였을 때 분류 성능의 감소 폭이 더 큼을 확인하였다. 또한, 추론 과정에서의 메모리 접근량은 특징 맵의 크기와 모델 내의 가중치의 수에 따라 비례하게 변화하지만, 각 딥러닝 모델 마다 특징 맵의 메모리 대비 가중치에 대한 메모리 비율이 다름을 확인하였다. 예를 들면, VGG-16/ ResNet50 모델의 크기는 각각 527/98MB을 차지하는 반면, 특징 맵이 차지하는 메모리는 87/82 MB이다. 더불어 행렬분해 기법은 1개의 계층을 2개 혹은 3개의 계층으로 분해하는 기법이기 때문에 가중치의 수는 줄어들 수 있지만, 특징 맵의 메모리 사이즈가 원본 모델 대비 오히려 증가할 수 있음을 확인하였다. 따라서, 입력되어지는 영상 공간해상도에 따라서 해당 기법을 적용하였을 때의 메모리의 감소효과가 달라질 수 있으며, 같은 모델의 구조로 가정하면 보다 더 작은 입력 영상 공간해상도에서 더 압축효과가 나올 수 있겠다.

Ⅳ. 결 론

본 논문에서는 이미지 분류의 대표적 CNN 모델인 VGG-16, ResNet50, MobileNetV2에 대하여 가지치기 및 행렬분해의 적용하고 임베디드 보드에서의 분류 성능 및 추론 속도에 대한 성능을 검증하였다. 실험결과 행렬분해 기법은 압축된 모델이 원본 모델 성능 대비 2%미만의 손실에서 모델의 크기를 1.3 ~ 11.2배 줄여줄 뿐만 아니라 실제 보드에서의 추론 속도과 메모리 소모량을 1.2 ~ 2.1배, 1.2 ~ 3.8배 감소함을 확인하였다. 또한, 실제 분류 성능도 2% 미만 감소의 Top-5 정확도 감소를 보였다. 반면에 가지치기 기법에서는 각각의 딥러닝 모델마다 희소도에 따라 다른 성능의 내성을 보여주었으며, 비슷한 압축율에서 행렬분해 기법 보다 더 큰 분류 성능 감소가 있음을 확인하였다. 실험결과를 통해서 제안하는 기법이 저전력 기기인 임베디드 보드에서의 인공신경망 모델을 이용한 영상 분류 응용에 효과적으로 적용이 될 수 있음을 확인하였다.

그러나, 가지치기 기법에서는 0으로 설정된 가중치를 제거하지 않는 모델 포맷의 한계로 실제 추론시간의 성능 향상을 보여주지는 못했으며, 따라서 추후 가지치기 기법에 대해 실질적인 추론 속도 감소를 할 수 있도록 포맷을 개선할 예정이다. 또한 더 다양한 모델 및 계층의 종류에 개선된 인공신경망 압축 기법을 적용하여 분류 성능 및 추론시간 성능을 검증할 예정이다.

참 고 문 헌 (References)

[1] S., Han, et al, “Deep Compression: Compressing Deep Neural Networks with pruning, trained quantization and Huffman coding,” In Proc. Computer Vision and Pattern Recognition (CVPR), Jun. 2015.

[2] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Leraning for Image Recognition,” In Proc. Computer Vision and Pattern Recognition (CVPR), Jun. 2016.

[3] A. Howard, M. Zhu, B. Chen, D. Kalenichenoko, W. Wang, T. Weyand, M. Andreetto, and H.Adam, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” In Proc. Computer Vision and Pattern Recognition (CVPR), Jul. 2017.

[4] X. Zhang, X. Zhou, M. Lin, and J. Sun, “ShuffleNet: An Exteremey Efficient Convolutional Neural Network for Mobile Devices,” In Proc. Computer Vision and Patter Recognition (CVPR), Jun. 2018.

[5] S. Jung, C. Son, S. Lee, J. Han, Y. Kwak, and S. Hwang, “Learning to Quantize Deep Networks by Optimizing Quantization Intervals with Task Loss,” In Proc. Computer Vision and Pattern Recognition (CVPR), Jun. 2019.
[6] M. Jaderberg, A.Vedaldi, and A. Zisserman, “Speeding up Convolutional Neural Networks with Low Rank Exapnsions,” In Proc. Computer Vision and Pattern Recognition (CVPR), Jun. 2014.
[7] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, “Speeding-up Convolutional Neural Networks Using Fine-tuned CP-Decomposition,” In Proc. Computer Vision and Patter Recognition (CVPR), Jun. 2015.
[8] H. Moon, H. Lee, and J. Kim, “Acceleration of CNN Model Using Neural Network Compression and its Performance Evaluation on Embedded Boards,” In Proc. KIBME Annual Fall Conf. Nov. 2019.
[9] QCS605 Specification, https://www.qualcomm.com/products/qcs605 (accessed Jan. 6, 2020).
[10] Tensorflow for Mobile & IoT, https://www.tensorflow.org/lite (accessed Jan. 6, 2020).
[11] Large Scale Visual Recognition Challenge 2012 (ILSVRC 2012), http://www.image-net.org/challenges/LSVRC/2012/ (accessed Jan. 6, 2020).
[12] Y. Luo, Y. Sho, Q. Huang, H. Hu, and L.Yu, “CE1 Report on Neural Network Compression of ZJU’s Proposal,” ISO/IEC JTC1/SC29/WG11 m50093, Oct. 2019
[13] H. Moon, J. Kim, S. Kim, S. Jang, and B. Choi, “KAU/KETI Response to the CE-1 on Neural Network Compression: CP Decomposition of Convolution Layers (Method5),” ISO/IEC JTC1/SC29/WG11 m52322, Jan. 2020.

저 자 소 개

문 현 철

- 2018년 2월 : 한국항공대학교 항공전자정보공학부 학사
- 2018년 3월 ~ 현재 : 한국항공대학교 항공전자정보공학과 석사과정
- ORCID : http://orcid.org/0000-0002-1672-2345
- 주관심분야 : 비디오 부호화, 영상처리, 딥러닝

이 호 영

- 2020년 2월 : 한국항공대학교 항공전자정보공학부 학사
- ORCID : https://orcid.org/0000-0002-0673-5601
- 주관심분야 : 영상처리, 딥러닝

김 재 곤

- 1990년 2월 : 경북대학교 전자공학과 학사
- 1992년 2월 : KAIST 전기 및 전자공학과 석사
- 1992년 2월 : KAIST 전기 및 전자공학과 박사
- 1992년 3월 ~ 2007년 2월 : 한국전자통신연구원(ETRI) 선임연구원/팀장
- 2001년 9월 ~ 2002년 7월 : Columbia University 연구원
- 2015년 12월 ~ 2016년 1월 : UC San Diego, Visiting Scholar
- 2007년 9월 ~ 현재 : 한국항공대학교 항공전자정보공학부 교수
- ORCID : http://orcid.org/0000-0003-3686-4786
- 주관심분야 : 비디오 부호화 표준, 비디오 신호처리, Immersive Video, Deep Learning