일반논문 (Regular Paper)
방송공학회논문지 제26권 제6호, 2021년 11월 (JBE Vol.26, No.6, November 2021)
https://doi.org/10.5909/JBE.2021.26.6.778
ISSN 2287-9137 (Online) ISSN 1226-7953 (Print)

비디오 인코더를 통한 딥러닝 모델의 정수 가중치 압축

김 승 환[a)], 류 은 석[a)‡]

Compression of DNN Integer Weight using Video Encoder

Seunghwan Kim[a)] and Eun-Seok Ryu[a)‡]

요 약

최근 다양한 분야에서 뛰어난 성능을 나타내는 Convolutional Neural Network(CNN)모델을 모바일 기기에서 사용하기 위한 다양한 연구가 진행되고 있다. 기존의 CNN 모델은 모바일 장비에서 사용하기에는 가중치의 크기가 크고 연산복잡도가 높다는 문제점이 있다. 이를 해결하기 위해 가중치의 표현 비트를 낮추는 가중치 양자화를 포함한 여러 경량화 방법들이 등장하였다. 많은 방법들이 다양한 모델에서 적은 정확도 손실과 높은 압축률을 나타냈지만, 대부분의 압축 모델들은 정확도 손실을 복구하기 위한 재학습 과정을 포함시켰다. 재학습 과정은 압축된 모델의 정확도 손실을 최소화하지만 많은 시간과 데이터를 필요로 하는 작업이다. Weight Quantization이후 각 층의 가중치는 정수형 행렬로 나타나는데 이는 이미지의 형태와 유사하다. 본 논문에서는 Weight Quantization이후 각 층의 정수 가중치 행렬을 이미지의 형태로 비디오 코덱을 사용하여 압축하는 방법을 제안한다. 제안하는 방법의 성능을 검증하기 위해 ImageNet과 Places365 데이터 셋으로 학습된 VGG16, Resnet50, Resnet18모델에 실험을 진행하였다. 그 결과 다양한 모델에서 2%이하의 정확도 손실과 높은 압축 효율을 달성했다. 또한, 재학습 과정을 제외한 압축방법인 No Fine-tuning Pruning(NFP)와 ThiNet과의 성능비교 결과 2배 이상의 압축효율이 있음을 검증했다.

Abstract

Recently, various lightweight methods for using Convolutional Neural Network(CNN) models in mobile devices have emerged. Weight quantization, which lowers bit precision of weights, is a lightweight method that enables a model to be used through integer calculation in a mobile environment where GPU acceleration is unable. Weight quantization has already been used in various models as a lightweight method to reduce computational complexity and model size with a small loss of accuracy. Considering the size of memory and computing speed as well as the storage size of the device and the limited network environment, this paper proposes a method of compressing integer weights after quantization using a video codec as a method. To verify the performance of the proposed method, experiments were conducted on VGG16, Resnet50, and Resnet18 models trained with ImageNet and Places365 datasets. As a result, loss of accuracy less than 2% and high compression efficiency were achieved in various models. In addition, as a result of comparison with similar compression methods, it was verified that the compression efficiency was more than doubled.

Keyword : Deep Learning Model Parameter Quantization, Weight compression, Lightweight model

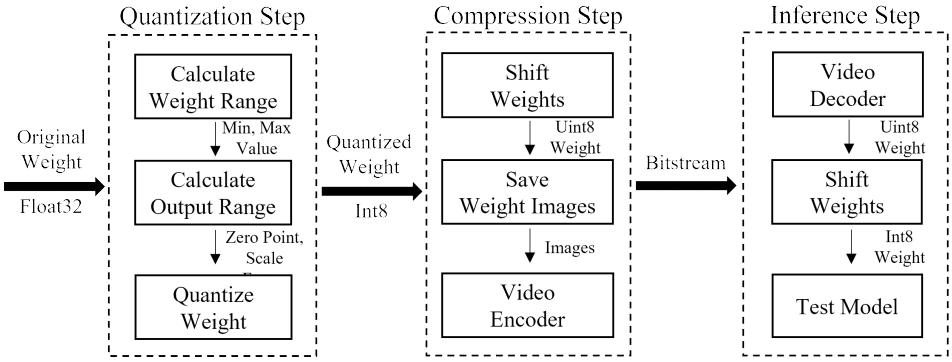

그림 1. 제안하는 방법의 전체 구조도
Fig. 1. Architecture of proposed method

Ⅰ. 서 론

최근 모바일 기기에서의 딥러닝 모델 사용에 대한 수요가 증가하면서 모델 경량화의 필요성이 증가하고 있다. 모바일 환경은 연산성능, 메모리, 저장 공간, 네트워크 대역폭 등이 제한되어 딥러닝 모델을 사용하기 위해서는 경량화할 필요가 있다. 또한, 딥러닝 모델은 주로 행렬의 곱셈연산으로 구성되는데, 대다수의 모델은 32비트의 실수형 변수(float32)를 통해 각 가중치를 표현한다. GPU 가속을 이용할 수 있는 서버환경과 달리 모바일 환경은 float32형 행렬의 곱셈에 최적화되어 있지 않아 사용이 제한된다. 이에 따라 딥러닝 모델을 구성하는 가중치를 32비트 실수형이 아닌 8비트 혹은 그 이하의 정수형 변수로 가중치를 변환하는 가중치 양자화(Weight quantization) 기법이 등장했다[1].

가중치 양자화는 모델을 구성하는 가중치의 표현 비트를 낮추는 과정과 활성 함수도 가중치에 맞게 조정하는 과정으로 구성되는 경량화 방법이다[2]. 가중치 양자화는 다양한 모델에 적용할 수 있으며 모델의 크기, 연산복잡도와 메모리 요구량을 감소시켜 모바일 환경에서도 사용할 수 있도록 만든다[3]. 8비트 정수로 양자화하는 경우 모델 가중치의 크기는 25%로 압축되지만, 여전히 전체적인 크기가 크고 모바일 네트워크 환경에서 전송하기에 부담되기 때문에 추가적인 압축이 요구된다.

모바일 환경에서 딥러닝 모델을 사용하기 위해 모델의 크기와 연산복잡도를 낮추려는 연구는 다양하게 진행되었다. 모델의 경량화는 정확도에 주는 영향이 적은 가중치나 연산을 간소화 혹은 제거하는 것으로 정확도는 보존하며, 모델의 요구 자원을 줄이는 것을 목표로 한다. Deep Compression[4]은 값이 작은 가중치를 0으로 변경하여 해당 값의 연산을 생략하는 Network pruning과 가중치를 공유하여 총 가중치의 수를 줄이는 Weight sharing을 적용하여 가중치를 압축하는 방법을 제안했다. Deep Compression은 유사한 가중치의 수를 줄이기 위해 K-means clustering 알고리즘을 사용했다. 그 결과 다양한 Convolutional Neural Network(CNN)모델을 정확도의 손실 없이 압축하는 성능을 나타냈다. Q-CNN[5]은 코드북의 형태로 가중치를 압축하고 압축된 가중치와 입력 데이터의 Sub-vector의 연산을 통해 원본 모델의 추론 결과를 근사하는 방법을 제안했다. 가중치를 Discrete Cosine Transform(DCT)[6]을 통해 주파수 영역으로 변환했을 때, 고주파 영역의 가중치를 제거하여 모델을 압축하거나[7] 모델의 연산 과정을 주파수 영역에서 진행하여 압축된 상태의 가중치를 그대로 추론에 사용하는 경량화 방법도 제안되었다[8].

본 논문은 영상압축 코덱을 이용하여 양자화된 정수 가

a) 성균관대학교 컴퓨터교육과(Department of Computer Education, Sungkyunkwan University)
‡ Corresponding Author : 류은석(Eun-Seok Ryu)
E-mail: esryu@skku.edu
Tel: +82-2-760-0677
ORCID: http://orcid.org/0000-0003-4894-6105
※ This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2021-2017-0-01630) supervised by the IITP (Institute for Information & communications Technology Promotion).
· Manuscript received August 19, 2021; Revised October 18, 2021; Accepted October 18, 2021.

중치를 추가로 압축하는 방법을 제안한다. 그림 1은 제안하는 압축방법의 전체적인 구조를 나타낸다. 그림 1에서 Quantization Step은 원본 가중치의 범위와 출력결과를 계산하고 정수로 양자화하는 단계를, Compression Step은 정수 가중치를 비디오 인코더를 통해 압축하는 과정을, Inference Step은 압축된 모델을 다시 사용하기 위한 과정을 각각 나타낸다.

Ⅱ. 관련연구

본 장은 기존의 연구된 가중치 압축이나 경량화 방법에 대한 내용을 설명한다. 본 장은 먼저 기존의 방법들이 포함하고 있는 Fine-tuning의 문제점에 대해 서술하고, Fine-tuning을 제거한 압축 방법을 소개한다. 그림 2는 본 장에서 소개하는 방법들과 본 논문이 제안하는 방법을 간소화하여 비교한 그림이다.

1. 기존의 압축 방법의 Fine-tuning 과정

서론에서 언급한 가중치의 압축방법들은 다양한 방법을 통해 가중치의 연산복잡도와 메모리 요구량을 크게 낮추는 높은 압축 성능을 보였지만, 정확도 손실을 최소화하기 위해 Fine-tuning 혹은 Retraining이라고 하는 재학습 과정을 포함하고 있다. 특히, Deep Compression[4]의 경우 pruning 이후 양자화를 거쳐 압축하는 방법으로 그림 2의 (a)처럼 다수의 재학습과정을 포함하고 있다. Model Pruning을 통한 경량화 방법은 경량화 이후 Fine-tuning 과정을 포함하고 있다[9]. 위 압축방법들은 Fine-tuning 과정을 제외하고 성능을 평가할 경우 큰 정확도 감소가 나타난다.

Fine-tuning 과정은 구체적으로 경량화 이후 네트워크의 일부 혹은 전체를 재학습하는 과정을 의미한다. Fine-tuning 과정도 모델의 학습 과정처럼 학습에 사용할 데이터 셋을 필요로 하는데, 이 데이터 셋은 학습에 사용된 데이터 전체 혹은 특성이 유사한 데이터를 사용한다[10]. Fine-tuning 과정은 경량화된 모델의 정확도를 높이지만, 두 가지 문제점이 있다. 최근 연구되는 딥러닝 모델의 학습 데이터 셋은 대용량의 데이터이다. 예를 들어 이미지 분류 데이터 셋인 ImageNet 데이터 셋[11]의 학습 데이터의 크기는 138GB로 모바일 환경은 물론 대부분의 PC환경에서도 부담되는 크기이다. 또한, 모든 데이터 셋이 모두가 언제나 접근 가능한 것은 아니다. 개인이 수집한 데이터 셋이나 민감한 개인정보를 포함한 데이터 셋의 경우 외부에서의 접근이 제한되어 Fine-tuning 역시 제한된다. 최근 Fine-tuning 과정을 간소화하거나 최적화하는 다양한 방법들이 제

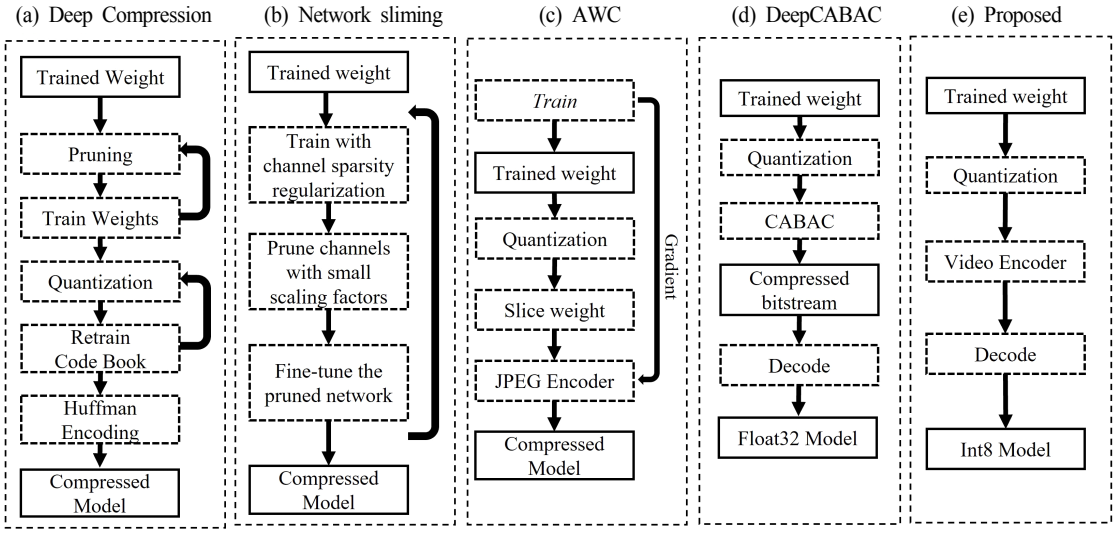

그림 2. 다른 가중치 압축 방법들(a~d)과 제안하는 방법(e)의 개요도
Fig. 2. Diagram of the weight compression methods (a~d) and the proposed method (e)

안되었지만[12], 딥러닝 모델의 학습은 여전히 많은 시간을 소모하는 과정으로 모델을 압축하는 단계에 비해 연산복잡도가 큰 과정이다. 따라서 압축 알고리즘을 범용적으로 사용하기 위해서는 Fine-tuning 과정이 간소화 혹은 제거되어야 할 필요성이 있다.

2. No Fine-tuning Pruning

Pruning 기법은 일부 레이어를 제거하거나 레이어 내부의 일부 채널을 제거하여 연산을 생략하는 경량화 기법이다. 레이어의 채널을 제거하는 경우, scale factor(γ)를 L1 Loss[13]를 통해 모델의 손실함수에 반영하여 학습한다[14]. 아래는 Pruning을 위한 손실함수를 나타낸 것이다.

$$L=\sum_{(x,y)} l(f(x,W),y)+\lambda\sum_{\gamma\in\Gamma} g(\gamma) \tag{1}$$

$\sum_{(x,y)} l(f(x,W),y)$는 기존 손실함수를 나타내며, $g(\gamma)$는 scale factor(γ)의 L1 손실함수를 나타낸다. Network Sliming은 γ를 정하는 방법으로 별도의 정보를 사용하지 않고 Batch Normalization(BN)[15,16]의 factor를 가져오는 방법을 제안했다[14]. 딥러닝 모델에서 데이터를 입력하는 단위인 Batch별 입력 값과 출력 값의 분포를 정규화하기 위한 BN의 출력(y)은 다음 수식으로 나타낼 수 있다.

$$y=\gamma\frac{x-\mu}{\sqrt{\sigma^2+\varepsilon}}+\beta \tag{2}$$

위 수식에서 scale factor(γ)와 shift factor(β)는 학습 가능한 변수로 보정 정도를 조절한다. BN은 딥러닝 모델에서 하나의 레이어로 동작하는데, Network Sliming은 모델에 포함된 BN의 학습된 γ를 pruning 과정에서 손실함수의 scale factor로 사용하여 추가적인 정보를 줄이는 Pruning 방법이다. Network Sliming을 VGG16 모델[17]에 적용한 결과 정확도 감소는 0.1%이하로 유지하면서 크기를 50% 압축했다. 하지만 Network Sliming은 Pruning 이후 Fine-tuning을 수행하는 과정을 여러 번 반복하는데, 위에서 서술한 것처럼 압축과정에서 Fine-tuning은 여러 문제점을 가지고 있다. 이를 해결하기 위해 No Fine-tuning Pruning(NFP)은 scale factor(γ)뿐만 아니라 shift factor(β)를 반영한 Pruning 방법을 제안했다[18]. NFP는 기존 방법에서 $\beta_{A_i^n}$를 추가로 저장하여 Fine-tuning을 생략하고 정확도를 유지했다. NFP를 Resnet50[19]에 적용한 실험결과 모델 크기의 34%를 압축하고 73.12%의 정확도를 나타냈다.

NFP는 재학습 과정 없이 연산에 사용하는 가중치의 수를 직접 줄여 모델을 경량화할 수 있지만, Batch Normalization Layer의 값인 γ와 β를 이용하기 때문에 Batch Normalization을 포함하지 않는 모델에는 적용할 수 없다는 문제점이 있다. 본 논문에서 제안하는 방법은 가중치의 수를 줄이는 Pruning이 아닌 각 가중치의 크기를 줄이는 Weight Quantization을 기반으로 압축하여 Batch Normalization을 포함하지 않는 VGG16 모델에도 적용할 수 있다.

3. 적응적 가중치 압축

순방향 신경망 모델(Feedforward neural networks)의 학습은 마지막 레이어 출력의 오류를 최소화하도록 각 가중치를 수정하는 과정이다. 각 가중치(w_{ji})의 수정은 역전파 알고리즘에 의해 다음과 같이 계산된다.

$$\Delta w_{ji}=-\eta\partial E \tag{3}$$

E은 모델 전체의 오류를, η는 Learning rate를 나타낸다. 이때 Gradient($\partial E/\partial w_{ji}$)는 가중치의 변화에 따른 모델의 오류의 변화 비율을 나타낸 것으로 해석된다. 적응적 가중치 압축(Adaptive Weight Compression, AWC)은 Fully-Connected Layer(FCL)들로 구성된 Multi-Layer Perceptron (MLP) 모델을 이미지 압축 기술인 JPEG을 사용하여 압축하는 방법이다[21]. 학습 과정에서 계산된 가중치의 Gradient 값을 활용하면 해당 가중치의 손실이 성능에 미치는 영향을 추정할 수 있고, 각 가중치의 손실률을 조절할 수 있다. AWC는 JPEG 압축과정에서 품질과 손실률을 조절하는 Quality Factor(QF)[20]를 Gradient에 따라 변경하여 효율적인 압축 성능을 달성했다. AWC는 가중치의 행렬에서 64개 단위로 가중치를 자른 다음, 8×8의 형태로 변형한다. 이후

각 8×8의 block들을 JPEG 알고리즘을 이용하여 압축하는데, 이때 각 block 내의 Gradient 값의 평균이 임계값 이하면 낮은 QF를, 그렇지 않으면 높은 QF를 적용하여 압축한다. 각 Block의 사용된 QF는 이진 값으로 QF map을 통해 저장한다.

하지만 일반적으로 학습이 완료된 가중치를 저장할 때, Gradient 값은 학습과정이 종료되고 성능을 실험하거나 가중치를 저장할 때 제거된다. 따라서 AWC를 비롯한 가중치의 Gradient를 활용하는 압축 방법들은 학습과정부터 이후 압축 과정에서 Gradient 값이 활용될 것을 가정하고 Gradient를 보존하여 이용한다. 만약 Gradient가 제거된 가중치를 압축하기 위해서는 Gradient를 구하는 과정이 필요한데, 이는 학습 과정과 마찬가지로, 높은 연산 자원을 요구한다. 또한 Gradient는 가중치의 수만큼 존재하기 때문에 가중치의 크기만큼 Gradient를 위한 공간이 추가로 요구된다. 따라서 이 Gradient를 이용한 압축 방법은 모델의 학습부터 압축단계를 고려하여 Gradient를 준비하는 경우에는 효율적으로 압축할 수 있지만, 학습이 완료된 가중치만 존재하는 경우에는 높은 연산복잡도와 저장 공간을 필요로 하기 때문에 모바일 장치 등에서 범용적으로 사용하기는 어렵다. 제안하는 방법은 HEVC를 이용하여 한 레이어 전체를 한 번에 압축하며, 다양한 장치에서 압축을 수행할 수 있도록 Gradient 제외하고 압축을 수행한다.

4. DeepCABAC

Context-adaptive binary arithmetic coding(CABAC)은 영상 압축에 사용되는 산술 부호화 알고리즘 중 하나이다[24]. CABAC은 이진 코드를 압축하기 위해 압축하는 데이터의 통계를 계산하고 이를 반영하여 효율적인 압축을 수행한다[28]. DeepCABAC은 이러한 CABAC를 이용하여 원본 가중치의 분포를 추정하며 압축하는 방법이다[29]. 영상 압축을 포함한 손실압축 알고리즘은 부호화후 복호화된 신호의 왜곡률과 부호화된 신호의 길이를 계산하여 모드를 선택한다. 이때 왜곡률은 원본 신호와 복구된 신호의 차이를 계산한 값이다. 하지만 딥러닝 모델 압축에서는 원본 가중치와의 차이보다는 정확도가 중요한 지표로 작용한다. DeepCABAC은 복구된 모델의 정확도를 고려한 양자화 및 CABAC를 정의하고 이를 활용하여 가중치를 압축하는 방법이다. DeepCABAC은 fine-tuning을 포함하지 않고 높은 압축 성능을 나타냈다. 특히 VGG16을 8.7MB까지 정확도의 저하없이 압축하는 성능을 보였다. 하지만 DeepCABAC의 양자화는 CABAC을 통한 압축효율을 높이기 위해 최적화된 양자화 방법으로 실제 모델 사용에 최적화된 양자화 방법은 아니다. 또한, DeepCABAC은 float32로 구성된 원본 가중치를 압축하는 방법으로 복구된 가중치 역시 float32 가중치이다. 따라서 압축 후 복구된 모델의 연산복잡도는 원본 모델과 동일하여, 모바일 장비를 포함한 고성능 GPU를 사용할 수 없는 환경에서는 모델의 사용이 제한된다.

Ⅲ. 가중치의 양자화 방법

본 논문이 제안하는 압축 방법은 행렬의 형태로 나타난 가중치를 정수로 양자화하고, 이를 비디오 인코더를 통해 추가로 압축하는 방법이다. 모바일 장치에서 압축한 가중치를 복호화하여 이용하기 위해서는 양자화된 상태에서 사용하는 것이 중요한데, 본 장에서는 가중치를 정수로 양자화하는 방법들과 본 논문에서 사용하는 양자화 방법을 서술한다. 또한 본 논문이 제안하는 방법은 양자화 가중치를 비디오 인코더를 통해 압축하는데, 비디오 인코더는 8비트, 10비트, 12비트를 지원한다[24]. 하지만 가중치의 양자화는 4비트의 배수로 구성하는 것이 효율적이기 때문에 4비트, 8비트, 16비트를 주로 사용한다[1,2]. 따라서 본 논문은 8비트 가중치를 채택하는 것을 전제로 한다.

1. 학습 후 양자화

가중치 양자화는 적용하는 단계에 따라 학습 후 양자화(Post-training Quantization)와 양자화 인식 훈련(Quantization-aware training, QAT)으로 구분된다[22,23]. QAT는 가중치의 하락을 최소화하기 위해 학습 단계부터 양자화 후의 결과를 예측하며 학습을 진행하고 Post-training quantization은 float32형으로 학습이 완료된 가중치에 직접 양자화를 수행한다. 정확도의 경우 학습단계부터 양자화를 고려한 QAT가 더 높게 나타난다. 따라서 모델을 처음부터 학습시키거

나 일부분을 재학습하는 경우에는 QAT를 적용하는 것이 좋지만, 재학습 과정은 관련 연구에서 언급한 문제점이 있다. 본 논문은 다양한 환경에서 압축할 수 있도록 하는 것이 목적이기 때문에 학습 후 양자화를 적용한다.

2. 텐서별 양자화

32비트 실수형의 원본 가중치 W와 양자화된 정수 가중치 W_q의 관계는 다음과 같이 표현된다. Z는 Zero point를, S는 가중치의 범위를 조절하는 Scale 값을 나타낸다[1].

$$W = (W_q - Z) \times S \tag{4}$$

양자화에 사용된 Z와 S는 활성함수와 행렬 연산을 보정하는 과정에 이용한다. 가중치 양자화는 해당 변수들을 계산하는 단위에 따라 채널별 양자화(Per-Channel quantization)와 텐서별 양자화(Per-Tensor Quantization)로 구분된다. 채널별로 Z와 S를 저장하면 텐서 전체에 하나의 Z와 S를 적용할 때보다 내적연산에 대해 더 세밀하게 연산이 가능하며 그 결과 압축 후 정확도가 증가한다[22]. 하지만 채널의 수가 증가하면 증가한 채널의 수만큼 Z와 S의 수도 증가하기 때문에 압축 효과를 감소시킨다. 또한, 보정과정은 실수연산을 포함하고 있어 채널별로 해당 연산을 수행하는 것은 연산복잡도가 증가하며 모바일 기기에서의 성능을 감소시킨다. 따라서 본 논문은 텐서별 양자화를 적용하여 추가적인 자원을 요구하지 않도록 한다.

3. 양자화의 범위

앞에서 언급한 가중치의 양자화 과정은 처리하는 가중치의 범위에 따라 비대칭(Asymmetric)과 대칭(Symmetric) 양자화로 구분된다[2]. W_{max}와 W_{min}은 각각 원본 가중치의 최댓값과 최솟값을 나타내며, 절댓값은 W_{max}가 더 크다고 가정한다. 대칭 양자화의 경우, Z를 0으로 고정하여 절댓값이 더 큰 W_{max}부터 $-W_{max}$의 가중치 범위를 정수로 사상한다. 비대칭 양자화의 경우, 대칭점을 0에서 Z만큼 이동하여 원본 가중치의 범위인 W_{min}부터 W_{max}의 가중치 범위를 정수로 사상한다. 비대칭 양자화를 적용할 경우, 대칭 양자화에 비해 정확한 범위를 변환하기 때문에 분산이 더 높게 나타나고 정확도도 더 높게 나타난다[2]. 하지만 가중치의 Z가 0이 아닐 때 Z와 Sclae 값을 곱하는 연산이 발생한다. 이 값을 0으로 만드는 대칭 양자화를 수행하면 해당 연산을 생략하여 연산 복잡도를 낮출 수 있다. 따라서 본 논문은 FCL에 포함된 가중치에 대칭 양자화를 적용한다.

표 1은 Imagenet으로 학습된 VGG16 모델을 위에서 언급한 양자화 방법들을 적용하여 실험한 표이다. 표 1에서 FCL1은 VGG16의 첫 번째 FCL에서 Bias를 제외한 Weight에 필요한 S와 Z의 수를 나타낸 것이다. 표 1에서 나타난 것처럼 양자화 방법에 따라 정확도 차이는 0.1% 이하로 나타나지만, 텐서별 대칭 양자화를 수행하면 추가 정보를 최소화할 수 있다. 따라서 본 논문은 텐서별 대칭 양자화를 적용한다.

Ⅳ. HEVC를 통한 양자화 가중치 압축

본 논문이 제안하는 가중치의 압축과정은 그림 3과 같다. 먼저 학습된 원본 가중치를 8비트 정수형으로 변환하는 텐서별 양자화를 수행한다. 이후 8비트 정수로 구성된 각

표 1. VGG16 양자화 방법에 따른 추가 정보과 정확도
Table 1. Additional data and accuracy of the VGG16 quantization method

Quantization Method		Accuracy		Additional Data		FCL1 Weight	
Unit	Range	Top 1	Top 5	Scale	Zero Point	Scale	Zero Point
Per-Tensor	Asymmetric	70.81%	90.06%	1 per tensor	1 per tensor	1	1
Per-Tensor	Symmetric	70.71%	89.95%	1 per tensor	0	1	0
Per-Channel	Asymmetric	70.71%	90.03%	1 per channel	1 per channel	4096	4096
Per-Channel	Symmetric	70.72%	90.03%	1 per channel	0	4096	0

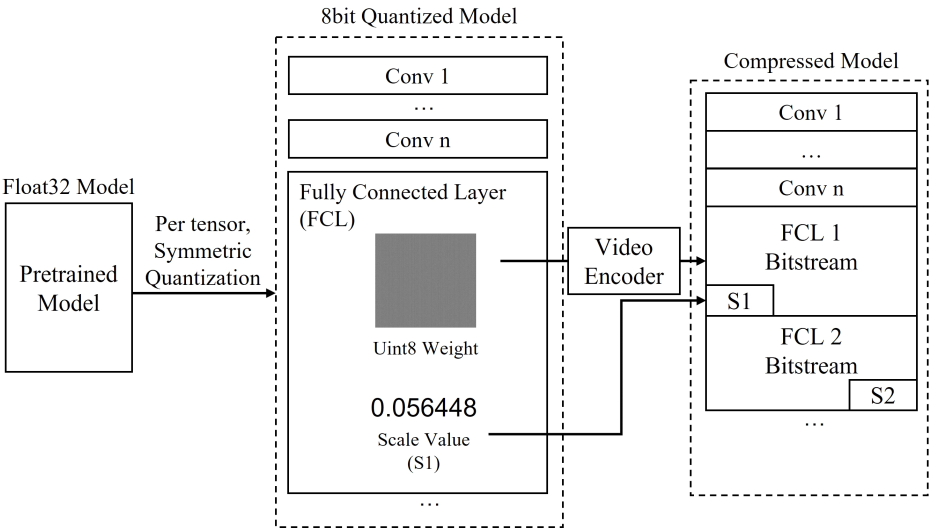

그림 3. 제안하는 압축 방법의 구조
Fig. 3. Structure of the proposed compression method

FCL의 가중치를 이미지들로 취급하여 High Efficiency Video Coding(HEVC) 인코더를 통해 압축한다[24]. HEVC는 기존 H.264(AVC)를 개선한 비디오 압축 표준으로 H.264에 비해 평균적으로 약 50%의 비트율 감소율을 나타냈다[25]. 특히 HEVC는 CTU 크기의 증가와 Intrapicture prediction의 예측 방향 증가로 인해 8K 이상의 고해상도 영상을 압축할 때 AVC에 비해 높은 성능이 나타났다. 본 연구에서 압축하는 FCL은 2K 이상의 고해상도 이미지에 대응되기 때문에 HEVC를 이용하여 압축한다. 압축된 모델을 사용하는 경우는 HEVC 디코더를 통해 FCL을 디코딩하고 Convolutional Layer와 함께 사용한다. 또한, HEVC는 Quantization Parameter(QP)를 통해 압축률과 손실률을 조절할 수 있다[24]. 낮은 QP를 적용하면 원본을 보존하는 대신 압축률이 감소하고, 높은 QP를 적용하면 압축률은 증가하지만, 데이터의 손실이 커진다.

V. 실험 결과 및 분석

실험은 3개의 CNN 모델과 2개의 데이터 셋의 조합으로 총 6가지 가중치를 사용한다. 데이터 셋으로 ImageNet 데이터 셋[11]과 Places365[26] 데이터 셋을 사용한다. ImageNet 데이터 셋은 1000가지 종류의 물체로 구성되어 종류마다 50개의 검증 데이터를 제공한다. Places365는 365가지 장소의 사진으로 구성된 데이터 셋으로 종류별로 100개의 검증 데이터를 제공한다. 모델은 VGG16[17], Resnet50[19], Resnet18을 사용한다.

실험에 사용한 서버는 CPU로 Intel Xeon E5-2620v3 2개, 64GB 메모리로 구성되었다. 비디오 인코더의 경우 GPU를 통한 가속이 가능하지만, GPU 성능의 영향을 배제하고 성능을 측정하기 위해 GPU 가속은 사용하지 않고 실험을 진행하였다. 모델의 압축과 테스트 과정에서 라이브러리는 Pytorch를 사용한다. 실험 과정에서 성능은 모델의 추론 결과에서 확률이 가장 높은 값과 정답을 비교한 Top1 정확도와 확률이 가장 높은 5개의 값 중에서 하나라도 정답과 일치하는지 평가하는 Top5 정확도로 평가한다.

1. Convolution Layer의 압축

본 논문이 제안하는 방법은 그림 3에서처럼 FCL을 대상으로 추가압축을 수행한다. Convolution Layer의 경우 양자화 이후 추가 손실에 민감하게 정확도가 감소하여 8비트로 양자화된 상태 그대로 전송한다. Convolution Layer는 주로

표 2. 합성곱 레이어의 압축 성능
Table 2. Performance of convolutional layers

QP	Top1 Acc	Top5 Acc
Original	71.10%	90.20%
Quant	70.65%	89.93%
21	65.77%	86.94%
22	63.86%	85.85%
23	59.82%	82.86%

(3,3)에서 (7,7)의 작은 크기의 행렬로 구성되어 있다. 표 2는 VGG16 모델의 Convolution Layer의 필터를 연결하여 하나의 이미지를 생성한 후 압축한 실험 결과이다. 양자화만 수행한 경우보다 적은 QP로 원본을 보존하여 압축하여도 정확도가 크게 감소하는 것을 볼 수 있다. 따라서 본 논문에서는 Convoution Layer는 추가압축을 수행하지 않는다.

2. VGG16을 압축한 실험 결과

ImageNet데이터 셋으로 학습된 VGG16의 가중치는 사용한 딥러닝 라이브러리인 Pytorch에서 제공하는 사전 학습된 가중치를 사용했다. Places365[26]의 경우 사전 학습된 가중치의 형식이 실험에 사용하기 부적절하여 직 접 학습을 진행하였다. 학습 결과 Top1 정확도는 53.64%, Top5 정확도는 83.96%로 나타났다. VGG16에 대한 압축 실험 결과는 표 3과 같다.

ImageNet으로 학습한 경우 모델의 크기는 원본 모델의 6.25% 수준으로 압축되었으며 Top1 정확도 손실은 0.6% 이하로 나타났다. 이는 양자화만 수행한 경우보다 약 4배의 압축효율을 보였다. Places365로 학습한 경우는 원본 모델의 4.8% 수준의 크기로 약 0.2%의 Top1 정확도 손실이 나타났다. 특히 압축을 적용한 FCL의 경우 원본에 비해 2% 수준까지 압축된 것을 확인했다. 표 3과 같이 VGG16은 FCL의 비율이 높은 모델이기 때문에 다른 경우에 비해 높은 압축효율이 나타났다. 또한, QP를 36으로 설정할 경우 정확도 손실은 1.36%로 더 높게 나타나지만, FC1은 QP가 32일 때에 비해서 60%의 크기가 감소했다.

3. Resnet50을 압축한 실험 결과

Resnet50은 ImageNet 데이터 셋과 Places365 데이터 셋 모두 사전 학습된 가중치를 사용하여 실험을 진행했다. 실험 결과는 표 4와 같다. Resnet50의 경우 VGG16보다 FCL의 크기가 작은 모델로 모델전체의 압축효율은 VGG보다 작게 나타났다. 하지만 FCL의 경우 원본에 비해 ImageNet에서 5.3%, Places365에서 6.3%수준으로 크기가 압축되며 정확도 손실은 약 1.5%로 나타났다. Resnet50에서 압축한 FCL은 모델의 출력 레이어로 정확도에 대한 영향이 높은 레이어이기 때문에 정확도 손실도 VGG16에 비해 상대적으로 높게 나타났다.

표 3. VGG16 압축 성능 실험 결과
Table 3. Compression results using VGG16

QP	VGG16 + ImageNet					VGG16 + Places365				
	Top1 Acc	Top5 Acc	FC1	FC2	Model Size	Top1 Acc	Top5 Acc	FC1	FC2	Model Size
Original	71.10%	90.20%	392 MB	64 MB	528 MB	53.64%	83.96%	392 MB	64 MB	518 MB
Quant	70.75%	90.00%	98 MB	16 MB	132 MB	53.33%	83.84%	98 MB	16 MB	129 MB
22	70.66%	89.93%	32.9 MB	8.5 MB	59 MB	53.53%	83.87%	33 MB	7 MB	55 MB
24	70.59%	89.97%	28.3 MB	7.7 MB	54 MB	53.53%	83.85%	28 MB	6 MB	50 MB
26	70.59%	89.90%	23.7 MB	7.0 MB	49 MB	53.61%	83.87%	24 MB	6 MB	45 MB
28	70.59%	89.86%	19.4 MB	6.2 MB	44 MB	53.58%	83.89%	20 MB	5 MB	40 MB
30	70.51%	89.89%	14.9 MB	5.5 MB	38 MB	53.57%	83.82%	15 MB	4 MB	35 MB
32	70.54%	89.85%	10.6 MB	4.7 MB	33 MB	53.53%	83.77%	11 MB	3 MB	30 MB
34	70.24%	89.78%	7.0 MB	3.9 MB	29 MB	53.43%	83.77%	7 MB	3 MB	25 MB
36	69.74%	89.62%	4.2 MB	3.1 MB	25 MB	53.31%	83.61%	4 MB	2 MB	22 MB

표 4. Resnet50 압축 성능 실험 결과
Table 4. Compression results with Resnet50

QP	Resnet50 + ImageNet				Resnet50 + Places365			
	Top1 Acc	Top5 Acc	FC	Model Size	Top1 Acc	Top5 Acc	FC	Model Size
Original	75.58%	92.78%	8003 KB	97.49 MB	54.77%	84.93%	2921 KB	92.5 MB
Quant	74.83%	92.37%	2000 KB	24.37 MB	54.28%	84.65%	737 KB	23.1 MB
22	74.45%	92.25%	525 KB	22.93 MB	53.84%	84.47%	186 KB	22.6 MB
24	74.05%	92.02%	430 KB	22.84 MB	53.18%	83.85%	152 KB	22.5 MB
26	73.37%	91.77%	339 KB	22.75 MB	52.93%	83.92%	119 KB	22.5 MB
28	72.17%	91.45%	253 KB	22.67 MB	51.02%	82.87%	88 KB	22.5 MB
30	67.83%	89.94%	167 KB	22.58 MB	47.11%	80.14%	55 KB	22.4 MB

4. Resnet18을 압축한 실험 결과

Resnet18은 ImageNet 데이터 셋과 Places365 데이터 셋 모두 사전 학습된 가중치를 사용하여 실험을 진행했다. Resnet50과 유사하게 표 5와 같이 나타났다. Resnet50과 Resnet18은 모델의 구조가 유사하지만 Convolution Layer의 깊이가 더 깊은 Resnet50에서 추출한 Feature가 더 많은 정보를 포함하고 있기 때문에 압축 후 감소하는 정확도의 크기가 크게 나타났다.

5. FCL의 압축효율 비교

본 논문은 모델 전체를 양자화한 후 FCL을 영상압축 코덱을 통해 압축하는 방법을 제안한다.

실험 결과 모델에 따라 FCL의 비중이 다르므로 모델 전체의 압축률은 다르게 나타났다. 그림 4는 모델별 FCL의 크기와 Top1 정확도를 나타낸 것이다. 그림 4에서 FCL은 모든 모델에서 원본 모델(Original)에 비해 낮은 정확도 손실과 높은 압축률을 나타내고 있으며, 양자화만 실시한 경우(Quant)에 비해서도 크기가 감소하고 있다. QP에 따른 정확도 손실은 모델에 따라 급격하게 변화했다. VGG16의 경우 모든 QP에서 2% 이하의 정확도 차이가 나타났지만, Resnet50와 Resnet18의 경우에는 QP가 증가함에 따라 정확도가 급격하게 감소하는 현상이 나타났다.

6. ThiNet과 NFP와의 성능 비교 결과

본 절에서는 Fine-tuning 과정을 포함하지 않는 딥러닝 모델 압축방법인 NFP[18], ThiNet[27]과 성능을 비교한다. 실험에서 사용한 모델은 ImageNet으로 학습된 Resnet50이며 각각 압축된 가중치의 크기와 정확도를 비교한다. Baseline은 각 압축방법 실험에서 사용한 원본 모델로 학습 방법이나 테스트 데이터에 따라 조금씩 다르게 나타났다. 본 실험에서 사용한 QP는 24로 표 4를 기준으로 정확도가 급감하기 전의 QP를 사용했다. 표 6은 비교 결과를 나타낸 것으로 같은 정확도에서 제안하는 방법의 모델 크기가 약 2배가량 작은 것을 확인할 수 있다.

표 5. Resnet18 압축 성능 실험 결과
Table 5. Compression results using Resnet18

QP	Resnet18 + ImageNet				Resnet18 + Places365			
	Top1 Acc	Top5 Acc	FC	Model Size	Top1 Acc	Top5 Acc	FC	Model Size
Original	68.93%	88.83%	2000 KB	44.59 MB	53.69%	83.78%	731 KB	43.34 MB
Quant	68.25%	88.41%	500 KB	11.70 MB	53.10%	83.51%	185 KB	10.83 MB
22	68.06%	88.26%	210 KB	10.86 MB	52.67%	83.41%	72 KB	10.72 MB
24	67.81%	88.18%	187 KB	10.84 MB	52.56%	83.20%	64 KB	10.72 MB
26	67.34%	88.09%	167 KB	10.82 MB	52.13%	83.00%	55 KB	10.71 MB
28	66.66%	87.72%	143 KB	10.79 MB	51.57%	82.55%	46 KB	10.70 MB

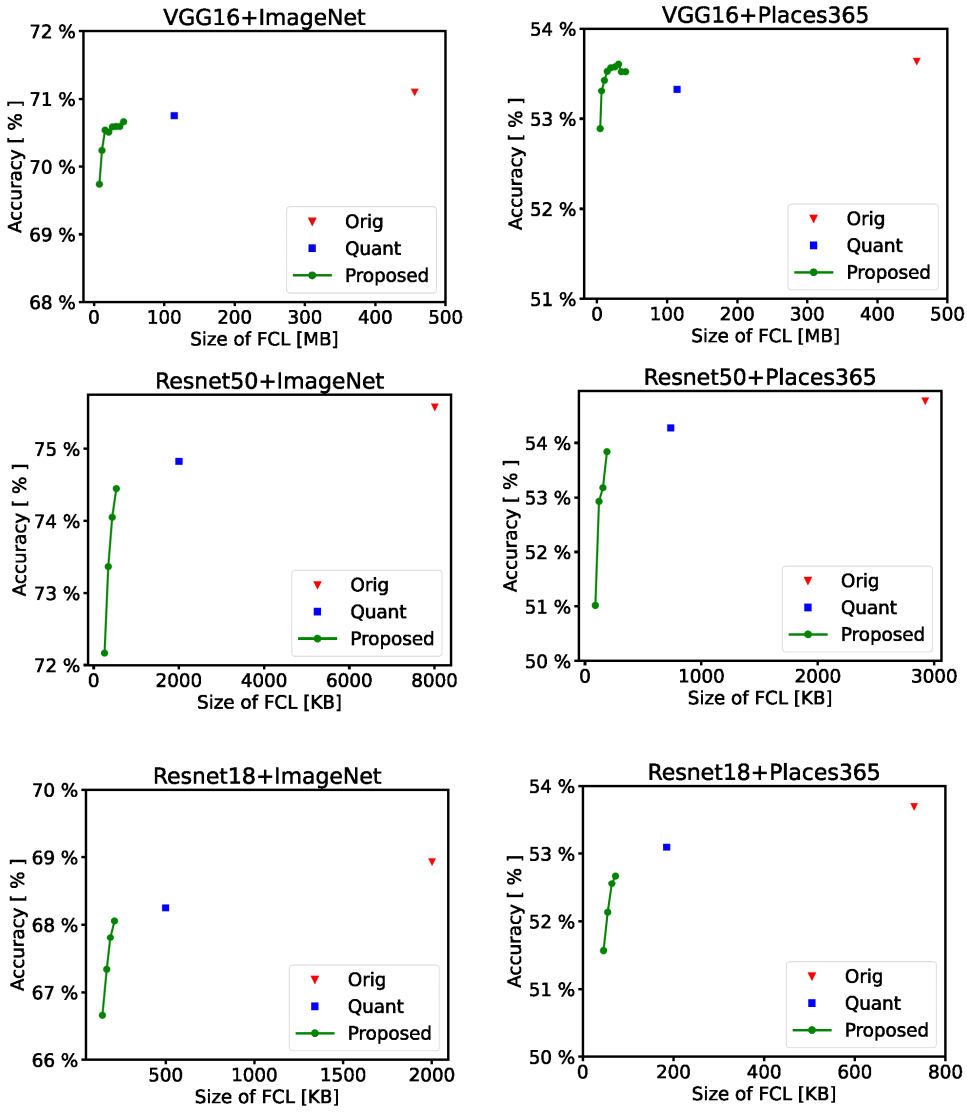

그림 4. FCL의 크기와 Top1 정확도
Fig. 4. Size of FCL and top1 accuracy

표 6. ThiNet과 NFP와의 성능 비교 결과
Table 6. Comparison result with ThiNet and NFP

Model		Top1 Acc	Model Size
ThiNet-70	Baseline	72.88%	97.49 MB
	Compressed	66.28%	64.03 MB
NFP	Baseline	73.45%	97.49 MB
	Compressed	73.12%	64.03 MB
Proposed (QP:24)	Baseline	75.58%	97.49 MB
	Compressed	74.05%	22.84 MB

7. AWC와의 성능 비교

위 관련연구에서 언급한 내용처럼 AWC는 효율은 가중치의 Gradient에 의존한다. 그림 5는 Gradient 정보가 없을 때 Resnet50 모델의 FCL에 AWC가 제안하는 JPEG을 통해 압축 방법(Single QF)과 본 논문이 제안하는 압축 방법의 성능을 비교한 그래프이다. 두 압축 방법 모두 본 논문에서 제안하는 양자화 방법을 동일하게 적용했을 때, 본 논문

에서 제안하는 방법의 성능이 높게 나타났다.

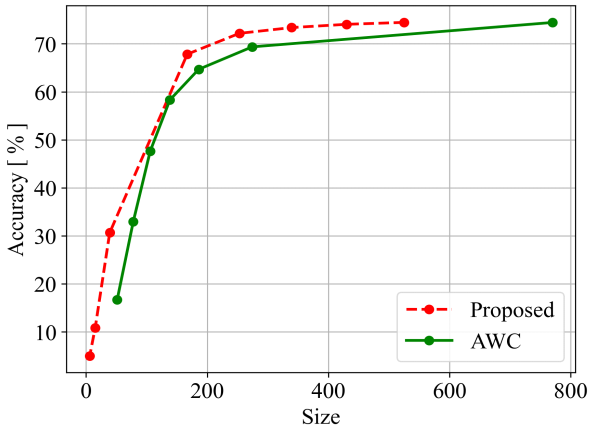

그림 5. AWC와 제안하는 방법의 성능비교
Fig. 5. Performance comparison between AWC and proposed method

또한, 본 논문이 제안하는 방법은 학습이 완료된 가중치 파일만 이용해서 압축이 가능하며, Block 단위로 나누는 등의 연산은 하드웨어 수준에서 이미 최적화된 비디오 코덱의 기능을 사용하여 다양한 기기에서 사용이 가능하다.

Ⅵ. 결 론

본 논문은 CNN 모델의 가중치의 크기를 압축하는 방법으로 8비트 정수로 양자화된 가중치를 이미지들로 처리하여 비디오 인코더를 통해 압축하는 방법을 제안한다. 이 방법은 원본 모델의 복잡도와 메모리 요구량은 8비트 정수로 양자화를 통해 감소시키고, 기기에 저장하거나 전송하기 위해 가중치의 크기를 줄이는 과정은 비디오 코덱으로 압축한다. 기존의 모델 압축방법들은 높은 압축률과 낮은 정확도 감소를 달성하기 위해 재학습 과정을 통해 정확도를 복원하는 과정을 포함한다. 하지만 재학습 과정은 모델의 학습에 사용된 데이터 셋이나 데이터 셋의 통계정보가 필요하다는 단점이 있었다. 이를 해결하기 위해 본 논문은 압축하는 과정에서 재학습과정이나 보정 과정을 제거하여 연산성능이 모바일 환경에서도 압축할 수 있으며 다양한 모델에 사용이 가능함을 실험을 통해 검증했다. 또한, 이 압축 방법은 압축된 모델의 비트스트림을 비디오 디코더를 통해 디코딩하고 정수 가중치를 바로 사용할 수 있다. 다양한 조건에서 실험한 결과로, 정확도를 심하게 감소시키지 않으면서 FCL의 크기는 원본 모델의 10% 이하로 압축되는 결과가 관찰되었다.

참 고 문 헌 (References)

[1] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, “Deep learning with limited numerical precision,”, International conference on machine learning. PMLR, 2015, pp. 1737 - 1746.
[2] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, “Quantization and training of neural networks for efficient integer-arithmetic-only inference,” Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 2704 - 2713.
[3] R. David, J. Duke, A. Jain, V. J. Reddi, N. Jeffries, J. Li, N. Kreeger, I. Nappier, M. Natraj, S. Regev et al., “Tensorflow lite micro: Embedded machine learning on tinyml systems,” arXiv preprint arXiv:2010.08678, 2020.
[4] 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, 2015.
[5] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, “Quantized convolutional neural networks for mobile devices,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 4820 - 4828.
[6] N. Ahmed, T. Natarajan, and K. R. Rao, “Discrete cosine transform,” IEEE transactions on Computers, vol. 100, no. 1, pp. 90 - 93, 1974.
[7] S. Kim, E.-S. Park, M. Ghulam, and E.-S. Ryu, “Compression method for cnn models using dct,” Proceedings of the Korean Society of Broadcast Engineers Conference. The Korean Institute of Broadcast and Media Engineers, 2020, pp. 553 - 556.
[8] Y. Wang, C. Xu, S. You, D. Tao, and C. Xu, “Cnnpack: Packing convolutional neural networks in the frequency domain.” NIPS, vol. 1, 2016, p. 3.
[9] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, “Pruning filters for efficient convnets,” arXiv preprint arXiv:1608.08710, 2016.
[10] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, “A survey of quantization methods for efficient neural network inference,” arXiv preprint arXiv:2103.13630, 2021.
[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” 2009 IEEE conference on computer vision and pattern recognition. Ieee, 2009, pp. 248 - 255.
[12] J. Frankle and M. Carbin, “The lottery ticket hypothesis: Finding sparse, trainable neural networks,” arXiv preprint arXiv:1803.03635, 2018.
[13] M. Schmidt, G. Fung, and R. Rosales, “Fast optimization methods for l1 regularization: A comparative study and two new approaches,” European Conference on Machine Learning. Springer, 2007, pp. 286 - 297.
[14] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, “Learning efficient convolutional networks through network slimming,” Proceedings o

f the IEEE international conference on computer vision, 2017, pp. 2736 – 2744.
[15] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep netw ork training by reducing internal covariate shift,” International confere nce on machine learning. PMLR, 2015, pp. 448 – 456.
[16] S. Ioffe, “Batch renormalization: Towards reducing minibatch depende nce in batch-normalized models,” arXiv preprint arXiv:1702.03275, 20 17.
[17] K. Simonyan and A. Zisserman, “Very deep convolutional networks fo r large-scale image recognition,” arXiv preprint arXiv:1409.1556, 201 4.
[18] R. Liu, J. Cao, P. Li, W. Sun, Y. Zhang, and Y. Wang, “Nfp: A no finet uning pruning approach for convolutional neural network compressio n,” 2020 3rd International Conference on Artificial Intelligence and Bi g Data (ICAIBD). IEEE, 2020, pp. 74 – 77.
[19] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” Proceedings of the IEEE conference on computer vision a nd pattern recognition, 2016, pp. 770 – 778.
[20] G. K. Wallace, “The jpeg still picture compression standard,” IEEE tra nsactions on consumer electronics, vol. 38, no. 1, pp. xviii – xxxiv, 199 2.
[21] J. H. Ko, D. Kim, T. Na, J. Kung, and S. Mukhopadhyay, “Adaptive we ight compression for memory-efficient neural networks,” Design, Auto mation & Test in Europe Conference & Exhibition (DATE), 2017. IEE E, 2017, pp. 199 – 204.
[22] H. Wu, P. Judd, X. Zhang, M. Isaev, and P. Micikevicius, “Integer quan tization for deep learning inference: Principles and empirical evaluatio n,” arXiv preprint arXiv:2004.09602, 2020.
[23] Y. Guo, “A survey on methods and theories of quantized neural networ ks,” arXiv preprint arXiv:1808.04752, 2018.
[24] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, “Overview of th e high efficiency video coding (hevc) standard,” IEEE Transactions on circuits and systems for video technology, vol. 22, no. 12, pp. 1649 – 1 668, 2012.
[25] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, “Overview of the h. 264/avc video coding standard,” IEEE Transactions on circuits and systems for video technology, vol. 13, no. 7, pp. 560 – 576, 2003.
[26] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, “Places: A 10 million image database for scene recognition,” IEEE transactions on pattern analysis and machine intelligence, vol. 40, no. 6, pp. 1452 – 1464, 2017.
[27] J.-H. Luo, J. Wu, and W. Lin, “Thinet: A filter level pruning method for deep neural network compression,” Proceedings of the IEEE internatio nal conference on computer vision, 2017, pp. 5058 – 5066.
[28] D. Marpe, H. Schwarz, and T. Wiegand, “Context-based adaptive binar y arithmetic coding in the h.264/avc video compression standard,” IEE E Transactions on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 620 – 636, July 2003.
[29] S. Wiedemann, H. Kirchhoffer, S. Matlage, P. Haase, A. Marban,T. Ma rinc, D. Neumann, A. Osman, D. Marpe, H. Schwarzet al.,“Deepcabac: Context-adaptive binary arithmetic coding for deep neuralnetwork com pression,”arXiv preprint arXiv:1905.08318, 2019.

저 자 소 개

김 승 환

- 2020년 2월 : 가천대학교 컴퓨터공학과 학사
- 2020년 3월 ~ 현재 : 성균관대학교 컴퓨터교육과 석사과정
- ORCID : https://orcid.org/0000-0002-7018-5114
- 주관심분야 : 멀티미디어 통신 및 시스템, 인공지능, 딥러닝 모델 경량화

류 은 석

- 1999년 8월 : 고려대학교 컴퓨터학과 학사
- 2001년 8월 : 고려대학교 컴퓨터학과 석사
- 2008년 2월 : 고려대학교 컴퓨터학과 박사
- 2008년 3월 ~ 2008년 8월 : 고려대학교 연구교수
- 2008년 9월 ~ 2010년 12월 : 조지아공대 박사후과정
- 2011년 1월 ~ 2014년 2월 : InterDigital Labs Staff Engineer
- 2014년 3월 ~ 2015년 2월 : 삼성전자 수석연구원/파트장
- 2015년 3월 ~ 2019년 8월 : 가천대학교 컴퓨터공학과 조교수
- 2019년 9월 ~ 현재 : 성균관대학교 컴퓨터교육과 부교수
- ORCID : https://orcid.org/0000-0003-4894-6105
- 주관심분야 : 멀티미디어 통신 및 시스템, 비디오 코딩 및 국제 표준, HMD/VR 응용분야