Journal of Korea Robotics Society (2022) 17(2):133-141
https://doi.org/10.7746/jkros.2022.17.2.133 ISSN: 1975-6291 / eISSN: 2287-3961

임베디드 시스템에서의 객체 분류를 위한 인공 신경망 경량화 연구

Neural Network Model Compression Algorithms for Image Classification in Embedded Systems

신 희 중[1] · 오 현 동†
Heejung Shin[1], Hyondong Oh†

Abstract: This paper introduces model compression algorithms which make a deep neural network smaller and faster for embedded systems. The model compression algorithms can be largely categorized into pruning, quantization and knowledge distillation. In this study, gradual pruning, quantization aware training, and knowledge distillation which learns the activation boundary in the hidden layer of the teacher neural network are integrated. As a large deep neural network is compressed and accelerated by these algorithms, embedded computing boards can run the deep neural network much faster with less memory usage while preserving the reasonable accuracy. To evaluate the performance of the compressed neural networks, we evaluate the size, latency and accuracy of the deep neural network, DenseNet201, for image classification with CIFAR-10 dataset on the NVIDIA Jetson Xavier.

Keywords: Deep Learning, Model Compression, Pruning, Quantization, Knowledge Distillation, Embedded System

1. 서 론

무인이동체가 자율적인 의사결정을 수행하기 위해서는 다양한 센서로부터 획득한 정보를 이용하여 주변의 상황을 실시간으로 파악하는 것이 중요하다. 특히, 센서 정보 중 카메라로부터 들어오는 영상 정보는 고차원의 정보를 담고 있어 유용하게 사용될 수 있다. 영상 처리 알고리듬을 이용하면 주어진 영상 데이터의 특징을 자동으로 추출하고 분석하여 객체 탐지 및 분류와 같은 주변 환경에 대한 고수준의 이해가 가능하다.

AlexNet[1]이 2014년 ImageNet[2] 객체 분류 대회에서 우승한 이후 합성곱 인공 신경망(Convolutional Neural Network)은 컴퓨터 영상처리 분야에서 독보적인 성능을 보이고 있다. 특히 잔차 연결(Residual connection) 구조를 이용하여 인공 신경망의 은닉 층(Hidden layer)의 개수가 많아짐에 따라 발생하는 기울기 소실 문제(Gradient vanishing problem)가 해결되었다[3]. 잔차 연결 구조가 기울기 소실 문제를 해결하면서 합성곱 인공 신경망은 높은 정확도를 얻기 위하여 더 많은 은닉 층과 가중치를 갖는 구조로 연구가 되고 있으며, 최근 Vision Transformer[4]의 경우 약 6억 3천만개의 가중치를 가진 인공 신경망을 사용한다.

이렇게 거대해진 인공 신경망은 연산 비용이 다른 알고리듬에 비해 상당히 높기 때문에 GPU와 같은 고성능 연산 장치가 필수적이다. 하지만 고성능 연산 장치는 저전력으로 구동되는 임베디드 시스템에 탑재하기에는 어려운 부분이 많다. 또한 임베디드 시스템 대부분이 제한된 메모리를 가지고 있기 때문에 인공 신경망의 탑재 자체가 불가능한 경우도 있다. 따라서 소형 자율 무인이동체와 같이 제한된 컴퓨팅 파워를 가졌지만 실시간성이 보장되어야 하는 분야 혹은 드론과 같이 제한된 비행 시간에서의 임무 수행을 위하여 효율적인 전력 관리를 요구하는 시스템[5]에서의 인공 신경망 구동을 위해서는 인공 신경망의 경량화 및 가속화가 필수적이다. 이에 따라 최근 인공 신경망을 경량화 및 가속화하여 임베디드 시스템에

Received : Mar. 7. 2022; Revised : Mar. 23. 2022; Accepted : Mar. 25. 2022
※ This work was supported by Theater Defense Research Center funded by Defense Acquisition Program Administration under Grant UD200043CD
1. Master Student, UNIST, Ulsan, Korea (godhj@unist.ac.kr)
† Associate Professor, Corresponding author: Mechanical Engineering, UNIST, Ulsan, Korea (h.oh@unist.ac.kr)

탑재하는 연구들이 활발히 진행되고 있다[6-8].

인공 신경망 경량화 및 가속화 알고리듬의 목표는 기존의 인공 신경망의 성능은 최대한 보존하면서도 가볍고 빠른 인공 신경망을 개발하는 것이다. 인공 신경망 경량화 알고리듬은 대표적으로 세가지가 있다. 첫번째는 가지치기(Pruning)로, 인공 신경망의 정확도에 큰 영향을 미치지 않는 가중치를 제거하는 알고리듬이다[9,10]. 두번째로는 양자화(Quantization)로, 인공 신경망 가중치의 데이터 정밀도를 타겟 하드웨어에 적합하게 낮추는 알고리듬이다[11]. 마지막으로 작은 인공 신경망 모델에게 큰 모델의 지식을 전달하여 작은 모델의 정확도를 향상시키는 지식 증류(Knowledge Distillation) 알고리듬이다[12]. 세가지 형태의 알고리듬 모두 인공 신경망을 기존의 정확도는 보존하면서도 효과적으로 경량화 및 가속화하는 것이 보고되었다[8,12-14]. 그러나 이 세가지 경량화 알고리듬을 복합적으로 임베디드 시스템에 적용하여 성능을 비교 및 분석한 연구는 많이 보고되지 않았다.

따라서, 본 논문에서는 인공 신경망 경량화 및 가속화 알고리듬을 이론적으로 정리하고 해당 알고리듬을 동일한 환경 및 합성곱 인공 신경망에 대해 복합적으로 적용하여 이미지 분류 문제에 적용하였다. 또한, 경량화 된 인공신경망의 성능을 임베디드 시스템인 NVIDIA Jetson Xavier의 CPU에서 비교 및 분석하였다.

2. 합성곱 인공 신경망 발전 추세

2012년 ImageNet에서 우승을 차지한 AlexNet은 11x11, 5x5, 3x3의 다양한 합성곱 필터의 크기를 조합하여 5개의 합성곱 층(Convolution layer)과 3개의 완전 연결 층(Fully connected layer)으로 총 8개의 층으로 구성되어 있다. AlexNet은 기존의 2개의 합성곱 층으로만 구성된 LeNet[15]에 비해 깊은 구조를 가지고 있으며 인공 신경망이 깊은 층 구조를 가지면 더 좋은 성능을 낼 수 있다는 것을 보였다[1]. VGGNet은 AlexNet의 연구 결과를 토대로 합성곱 인공 신경망의 구조를 체계적으로 더 깊게 구성하였다. VGGNet은 3x3의 합성곱 필터만 이용하여 19개의 깊은 합성곱 층을 구성하여 ImageNet대회에서 2위를 차지하였다[16]. 이후 VGGNet과 같이 신경망 은닉 층을 깊게 쌓음으로써 인공 신경망의 성능을 이끌어 내려는 시도가 다수 이루어졌다. 그러나 이는 인공 신경망을 특정 깊이 이상으로 깊게 구성하면 성능이 하락하는 기울기 소실 문제로 인해 한계에 도달했다.

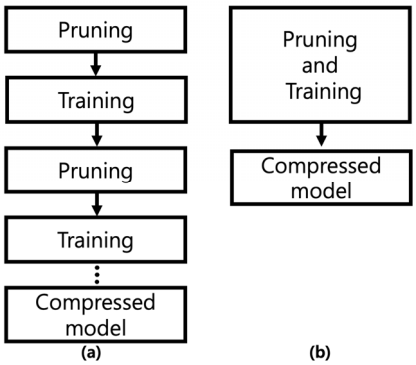
하지만 2015년 개발된 ResNet은 잔차 연결 구조를 이용하여 기울기 소실 문제를 해결하였다. ResNet은 잔차 연결 구조를 이용하여 기존 VGGNet에 비해 8배 더 깊은 인공 신경망을 구성함으로써 2015년 ImageNet 대회에서 우승을 차지하였다[3]. DenseNet은 이러한 잔차 연결 구조를 더 발전시켜 은닉 층으로부터 나온 특징점을 잔차 연결 구조를 이용하여 더하는 것이 아닌 특징점 자체를 병합(Concatenate)하였다[17]. 앞서 언급한 연구들은 이미지 분류의 정확도를 높이기 위해 더 깊고 많은 합성곱 필터 개수를 가지는 추세로 발전되어왔다. 하지만 이러한 인공 신경망들은 그 크기와 연산량 때문에 스마트폰과 같은 모바일 디바이스 혹은 임베디드 시스템에서 구동되기 어려운 점들이 많다. 따라서 본 연구에서는 앞서 언급한 합성곱 인공 신경망 중에서 가장 우수한 성능을 보이는 DenseNet계열의 인공 신경망에 대하여 경량화 및 가속화 알고리듬들을 적용하여 임베디드 시스템에서의 활용 가능성을 보였다.

3. 가지치기

인공 신경망은 수백만 개부터 많게는 수억 개의 이르기까지 매우 많은 가중치들로 구성되어있다. 학습이 완료된 인공 신경망은 다양한 실수 값을 가지는 가중치들로 구성되는데, 이러한 가중치들 중 절댓값이 작은 가중치들은 인공 신경망의 추론에 있어서 기여도가 낮은 것이 알려져 있다[18-20]. 따라서 기여도가 낮은 가중치들을 선별하여 제거하는 방식으로 인공 신경망을 경량화 할 수 있는데 이러한 알고리듬을 가지치기라고 한다.

가지치기 알고리듬은 크게 Unstructured Pruning과 Structured Pruning 두 가지로 나눌 수 있다. 전자는 인공 신경망의 가중치에 대하여 추론에 있어서 불필요하다고 생각되는 가중치의 값을 0으로 설정한다. 그 결과로 희소 가중치 행렬(Sparse Weight Matrix)이 생성되며 COOrdinate list (COO), Compressed Sparse Row (CSR), Compressed Sparse Column (CSC)과 같은 자료구조를 이용하여 인공 신경망을 압축할 수 있다. 후자는 인공 신경망의 개별 가중치가 아닌 여러가지 구성요소(합성곱 필터, 개별 은닉 층 등)에 대하여 적용하는 알고리듬으로서, 불필요한 가중치들이 많이 속해있는 구성 요소를 아예 삭제하는 방식으로 가지치기를 진행한다. 따라서 Structured Pruning을 이용하면 COO, CSR, CSC와 같은 자료구조를 이용하지 않더라도 인공 신경망 경량화가 가능하다. 하지만 해당 알고리듬은 합성곱 필터 혹은 특정 은닉 층 안에 있는 유의미한 가중치들 또한 함께 제거할 가능성이 있어서 일반적으로 Unstructured Pruning에 비하여 정확도를 보존하기는 어렵다.

가지치기 알고리듬을 인공 신경망에 적용함에 있어서 가장 중요하게 고려 되어야할 부분은 해당 모델의 정확도를 최대한 보존하는 것이다. 한 번에 너무 많은 가중치에 대해 가지치기를 적용하면 인공 신경망의 정확도가 심각하게 저하되기 때문



[Fig. 1] Flow chart of the pruning process: (a) shows a general pruning algorithm which takes long time to get compressed neural networks; and (b) combines pruning and training process to obtain compressed neural networks in a short time

에 대부분의 알고리듬은 [Fig. 1(a)]와 같이 소량의 가중치에 대해 알고리듬을 적용한 후 다시 재학습의 과정을 거친다. 재학습으로 정확도를 회복한 인공 신경망에 대해 다시 같은 방식으로 가지치기 알고리듬을 반복 적용하는 방식으로 최종 경량 모델을 생성하게 된다.

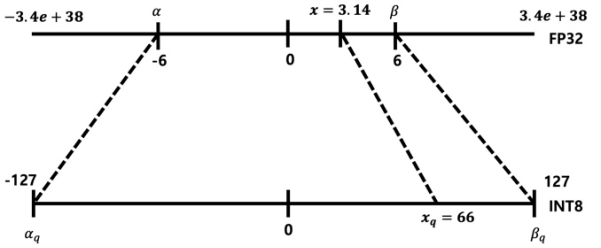
하지만 이러한 가지치기와 재학습이 반복되는 구조는 경량 인공 신경망을 생성하는데 소요되는 학습 비용이 크다는 단점이 있다. 이를 해결하기 위해 [Fig. 1(b)]와 같이 가지치기와 재학습을 통합한 Gradual Pruning 알고리듬이 제안되었다[13]. Gradual Pruning은 Unstructured pruning 계열의 알고리듬으로, 이를 이용하면 정확도를 최대한 보존하면서도 적은 학습 비용만으로 경량 인공 신경망을 생성할 수 있다. 해당 연구[13]에서는 식 (1)과 같은 Pruning Scheduling을 제안했다.

$$s_t = s_f + (s_i - s_f)\left(1 - \frac{t - t_0}{n\Delta t}\right)^3 \quad \text{for } t \in \{t_0, t_0 + \Delta t, \cdots, t_0 + n\Delta t\}. \tag{1}$$

여기서 t는 학습 중 가지치기 알고리듬이 적용되는 시점이며 t_0와 n을 지정함으로써 학습 중 가지치기 알고리듬이 적용될 구간을 정할 수 있다. s_i는 처음 인공 신경망에 적용할 희소 비율(Sparsity)로 일반적으로 0으로 설정한다. s_f는 최종적으로 인공 신경망이 가질 희소 비율이다. 식 (1)과 경사하강법을 통하여 가중치가 업데이트 되면 전체 가중치 중 $s_t\%$의 가중치가 가지치기 된다. 본 연구에서는 Gradual Pruning을 이용하여 DenseNet201을 정확도 하락 없이 경량화 하는 연구를 수행하였다.

4. 양자화

신경망의 가중치들은 오차 역전파(Backpropagation)와 경사하강법(Gradient descent)을 통하여 업데이트 되기 위해 정밀한 소수점 표현이 가능한 32비트의 부동 소수점(Floating point)으로 표현된다. 이러한 부동 소수점은 고정 소수점(Fixed point)에 비해 표현할 수 있는 수의 범위는 넓지만 실수 연산을 정수 연산으로 대체할 수 있는 고정 소수점 방식에 비해 연산 속도가 상당히 느리다. 따라서 학습이 완료된 인공 신경망의 부동 소수점 가중치들을 고정 소수점으로 변경함으로써 인공 신경망의 추론 가속화가 가능하다. 또한 32비트로 표현된 가중치들을 16비트 또는 8비트와 같이 낮은 비트로 표현하면 인공 신경망의 크기를 줄일 수 있다. 예를 들어 DenseNet201의 경우 약 1,800만개의 가중치를 가지고 있는데, 가중치가 32비트로 표현되었다면 70 MB의 모델 크기를 가지지만 8비트의 가중치를 이용한다면 19 MB의 크기만을 가지도록 할 수 있다. 이러한 컴퓨터 연산 및 구조의 특성을 이용하여 인공 신경망 경량화 및 가속화를 가능하게 만드는 알고리듬이 양자화다.



[Fig. 2] Symmetric linear quantization: the symmetric linear quantization algorithm maps 32bit floating point representation to 8bit integer representation

[Fig. 2]는 32비트의 부동 소수점의 값이 8비트의 정수 값으로 양자화 되는 과정을 보여준다. Symmetric linear quantization은 식 (2)~(3)을 이용하여 양자화를 진행한다.

$$x_q = clip\left(round\left(\frac{1}{s}x\right), \alpha_q, \beta_q\right) \quad \text{where } s = \frac{\beta - \alpha}{\beta_q - \alpha_q} \tag{2}$$

$$clip(x, a, b) = \begin{cases} a, (x < a) \\ x, (a \le x \le b) \\ b, (b < x). \end{cases} \tag{3}$$

여기서 x는 양자화 대상이 되는 가중치 혹은 활성화 함수의 입출력 값이며 α, β는 x의 범위다. Symmetric linear quantization의 경우 $\alpha = -\beta$의 관계를 가진다. s는 Scale factor로 서로 다른 비트 수로 표현된 값들을 변환할 때 사용되는 값으로 식 (2)를 이용하여 구할 수 있다.

양자화 알고리듬은 가지치기 알고리듬과 마찬가지로 기존 신경망의 정확도를 최대한 보존하는 것이 중요하다. 32비트에

서 16비트의 양자화의 경우 표현할 수 있는 수의 범위가 줄어들지만 인공 신경망의 정확도를 크게 하락시키지는 않는다. 하지만 8비트의 양자화의 경우 표현할 수 있는 수의 범위가 32비트에 비하여 매우 감소하므로 인공 신경망의 정확도가 크게 저하될 수 있다. 따라서 기존 인공 신경망의 정확도 보존을 위하여 양자화 알고리듬에서 가장 중요한 부분은 양자화를 위한 s값을 선택하는 것이다. 따라서 양자화 알고리듬은 해당 파라미터 s를 구하는 방식에 따라 [Table 1]과 같이 크게 세가지로 분류할 수 있다.

첫번째는 동적 양자화(Dynamic Quantization)로 주어진 가중치 행렬을 수식 (2)를 이용하여 양자화 한다. 활성화 함수의 입출력 값에 대한 양자화는 인공 신경망을 실제 작동 환경에서 구동함으로써 실시간으로 s를 계산하여 진행한다. 동적 양자화는 인공 신경망이 실행될 때마다 활성화 함수의 입출력 값에 대한 양자화를 진행하기 때문에 다른 양자화 알고리듬에 비해 속도가 떨어지는 단점이 있다. 두번째 양자화 알고리듬은 정적 양자화(Post Static Quantization)이다. 정적 양자화는 동적 양자화와는 다르게 양자화 알고리듬을 적용함에 있어서 학습에 사용되었던 데이터의 일부(Representative data)를 인공 신경망에 입력함으로써 활성화 함수의 입출력 값을 양자화 하기 위한 s값을 미리 계산해 두는 방식의 알고리듬이다. 정적 양자화는 동적 양자화가 인공 신경망이 구동될 때 실시간으로 양자화 하는 부분이 필요 없기 때문에 동적 양자화에 비해 빠른 추론 속도를 가질 수 있다. 마지막 양자화 알고리듬은 양자화 인식 학습(Quantization Aware Training, QAT)으로, 인공 신경망이 학습할 때 사용했던 데이터 전부를 사용하여 양자화된 인공 신경망을 다시 학습시키는 알고리듬이다. 양자화 인식 학습은 세가지 알고리듬 중 가장 정확도가 보존이 잘된다는 장점이 있고 정적 양자화와 같은 수준의 추론 속도를 보장하지만 경량 인공 신경망의 생성을 위해 추가적인 학습이 필요한 단점이 있다. 대부분의 경우 정적 양자화만 적용하더라도 기존의 정확도를 보존한 경량 인공 신경망을 생성할 수 있다. 본 연구에서는 정적 양자화와 양자화 인식 학습을 적용하여 정확도를 비교하였고 임베디드 시스템상에서 추론 가속화 성능을 확인하였다.

[Table 1] Quantization algorithms: Quantization algorithms can be categorized into dynamic quantization, post static quantization and quantization aware training

	Latency	Accuracy	Data Requirement
Dynamic Quantization	Low	Best	No
Post Static Quantization	Lowest	Good	Partial
Quantization Aware Training	Lowest	Best	All

5. 지식 증류

임베디드 시스템과 같이 제한된 컴퓨팅 리소스를 가진 시스템에 인공 신경망의 탑재 및 원활한 구동을 위해서는 임베디드 시스템에 적합한 경량 인공 신경망의 설계가 필수적이다. 하지만 이러한 신경망은 가중치의 개수가 충분하지 않아 강건한 성능을 기대하기 어렵다. 따라서 해당 인공 신경망의 성능을 최대한 향상하기 위한 학습 방법이 필요하다.

이를 위해 Teacher와 Student의 개념을 이용하여 경량 인공 신경망의 성능을 향상시킨 Knowledge Distillation이 제안되었다[12]. Teacher는 성능이 좋은 인공 신경망으로 Student에 비하여 많은 가중치를 가지고 있으며 반대로 Student는 Teacher보다 상대적으로 크기가 작고 성능이 떨어지는 인공 신경망이다. [Fig. 3]과 같이 Student는 Teacher와 같은 입력 데이터에 대하여 서로 다른 값을 예측하며 Student는 식 (4)~(7)을 이용하여 Teacher의 예측 값을 참조하여 학습한다.

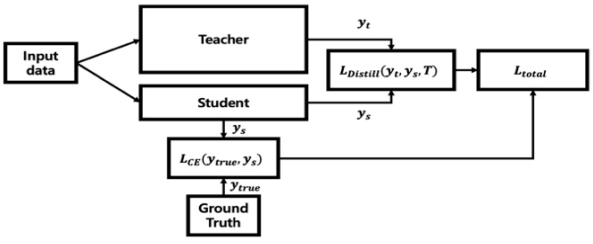
$$L_{CE}(y_{true},y_s) = Crossentropy(y_{true},y_s) \tag{4}$$

$$L_{Distill}(y_t,y_s,T) = L_{KLD}(p(y_t,T),p(y_s,T)) \tag{5}$$

$$p(y_i,T) = \sum_j \frac{\exp\left(\frac{y_i}{T}\right)}{\exp\left(\frac{y_j}{T}\right)} \tag{6}$$

$$L_{total} = \alpha L_{CE} + (1-\alpha) L_{Distill} \tag{7}$$
$$\text{where } \alpha \in [0,1], T > 0.$$

여기서 α와 T는 하이퍼 파라미터로서 α는 Student가 학습 시 Teacher의 예측 값을 얼마나 참조할 지에 대한 지표다. T는 식 (5)와 (6)을 이용하여 Teacher와 Student의 예측 값의 분포의 형태를 조절한다. y_s와 y_t는 각각 Student와 Teacher 신경망의 출력 값이며 y_{true}는 입력 데이터에 대한 ground truth data다. L_{KLD}는 Teacher와 Student의 예측 값의 분포를 Kullback-Leibler



[Fig. 3] Knowledge distillation: the student network refers to the output of the teacher network for better training result

divergence를 이용하여 계산한 손실 함수다.

기존의 지식 증류 알고리듬이 Teacher의 마지막 층의 예측 값만을 이용하는 것에 반해 Teacher의 은닉 층의 예측 값 또한 Student가 참조할 수 있는 지식 증류 알고리듬이 존재한다. 이는 Teacher의 은닉 층을 참조할 수 있기 때문에 기존의 Knowledge Distillation보다 일반적으로 더 좋은 성능을 보이며 분류 문제뿐만 아니라 회귀 문제에서도 사용 가능하다는 이점이 있다. B. Heo et al.은 아래의 식 (8)~(9)를 이용하여 Teacher 인공 신경망의 은닉 층에 존재하는 뉴런의 Activation boundary를 Student가 모방하는 지식 증류 기법을 제안하였고 이를 이용하여 Student 모델의 정확도를 크게 개선하였다[14].

$$L(I) = \|\rho(T(I)) \odot \sigma(\mu_1 - r(S(I))) + (1-\rho(T(I)) \odot \sigma(\mu_1 + r(S(I)))\|_2^2 \quad (8)$$

$$\text{where } \rho(x) = \begin{cases} 1 & if \quad x > 0 \\ 0 & otherwise. \end{cases} \quad (9)$$

여기서 $T(\cdot)$와 $S(\cdot)$은 각각 Teacher와 Student의 활성화 함수의 출력 값이며 I는 입력 데이터이다. $\sigma(\cdot)$는 ReLU 활성화 함수이며 μ_1은 unit margin vector이다. $r(\cdot)$은 A. Romero et al.[21]이 제안한 connector function으로 Teacher와 Student의 출력 값의 차원을 일치시키는 역할을 한다. $\odot$은 요소별 곱셈(element-wise product)이며 $\rho(\cdot)$는 신경망의 뉴런의 활성화 여부를 계산하는 함수다. 해당 지식 증류 알고리듬은 지식 증류의 과정을 Initialization과 Training 두 가지로 나누어 진행한다. Initialization에서는 Student가 Teacher와 유사한 Activation boundary를 형성하는 과정으로 이 과정에서는 y_{true}를 참조하지 않는다. Initialization 과정이 끝난 후 Student 모델은 전체 학습 데이터를 이용하여 Teacher의 도움 없이 스스로 학습을 진행한다. 본 연구에서는 B. Heo et al. [14]의 알고리듬을 이용하여 큰 모델인 Teacher (DenseNet201)를 이용하여 작은 신경망 모델인 Student (DenseNet121)의 성능을 향상시키는 연구를 수행하였다.

6. 신경망 경량화 결과

6.1 인공 신경망 학습 파라미터 및 타겟 인공 신경망 설정

본 연구에서는 Tensorflow를 사용하여 DenseNet201을 CIFAR-10 데이터 셋을 이용하여 학습시켜 이미지 분류(Image Classification) 문제에 대해 적용하였다. 학습이 완료된 DenseNet201에 대해 앞서 언급한 세가지의 경량화 알고리듬을 적용하였다. DenseNet201의 학습에 대한 설정은 [Table 2]와 같다.

[Table 2] DenseNet201 training parameters and result: For training, Tensorflow framework, CIFAR10 dataset and Nvidia RTX 2070SUPER GPU are used

Epochs (Steps)	50 (312,500)	
Batch size	8	
Optimizer	Stochastic Gradient Descent (Nesterov, Momentum=0.9)	
Learning Rate	10^{-3}	$1 \leq \text{epoch} \leq 25$
	10^{-4}	$26 \leq \text{epoch} \leq 38$
	10^{-5}	$39 \leq \text{epoch} \leq 50$
Top-1 Accucracy	95.15%	
Model Size	70 MB	

DenseNet은 ImageNet 데이터 셋에 최적화 되어있기 때문에 CIFAR-10을 ImageNet 데이터 셋의 크기와 같은 224x224의 크기를 갖도록 재조정 하였으며 각 채널의 값이 0과 1사이의 실수 값을 가지도록 정규화(Normalization) 하였다. 또한, 데이터 증강(Data augmentation)을 복합적으로 적용하였다[22].

여기서 Top-N Accuracy는 이미지 분류 문제에서 사용되는 평가 지표로서, 인공 신경망의 마지막 층에서 나온 이산화 된 확률 분포 값에서 상위 N개의 값 중 정답 데이터와 일치하는 경우가 있을 경우 정답으로 처리하는 지표다.

학습이 완료된 DenseNet201에 경량화 알고리듬을 적용하였으며 임베디드 시스템에서의 성능 파악 및 비교 분석을 위해 경량 인공 신경망을 NVIDIA Jetson Xavier에 탑재하여 성능 비교 및 분석하였다. 또한 임베디드 시스템에 탑재 시 Tensorflow Lite의 경량 모델 탑재 최적화 기능을 사용하여 경량화 및 가속화 성능을 확인하였다.

6.2 가지치기 알고리듬 적용 및 결과

학습이 완료된 DenseNet201을 Gradual Pruning을 이용하여 가지치기를 50%부터 99%까지 다양한 비율로 적용하였다. $s_i = t_i = 0$으로 설정하였으며 n은 총 steps의 80% (250,000)로 설정하여 학습 후반부에는 미세조정(Fine-Tuning)할 수 있도록 하였다. 또한 M. Zhu et al. [13]이 제시한 대로 기존 학습에서 사용했던 Learning Rate의 1/10을 곱한 값(10^{-4}, 10^{-5}, 10^{-6})을 사용하였다.

[Table 3] 에서 확인할 수 있듯이 DenseNet201의 가중치를 90%까지 가지치기 적용했을 때 정확도 하락 없이 약 6배 경량화가 가능한 것을 확인할 수 있다. 특히 가지치기 적용 시 작은 Learning Rate와 학습 후반부에는 신경망이 학습 데이터에 대한 미세조정(Fine-Tuning) 할 수 있도록 하여 가지치기를 적용해도 인공 신경망의 성능이 소폭 향상되는 것을 확인했다. 하

[Table 3] DenseNet201 gradual pruning result: Gradual pruning compressed the neural network. we specified an increase or decrease in accuracy and compression ratio

Sparsity [%]	Top-1 Accuracy [%]	Size [MB]
0	95.15 (+0%)	70 (100%)
50	97.44 (+2.29%)	51 (73%)
60	97.25 (+2.10%)	41 (59%)
70	97.41 (+2.26%)	32 (46%)
80	96.91 (+1.76%)	22 (31%)
90	95.43 (+0.28%)	12 (17%)
95	92.23 (-2.92%)	6.2 (8.9%)
99	71.56 (-23.6%)	2.2 (3.1%)

[Table 4] Static quantization and QAT result

	Size [MB]	Top-1 Accuracy [%]	latency[ms]
FP32	70 (100%)	95.15 (+0%)	1274.03
INT8	19 (27%)	93.23 (-1.92%)	763.93
INT8*	19 (27%)	97.09 (+1.94%)	768.83

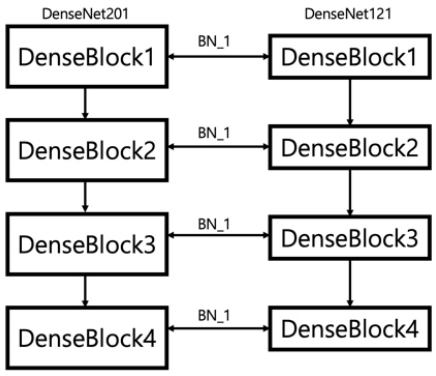
지만 가지치기의 비율을 99%까지 늘렸을 때 인공 신경망의 성능이 급격하게 23.6% 감소하는 것을 볼 수 있었다.

6.3 양자화 알고리듬 적용 및 결과

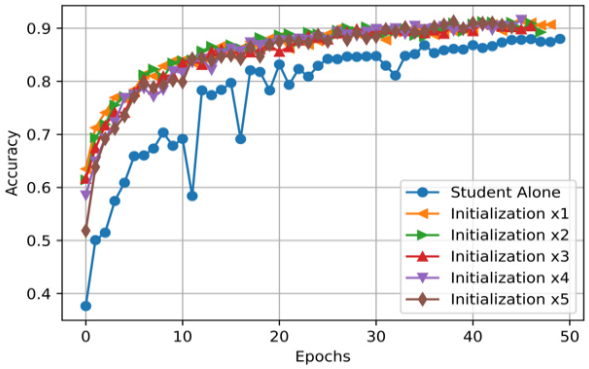
DenseNet201의 가중치를 정적 양자화(INT8)와 양자화 인식 학습(INT8*)을 이용하여 INT8로 양자화 하였다. 양자화 인식 학습의 경우에 DenseNet201의 학습 파라미터를 동일하게 적용했다. 가중치 데이터 정밀도로 INT8을 이용 시 기존 FP32에 비해 인공 신경망의 크기가 약 73% 감소하는 것을 확인할 수 있다. 또한 정적 양자화는 기존 인공 신경망의 정확도에 비해 1.92% 감소했지만, 양자화 인식 학습을 적용하면 가중치를 양자화 한 후 기존 학습에서 사용됐던 Learning Rate보다 더 작은 값(10^{-4}, 10^{-5}, 10^{-6})으로 학습을 추가적으로 진행하여 정확도가 1.94% 증가한 것을 확인할 수 있다. 마지막으로 부동 소수점에서 고정 소수점으로 변경한 연산의 이점으로 NVIDIA Jetson Xavier의 CPU에서 1274.03 ms 걸리던 추론 시간을 768.83 ms로 약 40% 감소 시킨 것을 확인할 수 있으며 그 결과를 [Table 4]에 정리했다. 추론 시간은 Batch size를 1로 설정한 후 총 50번의 추론에 소요된 시간을 측정한 값의 평균 값을 이용했다.

6.4 지식 증류 알고리듬 적용 및 결과

지식 증류 기법[14] 적용을 위해 DenseNet201을 Teacher로



[Fig. 4] Hidden layers for knowledge distillation: the student network refers to the feature from the batch normalization layers in the dense blocks in the teacher network



[Fig. 5] Knowledge distillation result: Knowledge distillation algorithm improved the accuracy of DenseNet121 using DenseNet201

[Table 5] Knowledge distillation result in terms of size, accuracy and latency: the performance of DenseNet121 has been improved using DenseNet201 by 3.5%

	Size [MB]	Top-1 Accuracy [%]	latency[ms]
Teacher	70 (100%)	95.15	1274.03
Student	29 (41%)	88.0 → 91.5 (+3.5%)	658.06

DenseNet121을 Student로 선정하여 [Fig. 4]와 같이 각각 신경망의 구성요소인 DenseBlock의 첫번째 Batch Normalization Layer의 Activation boundary를 학습하도록 설정하였다.

Initialization은 1회에서 5회까지 적용하여 정확도를 비교하였으며 Epoch은 Initialization을 포함하여 50으로 지정하였다. 실험 결과 지식 증류 알고리듬을 1회 이상 적용하면 비슷한 정확도 상승 효과를 확인할 수 있었으며 학습 결과는 [Fig 5]와 같다.

또한, [Table 5]에서 확인할 수 있듯이 지식 증류를 적용하면 적용 하지 않은 경우보다 성능이 최대 3.5% (Initialization x4의 경우) 증가하는 것을 확인할 수 있으며 NVIDIA Jetson

Xavier의 CPU에서 658.06 ms의 추론 시간을 보여 기존 DenseNet201 (1274.03 ms) 대비 48.3% 추론 시간이 감소하였음을 확인하였다.

6.5 경량화 알고리듬 융합 적용 결과

앞서 언급한 인공 신경망 경량화 알고리듬은 각자 독립적인 방식으로 인공 신경망를 경량화 하며, 이들을 적절히 융합하면 더 나은 성능을 기대할 수 있다. 앞서 언급한 세가지 경량화 알고리듬을 융합하기 위해서는 먼저 지식 증류를 적용한 경량 인공 신경망을 구현한 후, 추가적으로 가지치기를 적용하여 경량 인공 신경망을 희소한(Sparse) 가중치를 가진 모델로 변환하여 인공 신경망을 더 압축해야 한다. 이렇게 압축된 모델에 마지막으로 양자화 인식 학습을 적용하여 부동 소수점 32비트의 가중치로 구성된 모델을 8비트 가중치를 가진 모델로 변환함으로써 세가지 알고리듬이 모두 적용된 경량 인공 신경망을 구현할 수 있다. 만약 양자화 알고리듬을 먼저 적용 후 가지치기를 하게 되면 정밀한 실수 값으로 표현되었던 가중치들이 양자화에 의해 정수로 표현되고, 이러한 정수 가중치들에 대해 절댓값이 낮은 순서대로 가지치기를 적용하게 된다면 신경망 모델의 심각한 정확도 하락을 가져온다. 이러한 이유로, 본 연구에서 학습을 위해 사용한 Tensorflow는 양자화 알고리듬이 적용된 인공 신경망에 대하여 추가적으로 가지치기 알고리듬을 적용하는 것을 지원하지 않는다. 따라서, 본 연구에서는 지식 증류가 적용된 DenseNet121에 대하여 가지치기를 적용한 후 양자화 인식 학습을 적용하는 순서로 융합 경량 인공 신경망을 생성하였다.

[Table 6] DenseNet121(knowledge distilled) pruning result

Sparsity [%]	Top-1 Accuracy [%]	Size [MB]
0	91.50 (+0%)	29 (100%)
50	92.42 (+0.92%)	20 (69%)
60	91.91 (+0.41%)	16 (55%)
70	91.36 (-0.14%)	12 (41%)
80	89.93 (-1.57%)	8.0 (28%)
90	85.88 (-5.62%)	4.3 (15%)
95	78.26 (-13.2%)	2.4 (8.2%)
99	10.00 (-81.5%)	0.9 (3.0%)

[Table 7] The performance of DenseNet121 on the embedded system: the network has been compressed by knowledge distillation, pruning and QAT

Sparsity [%]	Size [MB]	Top-1 Accuracy [%]	latency [ms]
80	4.6	93.35	499.26
90	2.8	88.60	487.65
95	1.9	85.54	497.30

[Table 8] Comparison of performance of the neural networks before and after model compression

Device	Size [MB]	Top-1 Accuracy [%]	latency [ms]
CPU	70	95.15	1274.03
GPU	70	95.15	366.85
CPU*	4.6	93.35	499.26

지식 증류 알고리듬이 적용된 DenseNet121에 대한 가지치기의 경우 앞서 DenseNet201에 적용한 방식과 같은 방식을 이용하여 진행하였다. Learning Rate를 감소 시켜가며 가지치기와 추가적인 학습을 동시에 진행하였기 때문에 특정 가지치기 비율까지는 인공 신경망의 성능이 보존되거나 상승하는 것을 확인할 수 있다. 그러나 가지치기 비율이 80%가 넘어가면서 급격하게 정확도가 하락하는 것을 확인하였으며 그 결과는 [Table 6]과 같다.

양자화 인식 학습에서는 가지치기 비율이 80%, 90%, 95% 적용된 경량 인공 신경망에 대하여 진행하였으며 그 결과는 [Table 7]과 같다.

마지막으로, 지식 증류가 적용된 DenseNet121에 대해 80% 가지치기와 양자화 인식 학습을 적용한 인공신경망을 최종 경량 인공 신경망으로 선정하여 NVIDIA Jetson Xavier의 CPU상에서 추론 시간을 측정하였으며 그 결과는 [Table 8]과 같다. 세가지 경량화 알고리듬이 모두 적용된 경량 인공 신경망은 정확도의 저하는 최소화 되면서도 그 크기는 4.6 MB로 기존 대비 15.2배 경량화 되었으며 추론 시간은 경량화 전(CPU) 보다 60.8%(CPU*) 개선 되었다. 참고로, GPU에서의 경량 인공 신경망 탑재 및 성능 평가는 Tensorflow Lite 및 NVIDIA TensorRT가 경량화 알고리듬 탑재를 완벽하게 지원하지 않아 수행하지 않았다. 비교를 위해 경량화 이전 인공 신경망을 NVIDIA Jetson Xavier의 GPU에 사용 시 latency를 표에 추가하였다.

여기서 첫번째 행(CPU)과 두번째 행(GPU)에 대한 결과는 경량화 알고리듬이 적용되지 않은 DenseNet201을 임베디드 시스템에서 구동시킨 결과이며 세번째 행(CPU*)은 DenseNet121 (Student)을 DenseNet201 (Teacher)를 이용하여 지식 증류 알고리듬을 적용한 후 가지치기, 양자화 인식 학습을 복합적으로 적용한 경량 인공 신경망을 동일한 임베디드 시스템에서 구동한 결과이다.

7. 결 론

본 연구에서는 가지치기, 양자화, 지식 증류 경량화 알고리듬의 이론적인 내용을 정리하고 이를 합성곱 인공 인공신경망

에 적용하여 그 효과를 분석하였다. 경량화 알고리듬을 인공 신경망에 적용함에 있어서 가장 중요한 부분은 기존의 인공 신경망이 가지고 있는 정확도 보존이다. 본 연구에서는 앞서 설명한 알고리듬들을 정확도 손실을 최소화하면서 합성곱 인공 신경망에 적용하였다. 첫번째로 가지치기의 경우 정확도 손실 없이 최대 5.83배 인공 신경망을 압축할 수 있음을 보였으며 추가적으로 정확도 2.92%의 하락만으로 11.3배 경량화 된 인공 신경망 생성이 가능함을 보였다. 두번째로 정적 양자화와 양자화 인식 학습을 적용하여 3.7배 경량화 된 인공 신경망을 생성하였으며 임베디드 시스템에서의 추론 시간을 39.7% 감소시켰다. 세번째로 지식 증류에서는 DenseNet201을 이용하여 DenseNet121의 정확도를 3.5%의 정확도를 향상하였으며, DenseNet201 대비 2.41배 경량화 된 인공 신경망을 이용하여 임베디드 시스템 상에서 46.6%의 추론 시간을 감소시킬 수 있음을 확인하였다. 마지막으로 세가지 경량화 알고리듬을 복합적으로 적용하여 인공 신경망의 성능을 최대한 보존하면서도 모델 크기는 15.2배, 추론 시간은 60.8% 감소된 경량 인공 신경망을 생성하였다. 향후, 앞서 언급한 경량화 알고리듬을 객체 탐지 알고리듬에 적용 후 FPGA (Field Programmable Gate Array)에 탑재하여 저전력, 고성능의 객체 탐지 임무를 수행할 계획이다.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84-90, May, 2017, DOI: 10.1145/3065386.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, "ImageNet: A Large-scale Hierarchical Image Database," *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, USA, 2009, DOI: 10.1109/CVPR.2009.5206848.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, DOI: 10.1109/CVPR.2016.90.

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale," *International Conference on Learning Representations*, Vienna, Austria, 2021. [Online], https://openreview.net/forum?id=YicbFdNTTy.

[5] S. Oh, H. Kim, S. Cho, J. You, Y. Kwon, W.-S. Ra, and Y.-K. Kim, "Development of a Compressed Deep Neural Network for Detecting Defected Electrical Substation Insulators Using a Drone," *Journal of Institute of Control, Robotics and Systems*, vol. 26, no. 11, pp. 884-890, Nov., 2020, DOI: 10.5302/j.icros.2020.20.0117.

[6] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv preprint arXiv:1704.04861*, 2017, [Online], https://arxiv.org/abs/1704.04861.

[7] T. Uetsuki, Y. Okuyama, and J. Shin, "CNN-based End-to-end Autonomous Driving on FPGA Using TVM and VTA," *2021 IEEE 14th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, Singapore, 2021, DOI: 10.1109/MCSoC51149.2021.00028..

[8] S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding," *arXiv preprint arXiv:1510. 00149*, 2015, [Online], https://arxiv.org/abs/1510.00149.

[9] Y. LeCun, D. John, and S. Sara, "Optimal Brain Damage," *Advances in Neural Information Processing Systems*, 1989, [Online], https://papers.nips.cc/paper/1989/hash/6c9882bbac1c7093bd25041881277658-Abstract.html.

[10] B. Hassibi and D. Stork, "Second Order Derivatives for Network Pruning: Optimal Brain Surgeon," *Advances in Neural Information Processing Systems*, 1992, [Online], https://proceedings.neurips.cc/paper/1992/hash/303ed4c69846ab36c2904d3ba8573050-Abstract.html.

[11] H. Wu, P. Judd, X. Zhang, M. Isaev, and P. Micikevicius, "Integer Quantization for Deep Learning Inference: Principles and Empirical Evaluation," *arXiv preprint arXiv:2004.09602*. 2020, [Online], https://arxiv.org/abs/2004.09602.

[12] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," *Advances in Neural Information Processing Systems*, 2014, [Online], https://arxiv.org/abs/1503.02531.

[13] M. Zhu and S. Gupta, "To prune, or not to Prune: Exploring the Efficacy of Pruning for Model Compression," *International Conference on Learning Representations Workshop*, 2018, [Online], https://arxiv.org/abs/1710.01878.

[14] B. Heo, M. Lee, S. Yun, and J. Choi, "Knowledge Transfer via Distillation of Activation Boundaries Formed by Hidden Neurons," *AAAI Conference on Artificial Intelligence*, 2019, DOI: 10.1609/aaai.v33i01.33013779.

[15] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based Learning Applied to Document Recognition," *IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov., 1998, DOI: 10.1109/5.726791.

[16] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large Scale Image Recognition," *International Conference on Learning Representations*, 2015, [Online], https://arxiv.org/abs/1409.1556.

[17] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 2017, DOI: 10.1109/CVPR.2017.243.

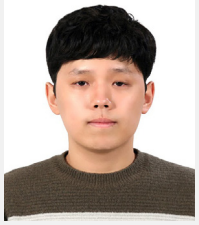
[18] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning Both Weights and Connections for Efficient Neural Networks," *arXiv:1506.02626*, 2015, [Online], https://arxiv.org/abs/1506.02626.

[19] A. See, M.-T. Luong, and C. D. Manning, “Compression of Neural Machine Translation via Pruning,” *The 20th Signll Conference on Computational Natural Language Learning*, Aug., 2016, DOI: 10.18653/v1/K16-1029.

[20] S. Narang, E. Elsen, G. Diamos, and S. Sengupta, “Exploring Sparsity in Recurrent Neural Networks,” *arXiv preprint arXiv: 1704.05119*, 2017, [Online], https://arxiv.org/abs/1704.05119.

[21] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, “Fitnets: Hints for Thin Deep Nets,” *arXiv:1412.6550*, 2015, [Online], https://arxiv.org/abs/1412.6550.

[22] S. Lim, I. Kim, T. Kim, C. Kim, and S. Kim, “Fast Auto-augment,” *Advances in Neural Information Processing Systems*, vol. 32, 2019, [Online], https://papers.nips.cc/paper/2019/hash/6add07cf50424b14fdf649da87843d01-Abstract.html.



신 희 중

2020년 국민대학교 기계공학과(공학사)

2020~현재 울산과학기술원 기계공학과 석사과정

관심분야: 기계학습, 무인기 제어, 딥러닝 경량화

오 현 동

2004년 KAIST 항공우주공학과(공학사)

2010년 KAIST 항공우주공학과(공학석사)

2013년 Cranfield University 항공우주공학과 (공학박사)

2013년~2014년 영국 University of Surrey 박사 후 연구원

2014년~2016년 영국 Loghborough University 조교수

2016년~현재 울산과학기술원 부교수

관심분야: 무인이동체 의사 결정, 협력 제어, 경로 계획, 비선형 유도/제어, 센서/정보 융합