The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 24, No. 1, pp.1-8, Feb. 29, 2024. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2024.24.1.1
JIIBC 2024-1-1

지능형 엣지 컴퓨팅 기기를 위한 온디바이스 AI 비전 모델의 경량화 방식 분석

Analysis on Lightweight Methods of On-Device AI Vision Model for Intelligent Edge Computing Devices

주혜현[*], 강남희[**]

Hye-Hyeon Ju[*], Namhi Kang[**]

요 약 실시간 처리 및 프라이버시 강화를 위해 인공지능 모델을 엣지에서 동작시킬 수 있는 온디바이스 AI 기술이 각광받고 있다. 지능형 사물인터넷 기술이 다양한 산업에 적용되면서 온디바이스 AI 기술을 활용한 서비스가 크게 증가하고 있다. 그러나 일반적인 딥러닝 모델은 추론 및 학습을 위해 많은 연산 자원을 요구하고 있다. 따라서 엣지에 적용되는 경량 기기에서 딥러닝 모델을 동작시키기 위해 양자화나 가지치기와 같은 다양한 경량화 기법들이 적용되어야 한다. 본 논문에서는 다양한 경량화 기법 중 가지치기 기술을 중심으로 엣지 컴퓨팅 기기에서 딥러닝 모델을 경량화하여 적용할 수 있는 방안을 분석한다. 특히, 동적 및 정적 가지치기 기법을 적용하여 경량화된 비전 모델의 추론 속도, 정확도 그리고 메모리 사용량을 시험한다. 논문에서 분석된 내용은 실시간 특성이 중요한 지능형 영상 관제 시스템이나 자율이동체의 영상 보안 시스템에 적용될 수 있다. 또한 사물인터넷 기술이 적용되는 다양한 서비스와 산업에 더욱 효과적으로 활용될 수 있을 것으로 기대된다.

Abstract On-device AI technology, which can operate AI models at the edge devices to support real-time processing and privacy enhancement, is attracting attention. As intelligent IoT is applied to various industries, services utilizing the on-device AI technology are increasing significantly. However, general deep learning models require a lot of computational resources for inference and learning. Therefore, various lightweighting methods such as quantization and pruning have been suggested to operate deep learning models in embedded edge devices. Among the lightweighting methods, we analyze how to lightweight and apply deep learning models to edge computing devices, focusing on pruning technology in this paper. In particular, we utilize dynamic and static pruning techniques to evaluate the inference speed, accuracy, and memory usage of a lightweight AI vision model. The content analyzed in this paper can be used for intelligent video control systems or video security systems in autonomous vehicles, where real-time processing are highly required. In addition, it is expected that the content can be used more effectively in various IoT services and industries.

Key Words : AI, Computer Vision, Deep Learning, On-device AI, Lightweight Device

[*]비회원, 덕성여자대학 사이버보안전공
[**]종신회원, 덕성여자대학 데이터사이언스학과
접수일자 2023년 12월 9일, 수정완료 2024년 1월 9일
게재확정일자 2024년 2월 9일

Received: 9 December, 2023 / Revised: 9 January, 2024 / Accepted: 9 February, 2024
[**]Corresponding Author: kang@duksung.ac.kr
Dept. of Data Science, Duksung Women's University, Korea

Ⅰ. 서 론

온디바이스 AI는 사용자의 전자기기, 엣지 디바이스 자체에서 딥러닝이나 컴퓨팅 작업을 직접 실행하는 것을 의미한다[1]. 온디바이스 AI는 데이터를 클라우드 서버로 전송하지 않고, 디바이스 내부에 유지하여 개인 정보를 보호할 수 있다. 또한, 서버로부터 디바이스로의 계산을 오프 로딩하여 배치되는 디바이스의 수가 증가할수록 운영 비용을 절감할 수 있다. 비전 AI를 사용할 경우, 고대역 실시간 스트림 데이터의 전송 지연 시간을 줄일 수 있다[2].

온디바이스 AI가 적용되는 경량 기기에서 딥러닝 모델을 실행할 때 가장 중요한 고려사항은 제한적인 컴퓨팅 자원의 효율적인 사용이다. 그러나 딥러닝 모델은 실행 중에 많은 중간 연산 값을 생성하여 메모리 사용량의 증가와 계산 병목 현상을 야기할 수 있다. 이러한 딥러닝 모델의 특성은 온디바이스 AI의 적용을 어렵게 하는 원인이 된다. 특히 온디바이스 AI가 적용되는 기기의 컴퓨팅 자원이 제한적이고 에너지 효율이 떨어짐에도 대부분 실시간으로 연산이 수행되어야 하는 경우가 많다. 또한, 점점 더 경량화된 기기에서 다양한 기능을 갖춘 소형기기가 선호되고 있다. 이러한 요구 사항은 메모리, 자원 및 전력은 한정적임에도 강력한 딥러닝 모델을 내장하는 방법을 요구하며, 제한된 자원의 효율적 사용을 더욱더 중요하게 만든다[17].

경량 기기에서 딥러닝 모델을 효율적으로 동작시키기 위해 모델의 크기를 줄여 연산량을 감소시켜 메모리 사용량을 최적화해야 한다. 이를 위해 네트워크 슬리밍 (Network slimming)[3], 정적 가지치기 (Static Pruning)[6], 동적 가지치기 (Dynamic Pruning)[4], 훈련 후 양자화 (Post-Training Quantization)[7], 그룹 컨볼루션 (Group Convolution)[11] 방식 등 다양한 모델 경량화 기법들이 제안되고 있다.

네트워크 슬리밍 기술과 동적 그리고 정적 가지치기 기술은 딥러닝 모델의 크기를 경량화하여 메모리를 효율적으로 사용하게 할 수 있지만, 성능이 저하될 수 있다는 단점도 있다. 또한, 딥러닝 모델과 학습시킬 데이터 세트의 종류가 매우 다양하여 항상 만족스러운 경량화를 달성하기 어렵다. 모델 경량화 과정에서는 모델 신경망의 깊이, 데이터 세트, 적용 기기 등 다양한 상황들을 세밀히 고려하는 것이 중요하다.

본 논문에서는 다양한 경량 기술 중 정적 가지치기 기술과 동적 가지치기 기술을 시험을 통해 분석한다. 분석에 적용되는 딥러닝 모델은 실시간 (Real Time) 객체 탐지 (Object Detect)를 목적으로 제안된 Yolo를 사용한다[9]. 경량화 기술의 영향을 분석하기 위해 사용되는 데이터 세트를 구성하는 클래스 개수와 신경망 깊이를 다르게 적용해서 비교한다. 딥러닝 모델의 경량화 후 성능 비교는 추론 시간 (Inference time), 정확도 (Accuracy), 그리고 메모리 사용량을 측정한다. 경량 방식의 비교를 통해 상황에 따른 가장 효율적인 경량화 방식을 제안하고자 한다.

Ⅱ. 관련 연구

사물인터넷 (IoT: Internet of Thing) 기술이 적용되는 서비스와 산업이 증가하면서 온디바이스 AI 기술이 헬스케어, 스마트팜 등 상업적으로 널리 사용되고 있다[12, 13, 15, 16]. 경량 임베디드 기기 (Embedded Device)에 AI 기술을 적용하기 위해서는 제한적인 자원을 효율적으로 사용해야 한다. 이에 따라 임베디드 기기에서 작동되는 AI 모델 크기를 줄일 수 있는 다양한 경량화 기법들의 연구가 활발히 진행됐다.

본 장에서는 가지치기를 수행하는 동적 방식과 정적 방식 기술이 적용된 선행 연구를 간략히 기술한다.

1. 동적 가지치기

동적 가지치기는 AI 모델의 학습 과정에서 특정한 조건이 만족할 때 가지치기를 적용한다. 제안된 기술들 중 LKAM (Learning Kernel-Activation Module) 기술에서는 모델의 모든 Conv. 계층 사이에 LKAM 모듈을 적용하여 학습 과정 중에 동적으로 원하는 희소성을 달성하며 가지치기를 수행하는 방식으로 모델의 연산량을 감소시킨다[4]. 그러나 이 방식은 커널 전환과 데이터 관리 작업에 대해 GPU에서 추가 연산이 필요한 한계를 가지고 있다.

CNN 모델에서 특징 강화와 억제 (Feature Boosting and Suppression) 기술을 적용하여 동적으로 가지치기를 수행하는 기술도 제안되었다[5]. 이 방식에서는 채널의 중요도에 따라 중요한 특징은 강화하고 불필요한 특징은 억제하는 방법을 사용한다. 해당 기법을 통해 모델의 연산량을 감소시켰음을 증명했다. 그러나 제시된 시험 환경으로는 경량 기기에서 직접 동작하는 온디바이스 AI의 성능을 확인하기는 어렵다.

2. 정적 가지치기

정적 가지치기 방식에서는 학습이 모두 종료되거나 시작 전 가지치기를 적용한다. 즉, 학습 도중에는 가지치기가 진행되지 않아야 한다.

정적 가지치기의 구현법으로 L1 노름 (L1 norm)과 L2 노름 (L2 norm)을 활용한 네트워크 슬리밍 방식이 제안되었다[6]. 이 방식은 특정 임계값 이하의 가중치를 가진 모든 연결을 네트워크에서 제거함으로써 모델의 희소성을 증가시켰다. 그러나 벤치마크는 CPU, GPU, 그리고 모바일 GPU에서만 수행되었기에, 모바일 GPU가 없는 경량 기기에서의 온디바이스 AI 성능을 평가하는 것은 어려움이 있을 수 있다.

헤시안 가중 왜곡 측정법 (Hessian-weighted Distortion Measure)을 사용해 정적 가지치기를 수행하는 옵티멀 브레인 서전 (Optimal Brain Surgeon) 기술도 제안되었다[7]. 이 방식은 헤시안 행렬을 활용하여 매개변수의 중요성을 결정하고, 그 결과를 통해 중요하지 않은 매개변수나 연결을 식별하고 제거한다. 그러나 기술된 실험 및 평가 결과는 DNN 모델에 적용된 것으로 CNN 모델에 대한 적용 실험 결과를 확인하기는 어렵다.

상기 기술한 선행 연구와는 다르게 본 연구는 경량 기기 중 하나인 라즈베리파이 (Raspberry Pi)를 사용하며 CNN 기반 딥러닝 모델을 대상으로 한다. 클래스 개수가 다른 데이터 세트와 신경망 깊이가 다른 모델에 동적 가지치기와 정적 가지치기를 적용한 뒤 라즈베리파이에서 실행되는 실시간 객체 탐지 과정에서 가장 높은 정확도와 추론 시간을 비교한다. 경우의 수는 총 4가지로, 클래스 개수가 많은 데이터 세트를 학습하는 신경망 깊이가 깊은 모델, 클래스 개수가 많은 데이터 세트를 학습하는 신경망 깊이가 얕은 모델, 클래스 개수가 적은 데이터 세트를 학습하는 신경망 깊이가 깊은 모델, 클래스 개수가 적은 데이터 세트를 학습하는 신경망 깊이가 얕은 모델로 구성된다. 이를 통해 메모리, 자원 및 전력이 한정적인 온디바이스 AI가 적용되는 경량 기기에서 데이터 세트와 신경망 깊이에 따라 성능 저하를 최소화하면서 모델을 경량화할 수 있는 가장 효율적인 경량화 방식을 제안한다.

Ⅲ. 응용 시스템 및 제안 기법

본 논문에서 사용된 응용 시스템은 소프트웨어 (Yolov5m, Yolov5n), 시스템 모니터링 도구 (Pidstat), 하드웨어(라즈베리파이, 로지텍 C270 웹캠 (Logitech C270 Webcam)), 그리고 데이터 세트 (COCO128, Yusuf Berk Sarıdoğan의 Traffic detection dataset)를 포함한다.

1. 비전 모델 및 하드웨어

본 논문에서 사용한 Yolo는 파이토치 (Pytorch)로 구현된 강력한 딥러닝 프레임워크이다. Yolov5m은 291개의 층 (Layers)과 21,190,557 개의 파라미터를 갖고 49.2 GFLOPs의 연산 속도를 보이며, Yolov5 시리즈 중 세 번째로 큰 모델 크기와 신경망 깊이를 가진다. Yolov5n은 214 개의 층과 1,872,157 개의 파라미터를 갖고 4.6 GFLOPs의 연산 속도를 보이며, 가장 작은 모델 크기와 얕은 신경망 깊이를 가진다. Yolov5는 빠르고 지속적인 업데이트와 함께 오픈 소스로 제공되고 있다. 이러한 요인들을 고려하여 Yolov5로 연구를 진행한다.

Pidstat은 sysstat 패키지에 포함된 시스템 모니터링 도구이다. 본 논문에서는 이 도구를 이용해 메모리 사용량을 측정한다.

본 논문에서는 온디바이스 AI의 정확도, 추론 시간 그리고 메모리 사용량의 비교를 위해 경량 오픈 하드웨어 플랫폼인 라즈베리파이 4를 사용한다. 스크립트의 실행은 라즈베리파이의 리눅스 터미널을 통해 이루어진다.

2. 시험에 적용되는 데이터 세트

클래스 개수에 따른 경량화 기술의 영향을 더욱 명확하게 분석하기 위해, 클래스 개수의 차이가 큰 두 개의 데이터 세트를 사용한다. 첫 번째로, 80 개의 클래스를 포함하는 COCO128 데이터 세트를 사용한다. 이 데이터 세트는 Microsoft COCO (Common objects in context)의 일부로서 다양한 객체를 포함하고 있다. 두 번째로, Yusuf Berk Sarıdoğan가 제작한 Traffic detection 데이터 세트를 사용한다. 이 데이터 세트는 데이터 사이언스 커뮤니티인 캐글 (Kaggle)에서 내려받아 사용하였으며, 총 5 개의 클래스를 가진다. 본 논문의 모든 학습, 평가에는 위 두 가지의 데이터 세트를 사용한다.

3. 적용 경량화 방식

동적 가지치기는 학습 과정 중에 가중치를 지속적으로 분석하여 제거하는 경량 기술이다. 동적 가지치기는 복잡한 연산이 필요하지만 모델의 성능을 유지하면서도 더 많은 가중치를 제거할 수 있다는 이점이 있다.

본 논문에서는 2차원 합성 곱 층 (Conv2d) 및 2차원 배치 정규화 층(Batchnorm2d)에 학습 과정 중 각 에폭(epoch)에서 L1 노름을 기반으로 가지치기를 수행한다. 이를 통해 2차원 합성 곱 층과 2차원 배치 정규화 층의 가중치 텐서에서 가장 작은 L1 노름 값을 가진 가중치를 제거한다. 2차원 합성 곱 층과 2 차원 배치 정규화 층의 희소성 연산과 가지치기 과정은 그림 1, 2, 3, 4에 표현된 알고리즘을 따르며 해당 알고리즘은 Yolov5의 torch_utils.py 내부의 함수를 기반으로 연구 요구사항에 맞게 적절히 수정하여 구현하였다. 모델 형식은 경량 기기에서 널리 활용되는 오픈비노 (OpenVINO) 형식을 채택하였다.

Algorithm 1 Sparsity calculation in Conv2D layer

Initialize total parameters (*total_params*) to 0
Initialize zero parameters (*zero_params*) to 0
for each module and its name (*m*, *name*) in *model.named_modules()* **do**
  if *m* is a Conv2D layer **then**
    Add the number of weights (*number_weight*) to the total parameters (*total_params*)
    Add the number of zero weights (*zero_weight*) to the zero parameters (*zero_params*)
  end if
end for
Calculate sparsity:
$Sparsity = \frac{zero\_params}{total\_params}$
Return sparsity (*Sparsity*)

그림 1. 2차원 합성 곱 층의 희소성 연산 의사 코드
Fig. 1. Pseudo code for calculating sparsity of conv2d layer

Algorithm 1 Sparsity calculation in BatchNorm2d layer

Initialize total parameters (*total_params*) to 0
Initialize zero parameters (*zero_params*) to 0
for each module and its name (*m*, *name*) in *model.named_modules()* **do**
  if *m* is a BatchNorm2d layer **then**
    Add the number of weights (*number_weight*) to the total parameters (*total_params*)
    Add the number of zero weights (*zero_weight*) to the zero parameters (*zero_params*)
  end if
end for
Calculate sparsity:
$Sparsity = \frac{zero\_params}{total\_params}$
Return sparsity (*Sparsity*)

그림 2. 2차원 배치 정규화 층의 희소성 연산 의사 코드
Fig. 2. Pseudo code for calculating sparsity of batchnorm2d layer

Algorithm 1 Static Pruning Algorithm

for each module and its name (*m*, *name*) in *model.named_modules*() **do**
  if *m* is a Conv2d layer **then**
    Apply L1 unstructured pruning on the weights of the Conv2d layer *m* with specified amount
    Remove the pruned weights from the layer *m*
    Log the message: "Static pruning: Model pruned to *Conv2d_sparsity* global sparsity"
  end if
end for

그림 3. 2차원 합성 곱 층의 가지치기 의사 코드
Fig. 3. Pruning pseudo code for conv2d layers

Algorithm 1 Static Pruning Algorithm

for each module and its name (*m*, *name*) in *model.named_modules*() **do**
  if *m* is a Batchnorm2d layer **then**
    Apply L1 unstructured pruning on the weights of the Batchnorm2d layer *m* with specified amount
    Remove the pruned weights from the layer *m*
    Log the message: "Static pruning: Model pruned to *Batchnorm2d_sparsity* global sparsity"
  end if
end for

그림 4. 2차원 배치 정규화 층의 가지치기 의사 코드
Fig. 4. Pruning pseudo code for batchnorm2d layers

정적 가지치기는 모델 학습이 완료되거나 시작 전 가중치의 분석을 수행하며, 이를 통해 절댓값이 작은 가중치를 중요도가 낮다고 판단하여 제거하는 경량 기술이다. 정적 가지치기 기법 역시 2차원 합성 곱 층과 2차원 배치 정규화 층에 대해 가지치기를 수행한다. 모델의 초기 구조를 가지치기 된 상태로 설정하기 위해 학습이 시작되기 전 2차원 합성 곱 층과 2차원 배치 정규화 층의 가중치 텐서에서 L1 노름 값이 가장 작은 가중치를 제거한다. 모델 형식은 경량 기기에서 널리 활용되는 오픈비노 형식을 채택하였다. 2차원 합성 곱 층과 2차원 배치 정규화 층에서 수행된 희소성 연산 및 가지치기에 대한 알고리즘은 동적 가지치기에서 사용된 것과 동일하다.

Ⅳ. 시험 및 분석

1. 시험 구성 환경

데이터 세트의 클래스 개수와 신경망의 깊이를 변화시킨 상황에서 가지치기 기술 적용 후의 성능을 비교하기 위해 이에 적합한 모델 구조와 데이터 세트를 선정하였다. 모델 형식은 경량 기기에서 널리 활용되는 오픈비노 형식을 채택하였다. 데이터 세트는 80 개의 클래스를 포함하는 COCO128 데이터 세트와 Yusuf Berk Sarıdoğan

이 제작한 5 개의 클래스를 포함하는 Traffic detection 데이터 세트를 활용하였다.

실험 장비로는 라즈베리파이 4와 로지텍 C270 웹캠을 사용했다. 메모리 사용량에 대한 분석은 Yolov5의 detect.py 스크립트를 리눅스 커널에서 웹캠을 통해 실행되는 동안 발생하는 메모리 변화를 Pidstat을 활용하여 1초 간격으로 로그에 기록하여 진행한다. 로그에 기록된 값들 중 최댓값을 메모리 사용량으로 판단하였다. 모델의 정확도와 추론 시간은 COCO128로 학습시킨 경우 COCO128로, Yusuf Berk Sarıdoğan의 Traffic detection 데이터 세트로 학습시킨 경우 Yusuf Berk Sarıdoğan의 Traffic detection 데이터 세트를 활용하여 측정한다. 정확도와 추론 시간의 측정은 Yolov5에서 제공하는 detect.py 스크립트를 활용해 수행한다.

2. 시험 결과

본 논문은 2차원 합성 곱 층과 2차원 배치 정규화 층에 가지치기를 적용하였다. 동적 가지치기는 학습 중 에폭마다 가지치기를 수행했고, 정적 가지치기는 학습 시작 전 모델 초기 구조에 가지치기를 적용하였다.

신경망의 깊이가 깊은 모델인 Yolov5m은 2차원 합성 곱 층에서 40%, 2차원 배치 정규화 층에서는 30%의 희소성을 달성시켰다. 그 결과 COCO128 데이터 세트로 학습한 Yolov5m 모델은 총 212 개의 층과 21,172,173 개의 파라미터를 가지며, 48.9 GFLOPs의 연산 속도를 보였다. Yusuf Berk Sarıdoğan의 Traffic detection 데이터 세트를 학습한 Yolov5m 모델의 경우, 212 개의 층과 20,869,098 개의 파라미터를 가지며, 47.9 GFLOPs의 연산 속도를 보였다.

신경망 깊이가 상대적으로 얕은 모델인 Yolov5n은 2차원 합성곱 층에서 20%의 희소성을, 2차원 배치 정규화 층에서는 9.91%의 희소성을 달성시켰다. 이에 따른 결과로 COCO128 데이터 세트로 학습한 Yolov5n 모델은 157 개의 층과 1,867,405 개의 파라미터를 가지며, 4.5 GFLOPs의 연산 속도를 보였다. Yusuf Berk Sarıdoğan의 Traffic detection 데이터 세트를 학습한 Yolov5n 모델의 경우, 157 개의 층과 1,765,930 개의 파라미터를 가지며 4.1 GFLOPs의 연산 속도를 보였다.

가. 정확도

모델 간의 정확도를 비교하기 위해 mAP50 지표를 활용하였으며, 이때 정확도를 평가하는 표의 작성 기준은 각각의 데이터 세트로 설정하였다.

표 1. COCO128 데이터 세트에 대한 가지치기 적용 결과
Table 1. Pruning results for COCO128 dataset

경량화 방식 / 모델	동적 가지치기	정적 가지치기
신경망 깊이가 깊은 모델 (YoloV5m)	0.901	0.968
신경망 깊이가 얕은 모델 (YoloV5n)	0.75	0.91

클래스 개수가 많은 데이터 세트에서는 신경망의 깊이에 따라 가지치기 수행 방식의 효과에 큰 차이가 있음을 확인할 수 있었다.

깊이가 깊은 신경망 모델에서는 가지치기 수행 방식에 따른 정확도 변화가 거의 미미하였다. 반면에 깊이가 얕은 신경망 모델에서는 가지치기 수행 방식에 따른 정확도 변화가 상당히 크게 나타났다. 동적 가지치기와 정적 가지치기의 적용에 따른 정확도 차이가 신경망 깊이가 깊은 모델은 7.43%에 불과했으나, 신경망 깊이가 얕은 모델은 차이가 21.33%까지 증가하였다. 반면에 클래스 개수가 적은 데이터 세트로 학습된 모델의 경우 신경망의 깊이나 가지치기 수행 방식에 따른 정확도의 변동이 크지 않았다. 이러한 결과는 클래스 개수가 많은 데이터 세트를 학습하는 상황에서 신경망 깊이가 얕은 모델에 동적 가지치기를 적용하는 것은 적절하지 않음을 나타낸다.

표 2. Traffic detection 데이터 세트에 대한 가지치기 적용 결과
Table 2. Pruning results for traffic detection

경량화 방식 / 모델	동적 가지치기	정적 가지치기
신경망 깊이가 깊은 모델 (YoloV5m)	0.924	0.928
신경망 깊이가 얕은 모델 (YoloV5n)	0.906	0.907

나. 추론 시간

웹캠을 이용한 detect.py의 실시간 실행을 통해 추론 시간을 측정하였다. 추론 시간은 신경망의 깊이에 따라 비교하였으며, 밀리 초 (ms) 단위로 표현하였다.

신경망의 깊이에 따른 모델 추론 시간을 비교한 결과, 클래스 개수가 적은 데이터 세트에 대해 신경망의 깊이가 얕은 모델이 동적 가지치기와 정적 가지치기를 수행했을 때의 추론 시간은 각각 848.4 ms와 847.6 ms로

큰 차이를 보이지 않았다. 그러나 클래스 개수가 많은 데이터 세트에 대해 정적 가지치기를 적용했을 때의 추론 시간은 1012.3 ms로 측정되었고 동적 가지치기를 적용하였을 때의 추론 시간인 962.4 ms에 비해 49.4 ms가 증가했음을 확인할 수 있었다.

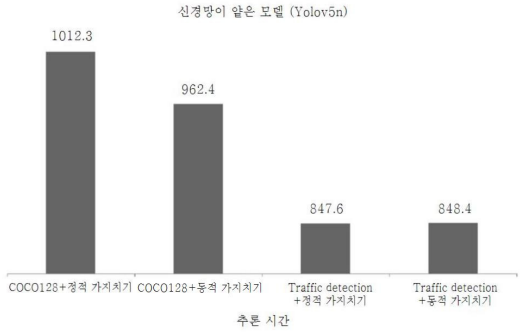

그림 5. 깊이가 얕은 모델 (Yolov5n)에 대한 가지치기 적용 결과
FIg. 5. Pruning results for shallow neural network (Yolov5n)

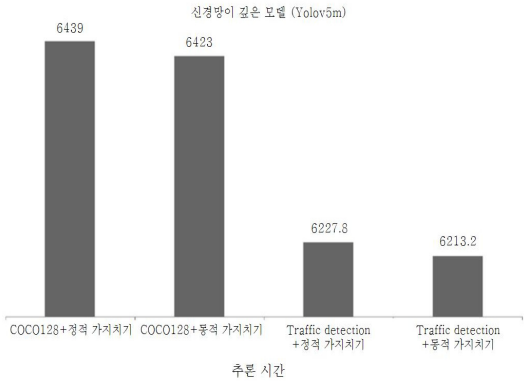

그림 6. 깊이가 깊은 모델 (Yolov5m)에 대한 가지치기 적용 결과
Fig. 6. Pruning results for deep neural network (Yolov5m)

신경망의 깊이가 깊은 모델이 클래스 개수가 적은 데이터 세트에 대해 동적 가지치기와 정적 가지치기를 수행했을 때의 추론 시간은 각각 6213.2 ms, 6227.8 ms로 큰 차이를 보이지 않았다. 클래스 개수가 많은 데이터 세트에 대해서도 동적 가지치기 수행 시 6423 ms, 정적 가지치기 수행 시 6439 ms로 큰 차이가 없었다. 이는 클래스 개수가 많은 경우 모델의 신경망 깊이와 관계없이 어느 가지치기 방식을 적용해도 크게 차이를 나타내지 않으나, 신경망의 깊이가 얕은 모델의 경우, 클래스 개수가 많은 데이터 세트를 학습하는 상황에서 경량화를 할 때는 정적 가지치기 방식의 적용이 비효율적임을 보여준다.

다. 메모리 사용량

웹캠을 이용해 실시간으로 detect.py를 라즈베리파이 4에서 실행시키며 객체 추론이 이루어지는 시작 시점부터 종료 시점까지의 메모리 사용량을 측정하였다. 이는 해당 스크립트 (detect.py)가 실행되는 동안 전체 시스템 메모리의 사용률 (%)을 나타낸다. 메모리 사용량은 신경망의 깊이에 따라 비교하였다.

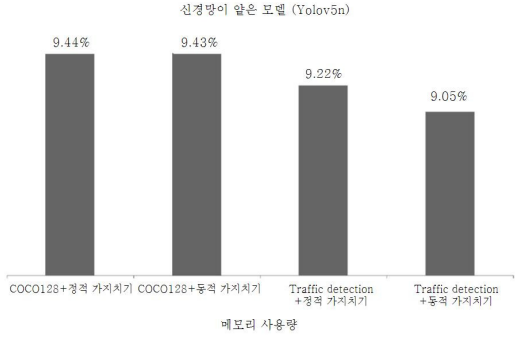

그림 7. 깊이가 얕은 모델 (Yolov5n)에 대한 가지치기 적용 결과
Fig. 7. Pruning results for shallow neural network (Yolov5n)

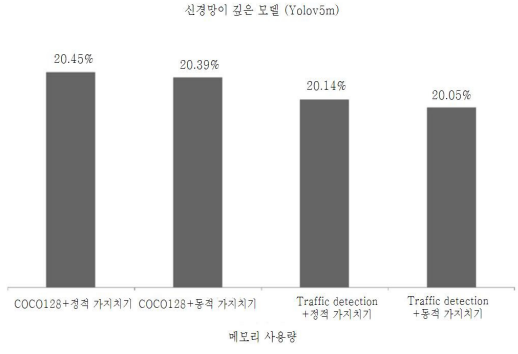

그림 8. 깊이가 깊은 모델(Yolov5m)에 대한 가지치기 적용 결과
Fig. 8. Pruning results for deep neural network (Yolov5m)

가지치기 방식에 따라 전체 시스템 메모리 중 스크립트가 웹캠을 통해 실시간으로 실행되며 사용하는 메모리 양을 비교한 결과, 데이터 세트의 클래스 개수나 신경망의 깊이와 관계없이 모든 경우에서 동적 가지치기를 수행했을 때 정적 가지치기보다 메모리 효율이 더 높음을 확인할 수 있었다.

V. 결 론

본 논문에서는 데이터 세트를 구성하는 클래스 개수와 신경망의 깊이를 변수로 삼아 가지치기를 수행하는 동적 방식과 정적 방식 기술의 성능을 비교하였다.

시험 결과, 클래스 개수가 많고 신경망의 깊이가 깊은 모델에서는 가지치기 수행 방식에 따른 정확도 변화가 크지 않았다. 이는 모델의 신경망 깊이가 깊고 클래스 개수가 적은 데이터 세트를 학습하는 상황과 모델의 신경망 깊이가 얕고 클래스 개수가 적은 데이터 세트를 학습하는 상황에서도 동일하게 나타났다. 그러나 모델의 신경망 깊이가 얕고 클래스 개수가 많은 데이터 세트를 학습하는 경우에는 동적 가지치기를 적용하면 정확도가 매우 저하되는 것을 확인하였다.

추론 시간에 초점을 맞추어 볼 때, 신경망의 깊이가 깊은 모델에서는 클래스 개수와 관계없이 어느 가지치기 방식을 적용하더라도 그 차이가 두드러지지 않았다. 반면에, 신경망의 깊이가 얕은 모델에서 클래스 개수가 많은 데이터 세트로 학습을 진행하는 경우, 동적 가지치기의 적용이 추론 시간을 효과적으로 줄이는 것을 확인할 수 있었다.

메모리 사용량 측면에서 볼 때, 동적 가지치기 방식이 스크립트 실행 시에 메모리를 더 효율적으로 사용하였다. 따라서 클래스 개수가 많은 데이터 세트를 학습에 사용하고, 신경망 깊이가 얕은 모델을 활용하는 경우, 정확도가 중요한 요소라면, 이러한 조건에서 더 높은 정확도를 보이는 정적 가지치기를 선택하는 것이 바람직하다. 반면에, 추론 시간이나 메모리 효율성이 중요한 상황에서는 동적 가지치기 방식의 선택이 더 적절하다. 동적 가지치기가 추론 과정의 속도를 높이는 데 더욱 효과적이며 메모리 사용량을 훨씬 더 효율적으로 관리할 수 있기 때문이다.

본 논문에서는 가지치기 방식에 대한 성능 비교를 중점적으로 다루었지만, 네트워크 슬리밍, 양자화, 그룹 컨볼루션 방식 등과 같은 다른 경량화 기법들의 성능 비교에 관한 추가적인 연구가 필요할 것으로 보인다.

References

[1] V. K. Prasad, P. Bhattacharya, D. Maru, S. Tanwar, A. Verma, A. Singh, A. K. Tiwari, R. Sharma, A. Alkhayyat, F.-E. Țurcanu, M. S. Raboaca, "Federated Learning for the Internet-of-Medical-Things: A Survey," Mathematics, Vol. 11, No. 1, pp. 1-151, 2022. DOI: https://doi.org/10.3390/math11010151

[2] M. Ham, J. J. Moon, G. Lim, W. Song, J. Jung, H. Ahn, S. Woo, Y. Cho, J. Park, S. Oh, H.-S. Kim, "NNStreamer: Stream Processing Paradigm for Neural Networks, Toward Efficient Development and Execution of On-Device AI Applications," CoRR, Vol. abs/1901.04985, 2019. DOI: https://doi.org/10.48550/arXiv.1901.04985

[3] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, C. Zhang, "Proceedings of the IEEE International Conference on Computer Vision (ICCV)," pp. 2736-2744, 2017. DOI: https://doi.org/10.1109/ICCV.1995.466933

[4] N. Fragoulis, I. Theodorakopoulos, V. Pothos, E. Vassalos, "Dynamic Pruning of CNN networks. In 2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA)," IEEE, 2019. DOI: https://doi.org/10.1109/IISA.2019.8900711

[5] X. Gao, Y. Zhao, Ł. Dudziak, R. Mullins, C. Xu, "Dynamic Channel Pruning: Feature Boosting and Suppression," ICLR 2019 Conference, 2018. DOI: https://doi.org/10.48550/arXiv.1810.05331

[6] S. Han, H. Mao, W.J. Dally, "Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding," arXiv: Computer Vision and Pattern Recognition, 2015. DOI: https://doi.org/10.48550/arXiv.1510.00149

[7] B. Hassibi 등, "Optimal Brain Surgeon and General Network Pruning," Proc. of 1993 IEEE Intl. Conference on Neural Networks, 1993. DOI: https://doi.org/10.1109/ICNN.1993.298572

[8] F. Daghero, D.J. Pagliari, M. Poncino, "Energy-efficient deep learning inference on edge devices," In Advances in Computers, Vol. 122, pp. 247-301, Department of Control and Computer Engineering, Politecnico di Torino, 2021. DOI: https://doi.org/10.1016/bs.adcom.2020.07.002

[9] Glenn-Jocher, "YOLOv5", Github, Available: https://github.com/ultralytics/yolov5

[10] Y. B. Sarıdoğan, "Traffic Detection Project dataset," Kaggle, Available: https://www.kaggle.com/datasets/yusufberksardoan/traffic-detection-project/data

[11] N.K. Jha, R. Saini, S. Nag, S. Mittal, "E2GC: Energy-efficient Group Convolution in Deep Neural Networks," In 2020 33rd International Conference on VLSI Design and 2020 19th International Conference on Embedded Systems (VLSID), 2020. DOI: https://doi.org/10.48550/arXiv.2006.15100

[12] A.-T. Shumba, T. Montanaro, I. Sergi, L. Fachechi, M. De Vittorio, L. Patrono, "Leveraging IoT-Aware Technologies and AI Techniques for Real-Time Critical Healthcare Applications," Sensors, Vol. 22, No. 19, 7675, 2022. DOI: https://doi.org/10.3390/s22197675

[13] S. Lee, H. Ahn, J. Seo, Y. Chung, D. Park, S. Pan, "Practical Monitoring of Undergrown Pigs for IoT-Based Large-Scale Smart Farm," IEEE Access, Vol. 7, pp. 173796-173810, 2019.
DOI: https://dx.doi.org/10.1109/ACCESS.2019.2955761

[14] Hong-Jin Park, "Trend Analysis of Korea Papers in the Fields of 'Artificial Intelligence', 'Machine Learning' and 'Deep Learning'," Journal of Korea Institute of Information, Electronics, and Communication Technology, Vol. 13, No. 4, pp. 283-292, 2020.
DOI: https://dx.doi.org/10.17661/jkiiect.2020.13.4.283

[15] Kyoung-Chul Kim, Dasom Seo, Inchan Choi, Young-Ki Hong, Gookhwan Kim, and Kyung-Do Kwon, "Development of Vespa velutina Monitoring System Based on Deep Learning," Journal of the Korea Academia-Industrial cooperation Society, Vol. 22, No. 10, pp. 31-36, 2021.
DOI: 10.5762/KAIS.2021.22.10.31

[16] Byoung-Guk Min and Sun-Ho Min, "Research on Home Digital Healthcare Urine Analyzer," The Journal of Korean Institute of Information Technology, Vol. 19, No. 2, pp. 11-19, 2021.
DOI: 10.14801/jkiit.2021.19.2.11

[17] Namhi Kang, "Development of an intelligent edge computing device equipped with on-device AI vision model," The Journal of The Institute of Internet, Broadcasting and Communication (IIBC), Vol. 22, No. 5, pp.17-22, 2022.
DOI: https://doi.org/10.7236/JIIBC.2022.22.5.17

저 자 소 개

주 혜 현(비회원)

- 2020년 ~ 현재 : 덕성여자대학교 사이버보안전공
- 관심분야 : 컴퓨터 비전, 온디바이스 AI, 사물인터넷 보안

강 남 희(종신회원)

- 1999년 : 숭실대학교 정보통신공학과 (공학사)
- 2001년 : 숭실대학교 정보통신대학원 (공학석사)
- 2005년 : University of Siegen 컴퓨터공학과 (공학박사-인터넷및시스템보안)
- 2009년 ~ 현재 : 덕성여대 미래인재대학 데이터사이언스학과 정교수
- 관심분야 : 유무선 인터넷통신, 인터넷보안, 시스템 보안, 사물인터넷 보안, 온디바이스 AI