ISSN 2383-630X(Print) / ISSN 2383-6296(Online)
Journal of KIISE, Vol. 51, No. 9, pp. 792-800, 2024. 9
https://doi.org/10.5626/JOK.2024.51.9.792

Octave-YOLO: 실시간 객체 탐지를 위한 직접적 다중 스케일 특징 융합

(Octave-YOLO: Direct Multi-scale Feature Fusion for Object Detection)

신 상 준 †　　신 동 군 ††
(Sangjune Shin)　(Dongkun Shin)

요 약 최근 객체 탐지 분야에서는 다양한 크기의 객체를 감지하기 위하여 여러 스케일의 특징 맵들을 융합하는 다중 스케일 특징 융합을 위한 많은 연구가 진행되고 있다. 이러한 다중 스케일의 특징맵을 융합하기 위해, FPN(Feature Pyramid Network)이나 PANet(Path Aggregation Network) 등의 다양한 네트워크 구조가 제시되었다. FPN을 개선한 PANet은 하향식 경로뿐만 아니라 상향식 경로를 추가하여 객체 탐지 분야에서 큰 성능 향상을 이루었다. 그러나 기존 PANet에서의 다중 스케일 특징 융합을 위한 업스케일링 또는 다운스케일링 과정은 원래 특징 맵에서 보존된 저수준 또는 고수준 정보의 손실을 야기했다. 본 논문에서는 옥타브 합성곱(Octave Convolution)을 통한 별도의 처리 없이 다양한 크기의 특징 맵을 원활하게 융합할 수 있는 Octave C2f 모듈을 제안하여 정확성을 향상시키고 계산 복잡도를 줄였다. PASCAL VOC 및 MS COCO 데이터 세트를 사용한 실험 결과, YOLOv8 기본 모델과 비교하여 개선된 정확성과 감소된 계산 작업량 및 매개변수 개수를 확인할 수 있었다.

키워드: 객체 탐지, YOLO, octave convolution, 다중 스케일 특징 융합

Abstract In object detection research, multiscale feature fusion—combining feature maps of different scales to detect objects of varying sizes—has become a critical focus. Network structures like Feature Pyramid Networks (FPNs) and Path Aggregation Networks (PANets) have been developed to address this challenge. PANet, an enhancement of FPN, integrates both top-down and bottom-up pathways, leading to significant improvements in object detection performance. However, during multiscale feature fusion, PANet's upscaling and downscaling processes can result in the loss of crucial low- or high-level information from the original feature maps. In this paper, we introduce the Octave C2f module, which employs octave convolution to seamlessly fuse feature maps of different sizes without the need for additional processing. This innovative approach enhances accuracy while reducing computational complexity. Experimental results on the PASCAL VOC and MS COCO datasets demonstrate improved accuracy, reduced computational effort, and a decrease in parameter count compared to the default YOLOv8 model.

Keywords: object detection, YOLO, octave convolution, multi-scale feature fusion

· 이 논문은 2024년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임(No.IITP-2017-0-00914, 지능형 IoT 장치용 소프트웨어 프레임워크)
· 이 논문은 2023 한국소프트웨어종합학술대회에서 'Octave-YOLO: Octave convolution을 통한 직접적 다중 스케일 특징 융합'의 제목으로 발표된 논문을 확장한 것임

† 비 회 원 : 성균관대학교 전자전기컴퓨터공학과 학생
sangjune97@g.skku.edu
†† 종신회원 : 성균관대학교 소프트웨어학과 교수
(Sungkyunkwan Univ.)
dongkun@skku.edu
(Corresponding author임)

논문접수 : 2024년 4월 3일
(Received 3 April 2024)
논문수정 : 2024년 8월 9일
(Revised 9 August 2024)
심사완료 : 2024년 8월 12일
(Accepted 12 August 2024)

정보과학회논문지 제51권 제9호(2024. 9)

1. 서 론

드론, 얼굴 인식, 자율 주행 등 다양한 분야에서 사용되는 객체 탐지 기술은 이미지 내에서 객체의 위치를 식별하고 클래스를 분류하는 기술이다. 이 중 YOLO 시리즈는 가장 광범위하게 활용되고 있는 대표적인 객체 탐지기 중 하나이다. 2015년 YOLO[1]가 처음 등장한 이후, YOLO 시리즈는 지속적으로 개선되어왔으며, 2023년에 등장한 가장 최신 모델인 YOLOv8[2]까지 이어져 왔다. YOLO 구조는 특징 추출을 위한 백본(Backbone) 계층, 다양한 규모의 특징 맵을 처리하고 통합하는 넥(Neck) 계층, 그리고 객체 분류 및 경계 상자 좌표를 추론하는 헤드(Head) 계층으로 이루어져 있다.

초기의 YOLO 시리즈인 YOLOv3[3]에서는 다중 스케일의 특징 맵을 융합하는 넥 계층에 Feature Pyramid Network (FPN)[4] 구조를 사용하였다. 일반적인 CNN 모델은 객체 탐지를 위해 백본 네트워크의 마지막 합성곱 계층의 특징 맵에만 의존하는 경우가 많아 다양한 크기의 객체를 감지할 때 유연성이 떨어졌다. 그러나 FPN은 하향식 경로를 통해 여러 합성곱 계층의 특징 맵을 통합하여 다양한 크기의 물체를 감지할 수 있도록 함으로써 이러한 한계를 해결했다. YOLOv4[5]의 넥 계층에서는 하향식 경로와 상향식 경로를 모두 통합하는 Path Aggregation Network(PANet)[6] 구조를 도입했다. PANet은 저수준 및 고수준 특징 맵을 완전히 통합하여 특징 맵의 세부사항과 표현력을 향상시킴으로써 다양한 규모의 객체에 대한 적응성을 개선했다. 하지만 이러한 접근 방식은 다양한 크기의 특징 맵 간의 정보를 교환하는 데 어려움이 있었다. 특히 업샘플링 또는 다운샘플링을 통한 특징 맵 크기 조정은 원래 해상도에서 중요한 특징 정보가 손실될 위험이 있다.

본 논문에서는 현재 가장 좋은 성능을 보이고 있는 객체 탐지 네트워크인 YOLOv8을 기반으로 넥(Neck) 계층을 개선한 Octave-YOLO를 제안한다. Octave-YOLO는 업샘플링이나 다운샘플링 등 별도의 처리 없이 다양한 크기의 입력 및 출력이 가능한 옥타브 합성곱 (Octave convolution)을 사용하여 YOLOv8의 PANet을 향상시켜 다양한 크기의 입력과 출력을 직접적으로 융합할 수 있다.

2. 관련 연구

객체 탐지는 주어진 입력 이미지에서 객체의 위치를 파악하고 동시에 해당 객체의 클래스를 분류하는 것을 목표로 한다. 최근에는 객체 탐지 문제를 해결하기 위해 다양한 딥러닝 기반의 접근 방식이 제안되고 있다. 이러한 접근 방식은 크게 2-stage detector와 1-stage detector의 두 가지로 분류 할 수 있다.

2-stage detector는 순차적으로 영역 제안을 통해 영역을 추출하고, 이후 추출된 영역 안에서 클래스를 추론하는 R-CNN[7] 구조를 기반으로 한다. 이후 R-CNN을 개선하여 더 효율적인 모델인 fast R-CNN[8], mask R-CNN[9], cascade R-CNN[10]등이 제안되었다. 이러한 2-stage detector는 성능이 매우 우수하지만 지연시간이 너무 길어 실시간 객체 탐지에는 사용하기 어렵다는 문제가 있었다.

이러한 문제를 극복하기 위해 제안된 SSD[11], YOLO[1], RetinaNet[12] 등의 1-stage detector는 2-stage detector에 비해 정확도는 약간 떨어지지만, 훨씬 빠른 추론으로 속도 개선을 이루었다. 이들은 영역 제안과 클래스 분류를 분리하지 않고 하나의 회귀 문제로 취급한다.

최근 몇 년간 개발된 객체 감지기는 백본 계층과 헤드 계층 사이에 중간 특징 맵을 수집하는 넥 계층을 추가하였다. 이러한 넥 계층은 Feature Pyramid Network (FPN)[3], Path Aggregation Network(PANet)[4], BiFPN[13], NAS-FPN[14] 등이 포함되며, 중간 특징 맵들을 통합하여 특징 맵의 세부사항과 표현력을 향상시키는 역할을 수행한다.

CNN 기반 감지기 이외에도 최근 연구에서는 트랜스포머 기반 감지기 또한 등장했다. Detection Transformer (DETR)[15]는 기존 CNN 기반 객체 탐지기에서 널리 사용되는 앵커(Anchor)나 중복된 경계 상자를 제거하기 위한 Non-maximum Suppression(NMS) 등의 전처리 및 후처리 작업을 사용하지 않아 객체 검출 파이프라인을 단순화하고, 하나의 네트워크만을 사용해 추론을 하는 End-to-end 객체 검출을 구현했다. 그 결과 CNN 기반 객체 탐지기보다 훨씬 높은 정확도를 달성하였으나, 트랜스포머의 높은 계산 비용으로 인해 실시간 객체 감지를 구현하기는 어려웠다. RT-DETR[16]과 같은 일부 실시간 객체 탐지기가 제안되었지만, 트랜스포머 기반 객체 탐지기는 대규모의 사전 학습된 백본 없이 새로운 분야에 적용하기 매우 어렵기 때문에 여전히 실시간 객체 탐지 분야에서는 YOLO 시리즈가 활약하고 있다.

3. 배 경

3.1 YOLOv8

YOLOv8[2]은 YOLO 시리즈의 최신 모델이며, 본 연구의 기준선으로 사용되었다. YOLOv8의 필수 구성 요소는 YOLOv5[17]의 C3(Convolution 3) 모듈을 개선한 C2f(Convolution 2 fast) 모듈이다. YOLOv5의 백본인 CSPDarkNet에서 사용된 C3 모듈은 Cross-Stage Partial Network(CSPNet)[18]에서 제안된 CSP 구조를 3개의 합성곱(Convolution) 계층으로 구현하여 정확도를 유지하면서도 연산량을 줄여 좋은 결과를 얻었다.

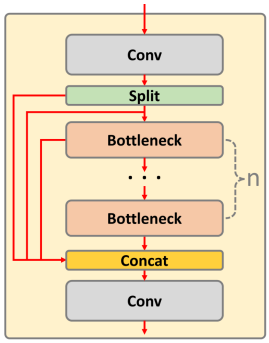


그림 1 C2f 모듈의 세부 구조
Fig. 1 Detailed structure of C2f module

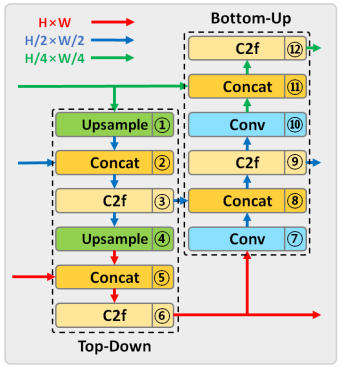


그림 2 YOLOv8의 넥(Neck) 구조
Fig. 2 Neck structure of YOLOv8

YOLOv8 백본 계층은 ELAN[19] 모듈에서 도입한 개념과 일치하는 C2f 모듈을 채택했다. C2f 모듈은 그림 1과 같이 1×1 커널을 사용하는 2개의 합성곱 계층과 3×3 커널을 사용하는 일련의 합성곱 계층을 가진 병목(Bottleneck) 블록으로 구성되어 C3모듈보다 빠른 속도로 연산을 수행할 수 있다. 모듈의 입력 특징 맵은 두 부분으로 나뉘는데, 하나는 병목 블록을 우회하고 다른 하나는 병목 블록을 통과한다. 병목 블록을 통과하며 연속된 각 합성곱 계층의 출력은 결합(Concatenate)되어 누적된다. 병목 블록을 모두 통과한 후, 이렇게 누적된 특징 맵들은 마지막으로 병목 블록을 우회한 특징 맵과도 결합되어 최종 1×1 합성곱 계층을 거쳐 C2f 모듈의 출력 특징 맵이 된다.

객체 탐지기의 백본은 여러 개의 합성곱 계층으로 구성된다. 계층이 깊어질수록 특징 맵은 점점 더 추상화되고, 다운샘플링으로 인해 특징 맵의 해상도가 감소하여 공간 정보뿐만 아니라 세밀한 특징도 손실될 수 있다.

미세한 특징을 보존하기 위해 YOLOv3[3]에는 FPN이 도입되었다. FPN은 넥(Neck)에 하향식 경로를 추가하여 동일한 크기의 특징 맵들을 병합한다. 그러나 모델이 깊어질수록 하위 계층에 위치한 특징 맵은 상위 계층에 도달하기 위해 더욱 긴 경로를 거쳐야 한다. 이로 인해 여러 계층을 통과하며 저수준 정보가 손실 될 수 있다.

이러한 문제를 극복하기 위해 YOLOv8의 넥 계층에는 PANet 구조가 도입되었다. YOLOv4에서 처음 도입된 PANet은 YOLOv3의 FPN 구조를 개선한 구조이다. 저수준 특징 맵을 고수준 특징 맵에 완전히 융합하기 위해 상향식 경로뿐만 아니라 하향식 경로도 추가했다.

그림 2에서 볼 수 있듯이 YOLOv8의 넥에는 각 경로에는 두 개의 C2f 모듈이 있다.

하향식 경로의 C2f 모듈 ③은 백본으로부터 두 가지 크기의 특징 맵을 수신한다. 먼저 작은 크기의 특징 맵은①에서 최근접 이웃 보간법을 통해 업샘플링되어 큰 크기의 특징 맵과 동일한 해상도로 맞춰진 뒤, ②에서 결합되어 ③의 입력으로 주어진다. C2f 모듈 ⑥는 백본으로부터 큰 크기의 특징 맵을, ③으로부터 작은 크기의 특징 맵을 받는다. ③으로부터 받은 작은 크기의 특징 맵은 ④에서 업샘플링 되고, ⑤에서 백본으로부터 받은 큰 크기의 특징 맵과 결합되어 ⑥의 연산을 진행한다. 이러한 방법은 업샘플링으로 인해 작은 스케일의 정보를 가진 특징 맵이 큰 스케일의 병목 블록을 통과하게 되므로 계산 비용이 증가한다.

또한 상향식 경로의 C2f 모듈 ⑨는 하향식 경로의 ⑥으로부터 큰 크기의 특징 맵을, ③으로부터 작은 크기의 특징 맵을 받는다. 큰 크기의 특징 맵은 보폭이 2인 합성곱 계층인 ⑦을 통과하여 다운샘플링한 된 후, ⑧에서 작은 크기의 특징 맵과 결합하여 ⑨의 입력으로 사용된다. 마지막으로 C2f 모듈 ⑫는 ⑨로부터 큰 크기의 특징 맵을, 백본으로부터 작은 크기의 특징 맵을 받는다. 마찬가지로 ⑩에서 다운샘플링 된 큰 크기의 특징 맵은 ⑪에서 작은 크기의 특징 맵과 결합되어 ⑫의 입력으로 사용된다. 그러나 이는 다운샘플링으로 인해 큰 스케일의 특징 맵의 정보의 손실을 야기할 수 있다.

3.2 옥타브 합성곱

컴퓨터 비전 알고리즘의 효율성과 정확성을 향상하기 위해 등장한 옥타브 합성곱(Octave Convolution)[20]은 다양한 주파수의 특징을 추출하면서도 계산 비용을 줄일 수 있다. 일반적으로 자연물을 촬영한 이미지는 부드럽게 변화하는 구조를 표현하는 낮은 공간 주파수 요소와 빠르게 급격히 변화하는 미세한 세부 사항을 표현하

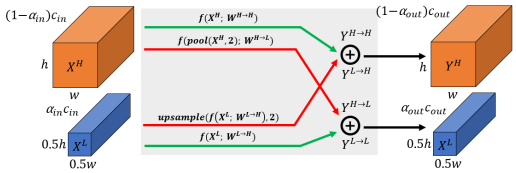


그림 3 옥타브 합성곱 계층의 세부 구조
Fig. 3 Detailed structure of octave convolution

는 높은 공간 주파수 요소로 구성된다. 또한, 이미지의 해상도에 따라 포함되는 공간 주파수 요소가 달라진다.

네트워크가 더 깊어짐에 따라, 특징 맵의 해상도는 줄어든다. 이 경우 원본 이미지 내 객체의 세부 정보를 명확하게 표현하기 어려워진다. 대신 윤곽과 구조적인 특성이 나타난다. 이는 낮은 주파수 이미지 범주에 속한다.

반면, 얕은 깊이에서 추출된 특징 맵은 더 큰 해상도를 가지며 객체의 세부 정보가 더 명확하게 나타난다. 이는 높은 주파수 이미지의 범주에 속한다.

옥타브 합성곱은 고해상도의 고주파수 및 저해상도의 저주파수 특징 맵 간의 상호 작용 및 융합을 포함하며, 그 구조는 그림 3에 나타나 있다. 녹색 화살표는 정보의 업데이트를 나타내며, 빨간색 화살표는 서로 다른 주파수 간의 정보 교환을 나타낸다. 옥타브 합성곱의 출력은 식(1) 및 식(2)와 같이 나타낼 수 있다.

$$Y^H = Y^{H\to H} + Y^{L\to H} = f(X^H; W^{H\to H}) + upsample(f(X^L; W^{L\to H}), 2) \quad (1)$$

$$Y^L = Y^{H\to L} + Y^{L\to L} = f(X^L; W^{L\to L}) + f(pool(X^H, 2); W^{H\to L}) \quad (2)$$

여기서 $Y^{P\to Q}$는 특징 맵 그룹 간의 합성곱 업데이트를 나타낸다. $Y^{H\to H}$와 $Y^{L\to L}$은 같은 주파수의 특징 맵 그룹 내 업데이트를 나타내며, $Y^{H\to L}$과 $Y^{L\to H}$은 다른 주파수의 특징 맵 그룹 간의 상호작용을 나타낸다. $f(X;W)$는 매개변수 W를 가지고 있는 합성곱을 나타내고, $pool(X,K)$는 커널 크기 $K\times K$ 및 보폭 K를 사용한 평균 풀링 연산을 나타낸다. $upsample(X,K)$는 K 배수 최근접 이웃 방법을 통한 업샘플링 연산을 나타낸다. C^{in}은 옥타브 합성곱의 전체 입력 특징 맵의 채널 수를 나타내고, C^{out}은 옥타브 합성곱의 전체 출력 특징 맵의 채널 수를 나타낸다. α^{in}은 저주파수 입력 특징 맵에 할당된 채널의 비율을 나타내고, α^{out}은 저주파수 출력 특징 맵에 할당된 채널의 비율을 나타낸다.

옥타브 합성곱은 전통적인 합성곱 연산을 대체할 수 있으며, 일반적으로 더 높은 성능을 달성하면서도 연산량을 줄일 수 있다. 이는 원본 특징 맵 대비 크기가 1/4인 저주파 특징 맵이 원본 특징 맵 채널의 일부분을 차지하기 때문이다.

4. 본 론

4.1 Octave-YOLO

그림 4은 본 논문에서 제안된 Octave-YOLO의 전체적인 구조를 보여준다.

해당 네트워크의 구조는 원본 YOLOv8 구조에서 넥을 개선한 것이다. 네트워크의 백본 구조는 스템(Stem) 계층과 4단계 스테이지로 구성된다. 스템 계층는 보폭이 2인 합성곱 계층이다. 각 스테이지 블록은 마찬가지로 보폭인 2의 합성곱 계층을 가지고 있으며, 이들은 특징 맵의 크기를 줄이는 다운샘플링 모듈로 활용된다. 백본에서 전달된 특징 맵은 Octave C2f 모듈을 통해 넥에서 융합된다. 기존 YOLOv8과 달리 Octave-YOLO의 개선된 넥 구조는 특징 맵의 크기 조절 없이 다양한 스케일의 입출력을 가질 수 있는 Octave C2f 모듈을 통해 추가 연산이나 정보 손실 없이 효율적으로 특징을 융합할 수 있다. 마지막으로 헤드 계층은 넥으로부터 융

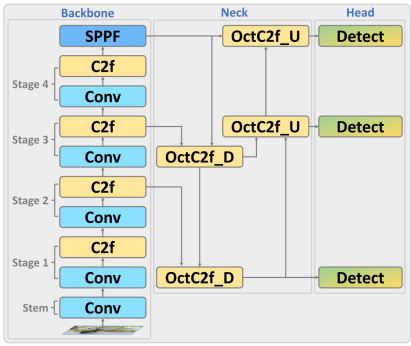


그림 4 Octave-YOLO의 전체 구조
Fig. 4 Overall architecture of Octave-YOLO

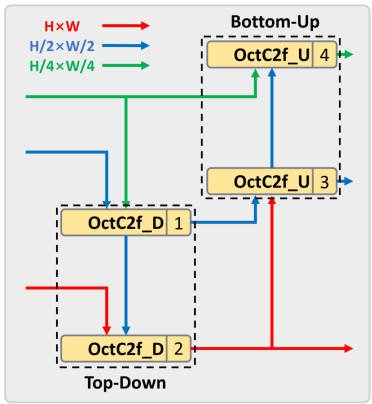


그림 5 Octave-YOLO의 넥 구조
Fig. 5 Neck structure of Octave-YOLO

합된 특징 맵을 입력받아 객체의 클래스, 크기, 위치를 예측한다.

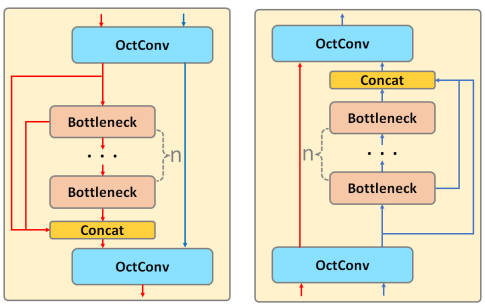
Octave-YOLO의 자세한 넥 구조는 그림 5에 나와 있다. 업샘플링 계층이나 다운샘플링 계층, 또는 결합 계층 없이 크기가 다른 특징 맵을 직접 융합하기 위해 Octave C2f 모듈을 사용한다. 우리는 각각 하향식 경로와 상향식 경로에 사용되는 Octave C2f_Downpath와 Octave C2f_Uppath라는 두 가지 종류의 Octave C2f 모듈을 설계했다.

4.2 Octave C2f Module

기존의 C2f 모듈은 전통적인 합성곱 연산을 사용했기 때문에 다양한 크기의 피처 맵을 융합하는 데 한계가 있었다. 이러한 한계를 해결하기 위해 우리는 옥타브 합성곱을 통해 C2f 모듈을 개선하여 업샘플링이나 다운샘플링 없이도 다양한 크기의 피처 맵을 직접 연결할 수 있는 Octave C2f 모듈을 고안했다.

4.2.1 Octave C2f_Downpath

하향식 경로에서 사용되는 Octave C2f_downpath 모듈은 서로 다른 크기의 입력 특징 맵을 직접 처리하며, 이는 그림 6 (a)에 표시되어 있다. 먼저 α^{out} 값이 0.5인 옥타브 합성곱 계층을 통해 크기가 다른 특징 맵 간에 정보 교환이 이루어진다. 그 후 크기가 큰 특징 맵만 병목 블록을 통과하고 작은 특징 맵은 그대로 유지된다. 마지막으로 병목 블록을 통과한 작은 특징 맵과 큰 특징 맵을 모두 활용하여 α^{out} 값이 0인 옥타브 합성곱 계층을 거쳐 저수준 및 고수준 특징이 융합된 큰 크기의 특징 맵을 생성한다. Octave C2f_downpath 모듈을 사용하면 YOLOv8의 하향식 경로의 C2f 모듈과 동일한 크기와 채널 수를 가진 특징 맵을 생성할 수 있다. 또한 특징 융합 과정에서 업샘플링 작업 없이 더 작은 크기의 특징 맵이 직접 통합되므로 더 풍부한 저수준 특징 정보를 보존하며 동시에 연산량을 줄일 수 있다.



(a) Top-Down (b) Bottom-Up

그림 6 Octave C2f 모듈의 세부 구조
Fig. 6 Detailed structure of Octave C2f module

4.2.2 Octave C2f_Uppath

Octave C2f_uppath 모듈은 Octave C2f_downpath 모듈과 마찬가지로 다양한 크기의 입력 특징 맵을 직접 처리할 수 있으며, 그림 6 (b)에 표시되어 있다. α^{out} 값이 0.5인 옥타브 합성곱 계층을 통해 크기가 다른 특징 맵 간에 정보 교환이 이루어진 뒤, 크기가 작은 특징 맵이 병목 블록을 통과하고 큰 특징 맵은 그대로 유지된다. 이후 병목 블록을 통과한 큰 특징 맵과 작은 특징 맵은 α^{out} 값이 1인 옥타브 합성곱 계층을 통해 다양한 스케일의 특징이 융합된 작은 크기의 특징 맵을 출력한다. Octave C2f_uppath 모듈은 YOLOv8의 상향식 경로를 대체할 수 있으며, 기존 C2f 모듈과 달리 고수준 입력 특징 맵을 다운샘플링하지 않음으로써 높은 수준의 특징 정보를 보존한다.

5. 실 험

5.1 실험 환경

본 논문에서는 Octave-YOLO의 성능을 평가하기 위해 객체 탐지 분야에서 비교를 위해 널리 사용되는데이터 세트인 PASCAL VOC 및 MS COCO 훈련 및 검증 데이터 세트를 사용해 500 에포크 동안 네트워크를 훈련했다.

PASCAL VOC 데이터 세트는 탈 것, 가정용품, 동물 등을 포함한 20종류의 객체 분류를 가지며, 16,551개의 학습 이미지와 4,952개의 테스트 이미지로 구성된다.

MS COCO 데이터 세트는 VOC 데이터 세트보다 훨씬 더 다양한 80종류의 객체 분류가 존재하며, 약 11만 8천개의 학습 이미지와 4만개의 테스트 이미지를 포함한다.

학습에는 SGD 옵티마이저를 사용했으며, 학습률은 코사인 스케줄러를 사용하여 초기 학습률 1e-02, 최종 학습률 1e-05로 설정했다. 가중치 감쇠는 5e-03이며, 모멘텀은 0.937로 설정하였다. 또한 배치 크기는 64로, 네트워크에 입력되는 이미지의 크기는 640×640으로 통일하여 훈련되었다.

운영체제는 Ubuntu 20.04.1을 사용하였으며, 데스크톱 컴퓨팅 환경으로 Python 3.10, PyTorch 2.2.0, Cuda 11.7을 사용하였다. 하드웨어는 NVIDIA RTX 4090 그래픽 카드가 사용되었다. 네트워크의 세부 구현 코드는 Ultralytics 8.1.24 버전을 기반으로 수정되었다.

모델별 초당 프레임 수(FPS) 측정은 배치 크기가 1로 측정이 되었으며, 하드웨어는 NVIDIA Titan X 그래픽 카드가 사용되었다.

5.2 모델별 세부 설정

Octave-YOLO는 YOLOv8-L(Large) 모델을 기반으로 다양한 크기의 다양한 네트워크 모델을 확장 가능한

표 1 Octave YOLO 모델의 백본 파라미터
Table 1 Octave-YOLO model's parameters in the backbone

Model	Stem	Stage 1		Stage 2		Stage 3		Stage 4		SPPF	
Size	c_out	c_out	n	c_out	n	c_out	n	c_out	n	c_out	n
n	16	32	1	64	2	128	2	256	1	1	256
s	32	64	1	128	2	256	2	512	1	1	512
m	48	96	2	192	4	384	4	576	2	2	576
l	64	128	3	256	6	512	6	512	3	3	512
x	64	128	3	256	6	512	6	512	3	3	512

표 2 Octave YOLO 모델의 넥 파라미터
Table 2 Octave-YOLO model's parameters in the neck

Model Size	Top-down				Bottom-up			
	1		2		3		4	
	c_out	n	c_out	n	c_out	n	c_out	n
n	128	1	64	1	128	1	256	1
s	256	1	128	1	256	1	512	1
m	384	2	192	2	384	2	576	2
l	512	3	256	3	512	3	512	3
x	640	3	320	3	640	3	640	3

방식으로 생성할 수 있다. 이는 C2f 모듈 내부의 병목 구조의 깊이를 줄이고 합성곱 계층의 채널 수를 줄이는 것을 포함한다. 그 결과 nano, small, medium, large, extra large 크기의 5가지 종류의 모델을 만들 수 있다.

Extra large 모델은 가장 강력한 탐지 정확도를 달성하지만 크기 매개변수가 가장 크고 계산 비용이 가장 높다. 반면 nano 모델은 파라미터 수가 적고 처리 속도가 빠르지만 정확도가 떨어진다.

이러한 확장성 덕분에 드론이나 자율주행차와 같은 엣지 컴퓨팅 장치부터 다중 GPU 서버에 이르기까지 다양한 환경에서 Octave-YOLO를 활용할 수 있다.

표 1 및 표 2는 다양한 크기의 모델의 세부 구현을 보여준다. 표1은 백본의 각 스테이지의 출력 채널과 병목 블록 깊이 n을 보여준다. 예를 들어 large 모델의 경우 백본의 1단계는 128개의 출력 채널을 가진 합성곱 계층과 병목 블록이 3번 반복되는 구조를 가진 C2f 모듈로 구성된다. 표2은 각각 넥에서 하향식 경로와 상향식 경로를 형성하는 Octave-C2f 모듈의 출력 채널과 병목 블록 크기 n을 보여준다.

5.3 실험 결과

우리는 YOLO의 최신 버전인 YOLOv8을 기준선으로 선택했다. 표 3와 표 4에서는 COCO 데이터 세트를, 그림 7에서는 PASCAL VOC 데이터 세트를 사용하여 학습을 진행하였으며, Octave-YOLO 모델과 다른 객체 탐지 모델들 간의 비교 분석을 제시한다.

표 3의 실험은 MS COCO 훈련 데이터 세트를 사용하여 훈련 후 테스트를 진행했다. nano 모델에서 우리의 접근 방식은 기준선에 비해 6.67% 더 적은 매개변수를 사용하고 2.30% 더 적은 계산을 요구하면서 2.1 더 높은 평균 정밀도(AP)를 보여준다. small 모델은, 우리의 기법이 매개변수를 6.31% 줄이고, 계산량을 4.89% 줄이며, AP를 1.1 향상시킨다. medium 모델의 경우, 매개변수 6.20% 감소, 계산량 3.81% 감소, AP 1.1 증가가

표 3 COCO 데이터 세트를 통한 YOLOv8과의 비교
Table 3 Comparison with YOLOv8 using COCO dataset

Model	#Param.	FLOPS	AP^{val}
YOLOv8-N	3.0M	8.7G	35.2%
Octave-YOLO-N	**2.8M**	**8.5G**	**37.3%**
improvement	-6.67%	-2.30%	+2.1
YOLOv8-S	11.1M	28.6G	42.7%
Octave-YOLO-S	**10.4M**	**27.2G**	**43.8%**
improvement	-6.31%	-4.89%	+1.1
YOLOv8-M	25.8M	78.9G	48.3%
Octave-YOLO-M	**24.2M**	**75.9G**	**49.2%**
improvement	-6.20%	-3.81%	+1.1
YOLOv8-L	43.6M	165.2G	50.4%
Octave-YOLO-L	**40.7M**	**159.8G**	**51.2%**
improvement	-6.65%	-3.27%	+1.2
YOLOv8-X	68.2M	257.8G	51.5%
Octave-YOLO-X	**63.6M**	**249.2G**	**51.9%**
improvement	-6.74%	-3.33%	+0.4

표 4 다양한 객체 탐지 모델과의 비교
Table 4 Comparison with various object detectors

Model	Backbone	FPS	AP^{val}
2-stage detectors			
Fast R-CNN	VGG16	0.5	19.7%
Faster R-CNN	ResNet-101-C4	5	34.9%
Faster R-CNN	ResNet-101-FPN	7	36.2%
Mask R-CNN	ResNet-101-FPN	8	38.2%
Mask R-CNN	ResNet-101-FPN	9	39.8%
Cascade R-CNN	ResNet-101-FPN	8	42.8%
1-stage detectors			
SSD513	ResNet-101-SSD	19	31.2%
DSSD513	ResNet-101-DSSD	21	33.2%
RetinaNet	ResNet-101-FPN	8	39.1%
RetinaNet	ResNet-101-FPN	7	40.8%
YOLOv3	Darknet53	35	33.0%
YOLOv4	CSPDarknet53	33	43.5%
YOLOv8-M	CSPDarknet53	57	48.3%
Octave-YOLO-M	CSPDarknet53	**58**	**49.2%**

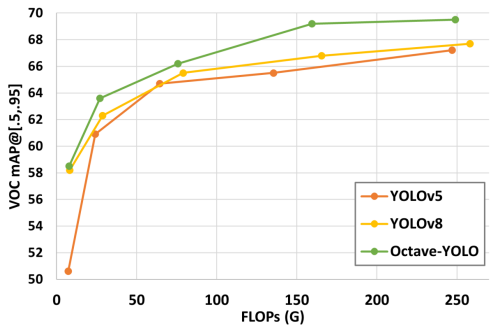


그림 7 VOC 테스트 데이터 세트를 통한 YOLO 시리즈 간 비교

Fig. 7 Comparison of YOLO series using VOC test dataset

관찰되었다. large 모델은 매개변수 6.65%, 계산 3.27% 감소와 함께 AP가 1.2로 성능 향상을 보였다. 또한 extra large 모델은 AP가 0.4% 증가했으며, 이에 따라 매개변수와 계산이 각각 6.74%와 3.33% 감소한 것으로 나타났다.

표 4의 실험에서는 NVIDIA Titan X를 통해 측정한 다양한 객체 탐지 모델들의 FPS를 비교하였다. 2-stage-detector들의 실험 결과는 [21]를 참조하였으며, 1-stage-detector의 실험 결과는 [5]을 참조하였다. 실험 결과에 따르면, 2-stage-detector들은 대부분 10FPS 이하의 낮은 추론 속도로 인하여 실시간 객체 탐지에서 활용하기 어려운 모습을 보인다. 반면, 1-stage-detector들은 2-stage-detector들에 비해 상당히 빠른 추론 속도로 인해 실시간 객체 탐지에 적합하며, 특히 본 논문에서 제안하는 Octave-YOLO는 최신 YOLO 모델인 YOLOv8과 약간 더 빠른 추론속도와 함께 정확도 개선을 함께 이루었다.

그림 7의 실험은 PASCAL VOC 훈련 데이터 세트를 사용하여 YOLOv5 및 YOLOv8와의 비교를 제공한다. 전체적으로 Octave-YOLO는 YOLOv5 및 YOLOv8 보다 연산량 및 정확도 측면에서 모두 우수한 성능을 보인다.

5.4 시각화 및 비교

이 분석에서는 VOC2007 테스트 세트에서 추출한 이미지를 사용하여 YOLOv8-S와 Octave-YOLO-S의 객체 감지 성능을 비교한다. 우리는 다양한 장면과 객체 범주에 따른 차이를 조사했다.

그림 8에서 볼 수 있듯이 버스가 차도에 주차되어 있고 오른쪽 멀리 자동차가 보인다. YOLOv8은 멀리 있는 자동차를 감지하지 못하였지만, Octave-YOLO는 성공적으로 식별했다.

그림 9에서는 여러 사람이 테이블 주위에 앉아 있고

그림 8 YOLOv8과 Octave-YOLO의 객체 탐지 비교 시나리오 1

Fig. 8 Object detection comparison scenario 1 with YOLOv8 and Octave-YOLO

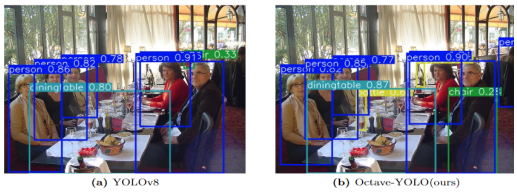


그림 9 YOLOv8과 Octave-YOLO의 객체 탐지 비교 시나리오 2

Fig. 9 Object detection comparison scenario 2 with YOLOv8 and Octave-YOLO

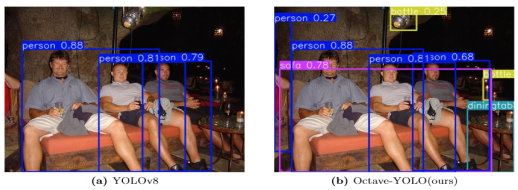


그림 10 YOLOv8과 Octave-YOLO의 객체 탐지 비교 시나리오 3

Fig. 10 Object detection comparison scenario 3 with YOLOv8 and Octave-YOLO

사람과 물체들의 경계 상자가 겹쳐 있다. Octave-YOLO는 겹친 물체들과 식탁 위의 병까지 감지하는 모습을 보이지만, YOLOv8은 사람은 성공적으로 탐지하였지만 겹친 물체들을 식별하는 데는 어려움을 겪었다.

그림 10는 야간에 촬영된 사진을 탐지한다. 이를 통해 저조도 환경에서 사람 이외의 물체를 감지하는 데 있어 YOLOv8의 한계를 확인할 수 있다. 반대로 Octave-YOLO는 열악한 조명 환경에서도 소파, 식탁, 병과 같은 물체를 식별하는 데 강력한 성능을 보인다.

6. 결 론

본 논문에서는 YOLOv8 모델을 기반으로 구축된 다중

스케일 객체 탐지기인 Octave-YOLO를 제안한다. 우리는 특히 YOLOv8 모델의 넥 아키텍처를 집중적으로 개선했다. 기존 C2f 모듈을 활용한 넥은 크기가 다른 맵 간의 특징 융합을 용이하게 하기 위해 업샘플링 또는 다운샘플링을 통해 특징 맵의 크기를 조정해야 했다.

이러한 문제를 극복하기 위해 우리는 옥타브 합성곱을 YOLOv8의 C2f 모듈에 통합한 Octave C2f 모듈을 제안했다. Octave C2f 모듈을 사용하면 다양한 크기의 특징 맵 간에 해상도를 변경하지 않고도 원활한 정보 교환이 가능하다. 결과적으로 Octave-YOLO는 보다 포괄적인 특징 정보를 결합하여 다양한 크기의 물체를 감지하는 능력을 크게 향상시킨다.

PASCAL VOC 및 MS COCO 데이터 세트를 사용한 실험에서 Octave-YOLO는 기준이 되는 YOLOv8 모델보다 뛰어난 성능을 보였다. 평균적으로 약 6%의 파라미터 크기 감소에도 불구하고 Octave-YOLO는 5가지 크기의 모델에서 YOLOv8 보다 모두 우수한 성능을 보였다.

또한 시각화를 통해 실제 물체 감지 작업에서 Octave-YOLO가 YOLOv8에 비해 우수한 성능을 보인다는 것을 확인할 수 있다. 특히 Octave-YOLO는 YOLOv8와 달리 작은 크기의 물체와 경계 상자가 겹치는 물체를 효과적으로 감지하는 성능을 보인다.

제안하는 방법에서는 옥타브 합성곱의 저주파수 입력 비율 및 출력 비율인 α^{in}과 α^{out}을 0.5로 고정하여 사용하였다. 향후 연구로 이 비율의 조정을 통해 더욱 다양한 크기와 다양한 구조의 모델을 확장 가능하도록 하여 임베디드 기기부터 다중 GPU 서버까지 다양한 분야에서 실시간 객체 탐지를 가능하게 하는 방법의 연구를 계획하고 있다.

References

[1] Redmon, J. "You only look once: Unified, real-time object detection." Proc. of the IEEE conference on computer vision and pattern recognition. 2016.

[2] Jocher, G., Chaurasia, A., Qiu, J.: (2023 Jan). Ultralytics YOLO [Online]. Available: HYPERLINK "https//github.com/ultralytics/ultralytics"https://github.com/ultralytics/ultralytics

[3] Redmon, Joseph. "Yolov3: An incremental improvement." arXiv preprint arXiv:1804.02767. 2018.

[4] Lin, Tsung-Yi, et al. "Feature pyramid networks for object detection." Proc. of the IEEE conference on computer vision and pattern recognition. 2017.

[5] Bochkovskiy, Alexey, Chien-Yao Wang, and Hong-Yuan Mark Liao. "Yolov4: Optimal speed and accuracy of object detection." arXiv preprint arXiv:2004.10934. 2020.

[6] Liu, Shu, et al. "Path aggregation network for instance segmentation." Proc. of the IEEE conference on computer vision and pattern recognition. 2018.

[7] Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." Proc. of the IEEE conference on computer vision and pattern recognition. 2014.

[8] Girshick, R. "Fast r-cnn." arXiv preprint arXiv:1504.08083. 2015.

[9] He, Kaiming, et al. "Mask r-cnn." Proc. of the IEEE international conference on computer vision. 2017.

[10] Cai, Zhaowei, and Nuno Vasconcelos. "Cascade r-cnn: Delving into high quality object detection." Proc. of the IEEE conference on computer vision and pattern recognition. 2018.

[11] Liu, Wei, et al. "Ssd: Single shot multibox detector." Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14. Springer International Publishing, 2016.

[12] Lin, Tsung-Yi, et al. "Focal loss for dense object detection." Proc. of the IEEE international conference on computer vision. 2017.

[13] Tan, Mingxing, Ruoming Pang, and Quoc V. Le. "Efficientdet: Scalable and efficient object detection." Proc. of the IEEE/CVF conference on computer vision and pattern recognition. 2020.

[14] Ghiasi, Golnaz, Tsung-Yi Lin, and Quoc V. Le. "Nas-fpn: Learning scalable feature pyramid architecture for object detection." Proc. of the IEEE/CVF conference on computer vision and pattern recognition. 2019.

[15] Carion, Nicolas, et al. "End-to-end object detection with transformers." European conference on computer vision. Cham: Springer International Publishing, 2020.

[16] Zhao, Yian, et al. "Detrs beat yolos on real-time object detection." Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024.

[17] Jocher, G., Chaurasia, A., Qiu, J.: (2020 May). Ultralytics YOLOv5 [Online]. Available: HYPERLINK "https//ithub.com/ultralytics/ultralytics"https://ithub.com/ultralytics/ultralytics

[18] Wang, Chien-Yao, et al. "CSPNet: A new backbone that can enhance learning capability of CNN." Proc. of the IEEE/CVF conference on computer vision and pattern recognition workshops. 2020.

[19] Zhang, Xindong, et al. "Efficient long-range attention network for image super-resolution." European conference on computer vision. Cham: Springer Nature Switzerland, 2022.

[20] Chen, Yunpeng, et al. "Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution." Proc. of the IEEE/CVF international conference on computer vision. 2019.

[21] Lin, Tsung-Yi, et al. "Focal loss for dense object detection." Proc. of the IEEE international conference on computer vision. 2017.

신 상 준
2023년 성균관대학교 컴퓨터공학과 (학사)
2023년~현재 성균관대학교 전자전기컴퓨터공학과 석사과정. 관심분야는 딥러닝 모델 압축, 컴퓨터 비전

신 동 군
정보과학회논문지
제 51 권 제 7 호 참조