석사학위논문

YOLO 기반 딥러닝 모델들의 흡연 감지 성능 비교 연구

인천대학교 정보기술대학원

컴퓨터 전공

남 춘 식

2025년 2월

석사학위청구논문

YOLO 기반 딥러닝 모델들의 흡연 감지 성능 비교 연구

지도교수 이 장 호

이 논문을 석사학위논문으로 제출함

2024년 12월

인천대학교 정보기술대학원

컴퓨터전공

남 춘 식

이 논문을 남춘식의 공학

석사학위 논문으로 인준함

2024년 12월

심사위원장	홍 윤 식	(인)
심사위원	김 우 일	(인)
심사위원	이 장 호	(인)

국문초록

YOLO 기반 모델들의 흡연 감지 성능 비교 연구

현대 사회에서 많은 공공장소가 금연 구역으로 지정되어 있으나, 단속 인력 부족과 흡연자의 비협조적인 태도로 인해 금연 구역 내 흡연 행위를 효과적으로 통제하지 못하는 경우가 종종 발생한다. 이러한 문제를 해결하기 위해 딥러닝 기반의 영상 처리 모델을 활용한 실시간 흡연 감지 시스템을 개발하면, 흡연 행위를 효율적으로 감지하고 경고 메시지 제공 등을 통해 금연 구역에서의 흡연을 방지하며, 위험 시설에서 흡연으로 인한 피해와 사고를 예방하는 데 기여할 수 있다.

많은 물체 감지 딥러닝 모델 중에서 YOLO 계열의 모델은 빠른 실시간 감지와 높은 정확도를 제공하여 연구자 및 산업 종사자들 사이에서 활발하게 연구되고 있는 모델이다. 본 연구에서는 흡연 데이터에 대한 YOLOv5부터 YOLOv10까지 다양한 최근 모델의 성능을 비교하고 검증하여 적용 상황에 따른 최적의 모델을 선정하는 연구를 수행하였다. 실험 결과, YOLOv9의 e 모델이 mAP@50 기준으로 가장 높은 성능을 보였으며, YOLOv8의 x 모델은 적은 파라미터 수에도 불구하고 비교 모델 중 우수한 성능을 나타냈다. 또한, YOLOv9의 t 모델이 적은 파라미터 수와 높은 성능을 동시에 충족시킴으로써 임베디드 환경 같은 모델 경량화가 필요한 곳에는 적합한 선택으로 평가된다.

다만, 흡연 감지 성능은 모델 구조뿐만 아니라 데이터 세트 구성, 학습률, 배치 크기 등 하이퍼파라미터 설정에 따라서도 달라질 수 있음이 확인되었다. 그러므로 성능 향상과 객관적인 결론을 도출하기 위해 대규모 고품질의 데이터 세트와 다양한 조건에서 추가 실험이 필요하다.

본 연구 결과는 흡연 감지 시스템 개발을 통해 사회 문제 해결에 일조하고 객체 감지 분야에 활용 자료가 되어, 영상 딥러닝 기술의 발전에 기여되기를 바란다.

주제어 : 딥러닝, 영상 처리, 실시간 객체 인식, 흡연 감지, YOLO

목 차

국문초록 ····· i
목 차 ····· ii
표 목 차 ····· v
그림목차 ····· vi

제 1 장 서 론 ····· 1
1.1 연구목적과 연구방법론 ····· 1
1.2 딥러닝 기반 흡연 감지 관련 연구동향 ····· 2
1.3 논문의 구성 ····· 6
제 2 장 배경 이론 연구 ····· 7
2.1 객체 감지 방식 분류 ····· 7
2.2 FPN(Feature Pyramid Networks) ····· 8
2.3 경로 집계 네트워크(PANet, Path Aggregation Network) ····· 9
2.4 앵커 박스와 앵커 프리 방식 ····· 11
2.4.1 앵커 박스 ····· 11
2.4.2 앵커 프리(Anchor-Free) 방식 ····· 12
2.5 병목(BottleNeck) 구조 ····· 15
2.6 공간 피라미드 풀링(Spatial Pyramid Pooling) ····· 16
2.7 CSPNet(Cross Stage Partial Network) ····· 18
2.8 데이터 증강 기법 (Data Augmentation) ····· 18
제 3 장 YOLO ····· 20
3.1 YOLO 처리 방식 ····· 20
3.2 YOLOv1 구조 ····· 22
3.3 손실함수 ····· 23

3.4 YOLO 역사 ····· 24
3.4.1 YOLOv1 ····· 24
3.4.2 YOLOv2 ····· 24
3.4.3 YOLOv3 ····· 26
3.4.4 YOLOv4 ····· 27
3.4.5 YOLOv5 ····· 28
3.4.6 YOLOv6(MT-YOLO) ····· 29
3.4.7 YOLOv7 ····· 31
3.4.8 YOLOv8 ····· 33
3.4.9 YOLOv9 ····· 37
3.4.10 YOLOv10 ····· 38
3.5 YOLO 버전들의 요약 ····· 40
제 4 장 YOLO 모델 성능 비교 ····· 44
4.1 연구 방법 ····· 44
4.1.1 연구 실험 환경 ····· 44
4.1.2 데이터 세트 ····· 45
4.2 학습 과정 ····· 46
4.3 실험 결과 ····· 50
4.3.1 실험 평가 도구 ····· 50
4.3.2 실험 진행 방법 ····· 52
4.3.3 실험 진행 결과표 ····· 53
4.3.4 실험 결과물 ····· 55
4.4 실험 분석 ····· 59
4.4.1 YOLO 버전들간의 mAP 성능 비교 ····· 59
4.4.2 YOLO 버전들간의 mAP 파라미터 수 비교 ····· 60
4.4.3 YOLO 버전들간의 추론 시간 비교 ····· 61
4.4.4 전이 학습과 스크래치 학습과의 비교 ····· 62

4.4.5 배치사이즈 변화에 따른 성능지표 변화 ····· 63
4.4.6 데이터 증강 옵션에 따른 성능지표 변화 ····· 64
제 5 장 결 론 ····· 66
5.1 종합 의견 ····· 66
5.2 성과, 한계 및 향후 연구 과제 ····· 67

표 목차

표 3.1 YOLO10-S/M 모델에서 효율성 개선 ····· 39
표 3.2 YOLO 버전들 특징 요약 ····· 40
표 4.1 개발 H/W 및 S/W 환경 ····· 44
표 4.2 YOLO 모델 크기 설정값 ····· 46
표 4.3 YOLO 버전 레이어, 파라미터, 그래디언트, GFLOPs ····· 47
표 4.4 YOLO 버전들의 하이퍼파라미터(Hyperparameters) 설정값 ····· 48
표 4.5 학습시 사용 소스 코드 위치 ····· 53
표 4.6 YOLOv5 실험 결과 ····· 53
표 4.7 YOLOv6 실험 결과 ····· 54
표 4.8 YOLOv7 실험 결과 ····· 54
표 4.9 YOLOv8 실험 결과 ····· 54
표 4.10 YOLOv9 실험 결과 ····· 55
표 4.11 YOLOv10 실험 결과 ····· 55
표 4.12 비교군 모델간의 파라미터수,mAP@50 점수 ····· 57
표 4.13 mAP 성능 대비 파라미터 수 비교 ····· 60

그림 목차

그림 2.1 물체 감지 방법에 따른 딥러닝 역사 ····· 7
그림 2.2 FPN 방식 과 타방식의 비교 ····· 8
그림 2.3 경로 집계 네트워크(PANet) 구조 ····· 9
그림 2.4 상향식 경로 집계 네트워크 예시 ····· 10
그림 2.5 박스 브랜치를 가진 적응형 특징 풀링 과정 ····· 10
그림 2.6 완전 연결 융합(fusion)을 가진 마스크 예측 분기(branch) ····· 10
그림 2.7 슬라이딩 윈도우 방식 ····· 11
그림 2.8 앵커 작동 방식 ····· 12
그림 2.9 FCOS 전체적인 개념도 ····· 14
그림 2.10 Centerness 개념 ····· 15
그림 2.11 병목구조 개념 ····· 15
그림 2.12 SPP 네트워크 구조와 전통적인 CNN 구조비교 ····· 17
그림 2.13 공간 피라미드 풀링 층 구조를 가진 네트워크 ····· 17
그림 2.14 DenseNet에 적용한 Cross Stage Partial Network ····· 18
그림 3.1 YOLO 감지 시스템 ····· 20
그림 3.2 YOLO 처리방식 ····· 21
그림 3.3 YOLOv1 감지 방식 ····· 22
그림 3.4 YOLOv1 구조 ····· 22
그림 3.5 DARK-19 구조도 ····· 25
그림 3.6 YOLO1과 비교 성능향상 원인 ····· 25
그림 3.7 DarkNet-53 구조도 ····· 26
그림 3.8 Residual Block 구조 ····· 26
그림 3.9 YOLOV3 다중 스케일 감지 구조 ····· 27
그림 3.10 YOLOv5 구조 ····· 29
그림 3.11 C3 구조도 ····· 29
그림 3.12 YOLOv6 모델 구조 ····· 30

그림 3.13 YOLOv6 v3.0의 목(Neck) ····· 31
그림 3.14 Extended Efficient Layer Aggregation 구조 ····· 32
그림 3.15 YOLO7 네트워크 구조 ····· 33
그림 3.16 C3(YOLOv5) 구조와 C2f(YOLOv8) 모듈 구조 비교 ····· 34
그림 3.17 YOLOv8-P5 네트워크구조 ····· 36
그림 3.18 PGI 와 관련 네트워크 구조, 방법 ····· 37
그림 3.19 GELAN 구조와 CSPNet, ELAN 비교 ····· 38
그림 3.20 다양한 네트워크에서 시각화 초기 가중치 결과 ····· 38
그림 3.21 YOLOv10 일관된 이중 할당 ····· 40
그림 4.1 개발환경 GPU 드라이버 및 CUDA 버전 ····· 44
그림 4.2 흡연 데이터 세트 구성 ····· 45
그림 4.3 흡연 데이터 세트의 이미지들 ····· 45
그림 4.4 학습 진행 과정 ····· 49
그림 4.5 학습 완료 결과 화면 ····· 50
그림 4.6 IoU 개념도 ····· 52
그림 4.7 YOLOv9 e모델(위)과 YOLOv10 n모델(아래) 검증 화면 ····· 56
그림 4.8 파라미터수가 비슷한 모델간 mAP@50 비교 ····· 57
그림 4.9 Yolov9e(좌)과 Yolov10n(우) 혼동행렬비교 ····· 58
그림 4.10 YOLOv9e(좌)과 Yolov10n(우) PR 곡선 비교 ····· 58
그림 4.11 모델간 mAP@50 성능 비교 ····· 59
그림 4.12 YOLO 모델 간 파라미터수 비교 ····· 61
그림 4.13 YOLO 추론 시간 비교 ····· 62
그림 4.14 전이 학습시 AP 증가율 ····· 63
그림 4.15 YOLOv8의 n모델에서 배치사이별 mAP@50 변화 ····· 64
그림 4.16 YOLOv8의 n모델에서 mosaic변화에 따른 mAP@50 변화 ····· 65

제 1 장 서 론

1.1 연구목적과 연구방법론

간접흡연으로 인한 연기에는 많은 유해 물질이 함유되어 있다. 이러한 유해 물질에 지속적으로 노출되면 비흡연자도 폐암, 후두암과 같은 질병 등이 발생할 수 있다. 이런 우려로 인해 사람들이 많이 이용하는 대도심의 지하철역, 버스 정류장 주변에는 길거리 야외임에도 불구하고 금연 구역으로 설정하고 있다. 그러나 지방자치 단체에서 거리나 대중교통 승하차 구역을 금연 구역을 지정하여 표지판을 설치하고 단속 시 과태료를 부과하고 있지만 단속 인력의 한계와 흡연자의 비도덕으로 인해 제대로 지켜지지 않는 경우가 많다.

금연 구역에서 흡연자를 카메라 영상을 기반으로 자동 감지하여 경고 메시지를 주거나 단속 관리자에게 알림 통보를 한다면 효율적인 관리를 할 수 있고 비흡연자의 간접흡연을 막는 효과를 얻을 수 있다. 또한 화재, 폭발의 위험을 안고 있는 가스, 정유시설, 군부대의 탄약고 등 각종 위험 시설에 대해 영상 인공지능 기술을 통해 고의 또는 실수로 할 수 있는 흡연 행위를 탐지하고 시설 관리를 한다면 불미스러운 사고를 미연에 예방하는 데 효과적일 것이다.

또한 많은 국가에서 청소년에 흡연 장면에 노출되는 것을 막기 위해 영화, TV에서 흡연 장면을 규제하고 있고 흡연 도구나 흡연 장면을 가려지도록 처리하고 있다, 본 연구는 유튜브, 각종 온라인 소셜 매체에 영상이나 이미지 속 흡연 장면을 사전에 검증하여 업로드를 못 하게 하거나 특정 부분을 마스킹하는 기술적인 도구로도 활용할 수 있다.

최근 딥러닝을 통한 물체 감지 기술이 활발히 연구되고 있다. 특히 YOLO 계열의 딥러닝 모델이 산업 분야에서 인기를 얻고 있는데 감지(Detection) 속도가 빨라서 실시간을 감지하는 유리하다는 장점이 있다. 흡연 장면은 실시간으로 파악해야 의미가 있기에 감지 속도가 빠른 YOLO 모델을 적용하는 연구를 진행하는 것이 적절하다고 판단되었다. YOLO는 2016년 조셉 레드몬(Joseph Redmon)의 최초 버전 발표

이후 여러 사람에 의해 다양하게 수정되고, 개선되면서 다양한 버전들이 지속적으로 공개되고 있다.

본 연구에서는 그중에 대중들에게 많이 언급되어 시리즈 형태로 붙은 최근 연구를 중심으로 버전 5부터 10까지 YOLO 기반 딥러닝 모델들을 흡연데이터 세트에 적용하여 흡연 탐지 성능을 비교하였다. 성능 지표에 근거하여 우수성을 판단하고 환경 조건에 따라 어떤 모델이 상황에 따라 적합한지에 대한 연구를 진행하였다.

1.2 딥러닝 기반 흡연 감지 관련 연구동향

Zhang et al.은 2018년 논문 “Smoking Image Detection Based on Convolutional Neural Networks”[1]에서 흡연 이미지 분류를 위한 Smoking Net 모델을 제안하였다. 이 모델은 GoogLeNet을 기반으로 하여 흡연 이미지의 특징을 효과적으로 최적화한 CNN 모델이다. 전통적인 기계학습 모델(HOG+SVM, LBP+Adaboost 등)과 비교했을 때, 제안된 모델은 더 높은 정확도와 성능을 보였으며, AlexNet, GoogLeNet과 같은 기존 CNN 기반 모델보다도 우수한 성능을 나타낸 것으로 보고되었다.

Jonel R. Macalisang 외 연구진은 “Eye-Smoker: A Machine Vision-Based Nose Inference System of Cigarette Smoking Detection using Convolutional Neural Network”[2]에서 기계 시각 기술을 이용한 흡연 감지 시스템을 제안하였다. 이 연구는 특정 구역 내에서 흡연자를 감지하기 위한 목적으로 데이터 세트를 구축하고, 딥러닝 알고리즘인 YOLOv3 모델을 학습하여 검증하였다. 연구 결과, 제안된 시스템은 흡연자 탐지에 있어 효과적인 성능을 발휘하는 것으로 보고되었다.

J. Tang 등은 2021년 논문 “Smoking Behavior Detection Based On Improved YOLOv5s Algorithm”[3]에서 YOLOv5의s 알고리즘과 이미지 처리를 결합한 흡연 행위 탐지 방법을 제안했었다. 담배와 같은 작은 대상을 감지하는 것이 목표였으며, K-means 알고리즘과 작은 대상 탐지 계층을 추가하여 YOLOv5의 s 알고리즘을 개선함으

로써 탐지 정확도를 크게 향상시켰다고 발표했다.

Zhen Zhang 외 연구진은 “Research on Smoking Detection Based on Deep Learning”[4]에서 맞춤형 주의 메커니즘(attention mechanism) 모듈과 개선된 잔여 네트워크(residual network)를 기반으로 각 단계의 특징을 융합한 백본(backbone) 네트워크를 설계하였다. 연구진은 자체 제작한 흡연 데이터 세트를 사용하여 탐지 실험을 수행하였고, 그 결과 제안된 모델이 Faster RCNN, SSD, YOLOv5 등 기존 모델들보다 더 높은 정확도를 기록하였다. 해당 연구에서 제안된 모델의 평균 정확도(mean Average Precision)는 86.32%에 도달하였으며, 탐지 속도는 55프레임/초(f/s)를 달성하였다.

Y. Ma, J. Yang, Z. Li, Z. Ma는 “YOLO-Cigarette: An effective YOLO Network for outdoor smoking Real-time Object Detection”[5]연구에서 기존 YOLOv5 네트워크 구조를 개선하기 위한 방법으로 세분화된 공간 피라미드 풀링 모듈(FSPP)과 다중 공간 주의 메커니즘(MSAM)을 융합한 YOLO-Cigarette 모델을 제안하였다. 본 모델은 작은 표적 검출의 정확도를 효과적으로 향상시켰으며, 모델 파라미터를 정량화하여 계산 복잡도를 줄임으로써 탐지 속도 또한 향상되었다. 실험 결과, YOLO-Cigarette 모델은 기존 모델보다 4.8% 더 높은 성능을 보였으며, 테스트 세트에서 95%의 정확도를 달성하였다.

J. Li는 “Study on Smoking and Telephone Behaviour Detection Based on Convolutional Neural Network”[6] 논문에서 공공장소에서 흡연과 전화 사용과 같은 행위를 규제할 필요가 있는 상황을 염두에 두고, 이러한 행동을 자동으로 감지할 수 있는 데이터 세트를 구축하고 CNN(합성곱 신경망) 모델 구성을 제안하였다 해당 연구는 이미지 분류에 초점을 맞춘 것으로, 물체의 위치를 파악하는 물체 감지 모델이 아닌, 이미지를 분류하는 모델을 기반으로 진행되었다. 이를 통해 흡연 및 전화 사용과 같은 특정 행동을 자동으로 식별하는 시스템의 개발을 목표로 하였다.

C. Wang, T. Zheng, F. Sun, H. Liu는 “A Smoking Detection Algorithm Based on Improved YOLOV5”[7]논문에서 흡연이 금지된 장소에서 흡연 행위를 신속하게 감지

할 수 있는 YOLOv5 기반 알고리즘을 제안하였다. 제안된 탐지 알고리즘은 특징 추출 네트워크를 개선하여 흡연을 보다 효과적으로 탐지할 수 있도록 설계되었다. 특히, YOLOv5의 특징 추출 네트워크에서 C3 모듈을 CoT 모듈로 대체함으로써 모델의 탐지 성능이 크게 향상되었다. 또한, SPPF 구조 앞에 CBAM 주의 메커니즘을 삽입하여 목표 특징을 더욱 효과적으로 추출할 수 있는 능력이 강화되었다. 실험적 비교 결과, 개선된 알고리즘은 기존 모델에 비해 탐지 성능과 실시간 처리 능력에서 유의미한 향상을 보였다.

J. Peng, C. Wang, Y. Li, H. Chen은 “Substation Personnel Smoking Detection Based On GhostNetV2-YOLOv5”[8] 논문에서 변전소 인원의 흡연 행위를 실시간으로 감지하기 위한 GhostNetV2-YOLOv5 기반 알고리즘을 제안하였다. 그 알고리즘은 GhostNetV2를 백본 네트워크로 활용하여 모델의 정확도를 향상시키는 동시에 실시간 탐지가 가능하도록 설계되었다. 실험 결과, 제안된 알고리즘은 기존 YOLOv5 알고리즘과 비교했을 때 총 mAP가 2.58% 향상되었으며, 예측 속도는 1.61배 향상되었다. 이러한 결과는 흡연 감지 네트워크 중에서도 우수한 성능을 보여주며, 실제 적용에 높은 가치를 지닌다.

Aditya 외 연구진은 “Smoking Detection using Deep Learning”[9] 논문에 담배 이미지에서 학습된 딥러닝 알고리즘인 YOLOv5를 사용하여 정확하고 효율적인 흡연 감지 결과를 도출하였다. 감지 모듈은 이미지가 흡연자를 묘사하는 경우에만 선택적으로 활성화되어, 거짓 양성(false positive)을 줄임으로써 성능을 크게 향상시켰다. 또한, 운전자의 흡연 습관이 운전 안전에 미치는 영향을 고려하여, FPN과 확장 합성곱(convolution) 기법을 활용한 운전자의 흡연 행위 감지 방법을 탐구하였다. 제안된 방법은 운전자의 이미지에서 작은 물체를 정확하게 식별하여 흡연 행동을 감지 및 식별할 수 있음을 시뮬레이션 실험을 통해 입증하였다. 실험 결과, 정확도는 94.75%, 재현율은 96%, 정밀도는 95.05%, AUC는 95.5%를 기록하였으며, 7,000개의 이미지로 구성된 데이터 세트에서 테스트한 결과 96.74%의 분류 정확도를 달성하였

다. 또한, 연기의 주요 특징인 투과성 개념을 도입하여 연기를 감지하고 그 두께 분포를 분석하는 데 사용하였다. 이 기술의 효과는 정량적 및 정성적 평가를 통해 검증되었다.

Robert P. Lakatos 외 연구진은 “A multimodal deep learning architecture for smoking detection with a small data approach”[10] 논문에서, 흡연 관련 콘텐츠를 감지하기 위해 사전 학습된 이미지 및 언어 모델을 활용한 멀티모달 아키텍처를 제안하였다. 본 연구에서 제안된 멀티모달 접근법은 담배와 관련된 미디어 콘텐츠에서 은밀한 광고를 탐지하는 데 있어 높은 잠재력을 지닌다. 이미지 처리 네트워크와 언어 모델을 멀티모달 아키텍처로 결합함으로써, 텍스트와 이미지 데이터를 동시에 활용할 수 있으며, 전문가의 입력을 통해 성능을 더욱 향상시킬 수 있는 통합 솔루션을 제시한다.

김동준 외 연구진은 “딥러닝 기술을 이용한 영상에서 흡연 행위 검출(Detection of Smoking Behavior in Images Using Deep Learning Technology)”[11] 연구 논문에서 실시간 객체 탐지 기술인 YOLOv8과 자세 추정 모델인 오픈 포즈(OpenPose)를 결합하여 흡연자의 행동을 감지하는 방법을 제안하였다. 해당 연구에서는 영상 내에서 흡연자와 흡연 행위가 동시에 검출될 경우 이를 흡연 행위로 판정하는 방식을 적용함으로써 모델의 정확성을 높이고자 하였다. 연구 결과, 제안된 모델은 영상에서 흡연 행위를 보다 효과적으로 탐지할 수 있음을 입증하였으며, 실시간 탐지 기술의 적용 가능성을 확인하였다.

Wang, Z., Liu, Y., Lei, L. 등은 “Smoking-YOLOv8: a novel smoking detection algorithm for chemical plant personnel”[12]연구에서 YOLOv8 알고리즘을 개선하여 화학 공장 내에서 흡연 행위를 감지하는 방법을 제안하였다. 연구진은 기존 YOLOv8 모델의 성능을 향상시키기 위해 손실 함수를 WIoUv3로 대체하고, Self-Distillation(SD) 어텐션 메커니즘을 구현하였다. 또한, 컨볼루션 스트라이드 및 목(Neck) 구조를 변경함으로써 85.88%의 정확도를 달성하였으며, 이는 기존 YOLOv8 대비 약 6.18% 향상된 결과이다. 이러한 결과는 화학공장에서의 실시간 흡연 감지 시스템에 적용

가능성을 제시한다.

1.3 논문의 구성

본 논문의 구성은 총 5장으로 구성되어 있으며, 각 장의 내용은 다음과 같다. 1장에서는 연구의 배경, 목적, 및 관련 연구 동향에 관해 서술하였다. 2장에서는 본 연구에 활용된 딥러닝에 대한 배경 이론을 설명하였다. 3장에서는 실시간 객체 탐지 알고리즘 중 하나인 YOLO(You Only Look Once)의 이론과 버전별 특징에 대해 언급하였다. 4장에서는 YOLO 최신 버전 중심으로 버전 5부터 10까지 버전별 학습 과정과 실험 결과를 제시하고, 모델 간의 성능 비교를 통해 상황에 따른 최적의 모델을 제안하고 입증하였다. 마지막으로 5장에서는 본 연구의 결론을 도출하고, 연구의 한계점과 함께 연구 방향에 대해 논의하였다.

제 2 장 배경 이론 연구

2.1 객체 감지 방식 분류

딥러닝에서 물체 인식은 크게 두 가지로 나눌 수 있다. 이미지 내 물체의 카테고리를 예측하는 이미지 분류(Image Classification)와 물체를 식별하고 그 위치까지 파악하는 객체 감지(Object Detection)이다. 객체 감지는 다시 단일 단계(One-Stage) 계열과 이중 단계(Two-Stage) 계열로 구분할 수 있다.

2단계 감지(Two-Stage Detector) 방식은 물체의 위치를 찾는 문제인 위치 추정(Localization)과 물체의 종류를 찾는 분류(Classification) 문제를 분리하여 순차적으로 처리하는 방식이다. 이 방식의 대표적인 알고리즘으로는 R-CNN, Fast R-CNN, 및 Faster R-CNN이 있다.

반면, YOLO(You Only Look Once)는 이미지 내 존재하는 객체 분류와 그 객체 위치를 한 번의 회귀 처리로 예측하는 단일 단계 감지(One-Stage Detector) 방식이다. 이 방식은 이미지를 한 번만 보고 객체를 예측한다는 점에서 실시간 성능이 뛰어나다. 그림 2.1은 객체 감지 역사를 도식화한 그림이다.

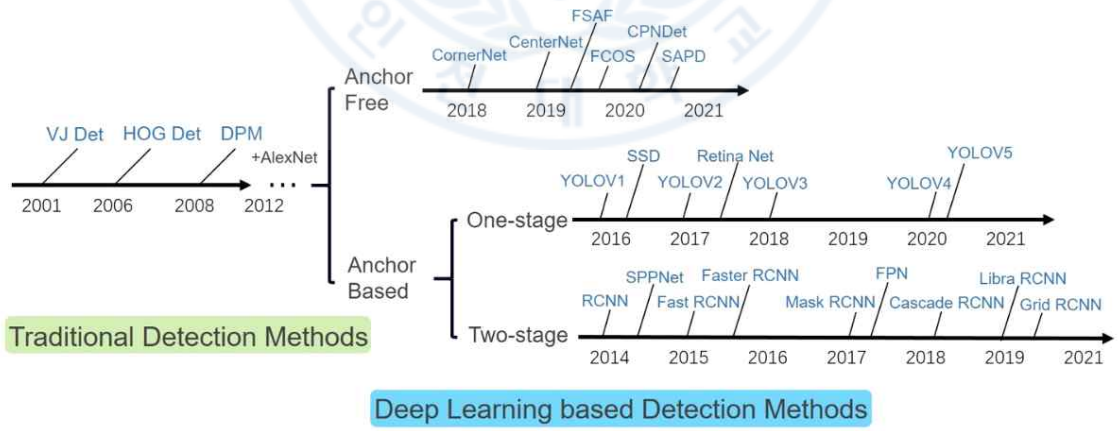

그림 2.1 물체 감지 방법에 따른 딥러닝 역사[13]

2.2 FPN(Feature Pyramid Networks)

FPN(Feature Pyramid Networks)은 다양한 크기의 객체를 효과적으로 탐지하여 모델 성능을 향상하기 위해 다중 스케일 특징맵을 활용하는 네트워크이다. FPN은 자체적으로 객체를 탐지하는 감지기가 아니며, Faster R-CNN, Mask R-CNN, YOLOv3 등 다양한 객체 검출 모델과 함께 작동하는 특징 추출기로 사용된다.

FPN의 "피라미드(Pyramid)"는 서로 다른 해상도의 특징맵을 층층이 쌓아 올린 구조를 의미한다. 기존에는 이미지 크기를 단순히 축소(resize)하여 학습하는 방식이 주로 사용되었으나, 이를 개선하기 위해 FPN이 제안되었다. FPN은 임의 크기의 단일 스케일(single-scale) 이미지로부터 특정 레이어에서 특징맵을 추출하여 피라미드를 형성하는데, 이 과정은 상향식 패스웨이(bottom-up pathway), 하향식 패스웨이(top-down pathway), 그리고 측방 연결(lateral connections) 방식을 통해 이루어진다. 이를 통해 FPN은 다양한 크기의 객체를 더 효과적으로 탐지할 수 있다.

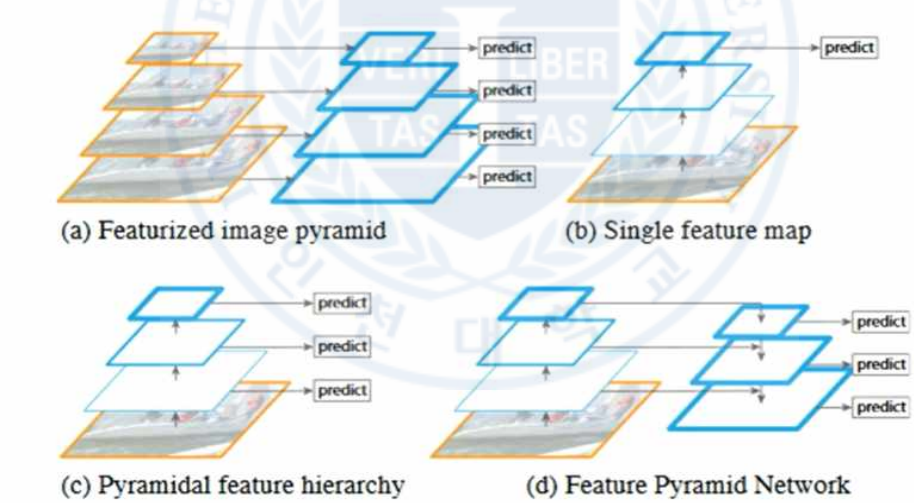

그림 2.2 FPN 방식 과 타방식의 비교[14]

상향식 패스웨이(Bottom-up pathway)는 입력 이미지로부터 2배씩 작아지는 특징맵을 추출하는 과정이다. 이 과정에서는 이미지의 해상도를 점차 축소시키며, 다양한 크기의 객체를 감지할 수 있는 정보가 포함된 다중 스케일의 특징맵을 생성한다. 반면, 하향식 패스웨이(top-down pathway)는 1×1 컨볼루션(convolution) 연산을 적용하여 채

널 수를 맞춘 후, 2배로 업샘플링(upsampling)하는 과정을 거친다. 이를 통해 상위 레벨의 추상적인 특징을 하위 레벨로 전파하여 보다 세밀한 정보를 보완한다.

마지막으로, 측방연결(lateral connections)는 업샘플링된 특징맵과 바로 아래 레벨의 특징맵을 원소별 덧셈(element-wise addition) 연산을 통해 결합하는 과정이다. 이 과정을 통해 각각의 스케일에서 풍부한 정보가 포함된 피라미드 특징맵(P2, P3, P4, P5, P6)을 생성할 수 있다. 이러한 구조는 다양한 크기의 객체를 효과적으로 탐지하는 데 기여한다.

2.3 경로 집계 네트워크(PANet, Path Aggregation Network)

2018년 9월에 경로 집계 네트워크(Path Aggregation Network)는 COCO 2017 Challenge Instance Segmentation 대회 1위와 대규모 배치 학습 없이 물체 감지 과제에서 2위를 차지한 모델로 Mask R-CNN 기반으로 인스턴스 세그먼테이션(Instance Segmentation)을 위한 모델이다[15]. FPN 백본에다 하향식 경로 집계(top-down pathway)에 추가적으로 상향식 경로 집계(bottom-up pathway)를 추가하여 하위 레벨의 위치 정보를 상위 레벨로 전달한다. 상향식 경로 집계와 적응적 특징 풀링(Adaptive feature pooling)은 YOLO v4에서도 사용된 기술이다. PANet 특징은 상향식 경로 집계, 적응적 특징 풀링, 완전 연결 혼합(Fully-connected fusion)이다.

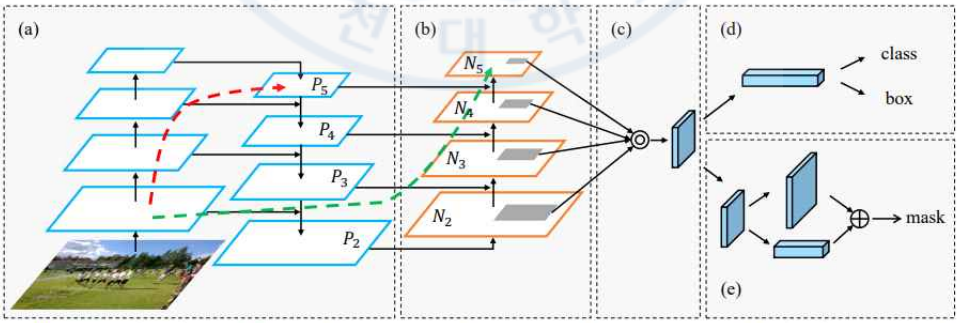

Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path augmentation. (c) Adaptive feature pooling. (d) Box branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature maps in (a) and (b) for brevity.

그림 2.3 경로 집계 네트워크(PANet) 구조[15]

상향식 경로 집계는 하위 레벨의 특징 정보를 상위 레벨로 전하는 것으로 기존 FPN방식이 100 레이어(Layer)이상 거쳐서 상위 레벨로 전달되는 반면 상향식 경로 집계 방식은 하위정보가 상위로 전달되는데 10 레이어(layer)만 사용되기에 정보 흐름을 단축해서 큰 객체 검출 시에도 하위정보를 활용할 수 있다.

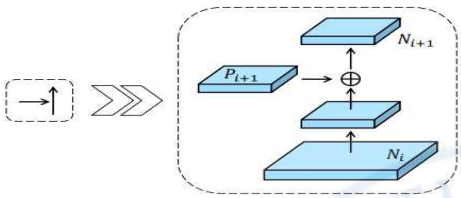

그림 2.4 상향식 경로 집계 네트워크 예시[15]

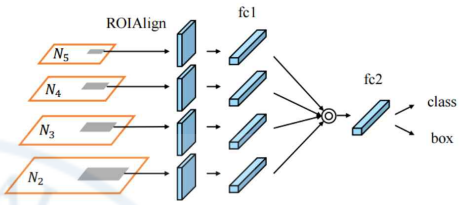

그림 2.5 박스 브랜치를 가진적응형 특징 풀링 과정[16]

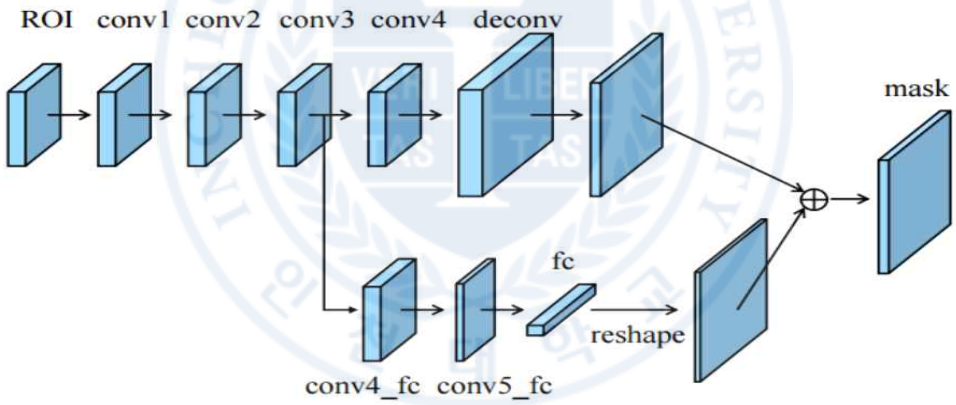

그림 2.6 완전 연결 융합(fusion)을 가진 마스크 예측 분기(branch)[15]

적응형 특징 풀링은 그림 2.5과 같이 N2~N5 각각의 특징 맵에 RPN(Region Proposal Network)를 적용하여 ROI(Region of Interest)를 생성하고 ROI 정렬(ROI Align)를 거친 후 일정한 크기의 벡터를 최대(max)연산으로 하나로 결합한다. 이 결합된 벡터에서 클래스와 박스를 예측한다.

완전 합성곱 신경망(Fully Connection Network)은 기존 AlextNet, VGGNet 등 이미

지 분류모델이 합성곱층(Convolution layer)와 완전연결층(Fully connected layer)로 구성되어 있는데 여기 완전연결층(FC Layer)를 제거하고 1×1 합성곱층(Convolution layer)를 추가하여 네트워크 구조를 모두 합성곱 층으로만 구성된 네트워크이다. 완전연결 결합(Fully-connected Fusion)은 완전 합성곱 신경망(FCN)과 완전 연결 계층(FC)를 그림 2.6과 같이 함께 사용하여 정보의 다양성 향상되고 좀 더 나은 마스크(mask)를 생성하게 된다.

2.4 앵커 박스와 앵커 프리 방식

2.4.1 앵커 박스

앵커 박스(Anchor Box)이란 앵커 상자는 이미지 내에서 객체가 있을 법한 특정 높이와 너비의 미리 정의된 경계 상자들의 모음이다. 앵커 박스를 사용하면 딥 러닝 신경망 프레임워크의 감지 부분에 대한 속도와 효율성이 향상된다.

초기 딥러닝에서는 슬라이딩 윈도우 방식을 사용하였는데 이것은 윈도우 왼쪽 상단에서부터 오른쪽으로 이동시키면서 물체를 감지하는 방식이다. 이것은 모든 영역에 특징을 추출하기 위해 이미지 검출 계산을 하는 스캔하는 방식이라 수행시간이 오래 걸리는 초기 물체 감지 방식이다.

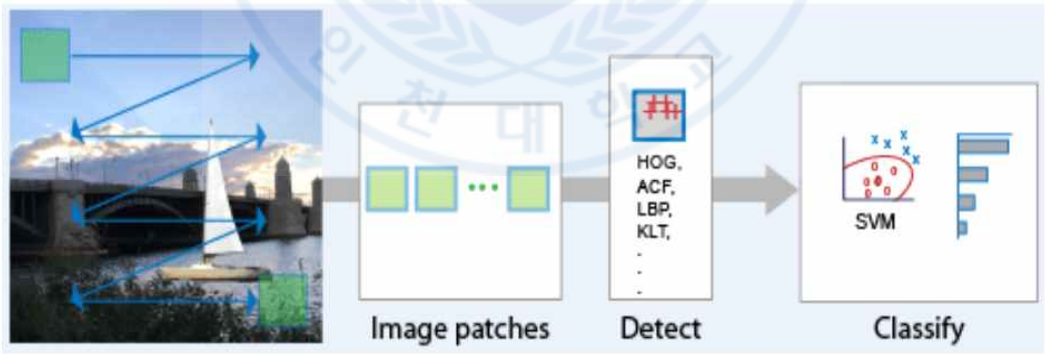

그림 2.7 슬라이딩 윈도우 방식[17]

앵커 박스를 사용하면 이미지에서 특징을 추출하기 위한 슬라이딩 윈도우 접근 방식을 대체하고 비용을 크게 줄일 수 있다, Faster R-CNN, RetinaNet, YOLOv2과 YOLOv3 등 다양한 모델에서 앵커 박스가 사용되었다. 앵커를 미리 정해 놓고 그로

부터 회귀(Regression)를 하면 보다 빠르고 정확히 학습하면서 물체를 감지할 수 있다.

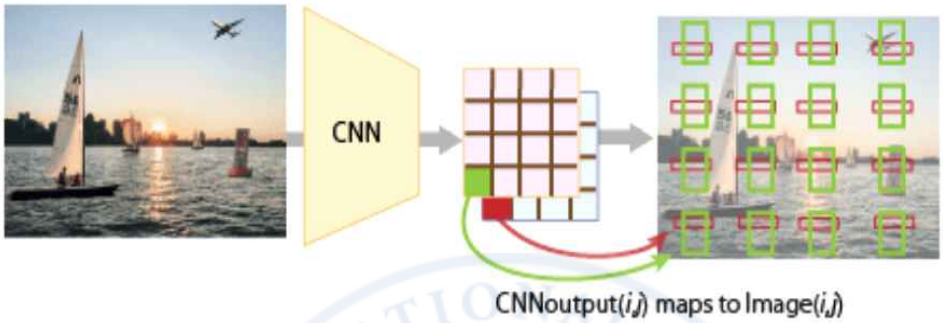

그림 2.8 앵커 작동 방식[17]

앵커 박스의 위치는 네트워크 출력의 위치를 입력 이미지에 다시 대응하여 결정된다. 이 과정은 모든 네트워크 출력에 대해 반복된다. 그 결과는 전체 이미지에 걸쳐 격자 형태로 배치된 앵커 상자 세트가 생성된다. 각 앵커 상자는 클래스의 특정 예측을 나타내며 예측 결과에 따라 객체 존재 여부와 클래스 정보를 제공한다. 예를 들어, 한 위치에서 두 개의 앵커 박스를 사용하는 경우, 각 앵커 박스는 해당 위치에서 서로 다른 객체 또는 클래스에 대한 예측을 수행할 수 있다. 다수의 예측 박스가 중복되거나 겹치는 경우, NMS(Non-Maximum Suppression) 알고리즘을 사용하여 신뢰도가 가장 높은 박스 하나만 남기고 나머지를 제거한다. 이를 통해 중복된 예측을 줄이고 최종 검출 결과를 정리한다.

또한, 앵커 박스에 대한 오프셋 예측과 클래스 예측은 손실 함수로 평가된다. 객체가 있는 위치에서는 앵커 박스의 위치 조정 및 클래스 예측의 정확도를 중점적으로 평가하며, 객체가 없는 위치에서는 불필요한 앵커 박스를 제거하는 데 중점을 둔다.

2.4.2 앵커 프리(Anchor-Free) 방식

전체 이미지를 차례로 스캔하지 않고 사전에 정의한 앵커를 바탕으로 예측하는

방식은 정확도와 속도향상에 큰 기여 했으나 비율이 정형화되어 있지 않는 형태를 가진 객체를 탐지할 경우에는 효용성이 없거나 오히려 학습에 방해가 되었다.

앵커 박스는 크기, 종횡비, 개수에 굉장히 민감하다는 단점이 있다. 이 앵커 박스를 어떻게 설계하느냐에 따라 모델 성능에 영향을 미치게 된다. 또한 사전 정의된 앵커(pre-defined anchor)는 모델 일반화 성능을 해치게 되고, 앵커 박스 크기와 다른 실제값(ground-truth)을 검출하기 어렵다는 단점이 있다.

앵커 프리방식은 객체 검출 시 미리 정의된 앵커 박스를 사용하지 않고, 객체의 중심을 직접 예측하거나 객체의 경계를 보다 자연스럽게 학습하는 방식이다. 앵커 박스를 사용하지 않는 대표적으로 FCOS[18] 모델이 있는데 그림 2.11과 같이 이 모델은 특징맵(feature map)의 모든 픽셀 위치에서 해당 픽셀이 특정 object에 속한 픽셀인지 아닌지를 계산하는 방식을 사용한다. 그런 후 해당 위치로부터 물체의 바운딩 박스의 경계까지 얼마큼의 거리인지를 왼쪽(Left), 위(Top), 오른쪽(Right), 아래(Bottom)로 표현하여 4D vector를 동시에 회귀(Regression)한다. FCNN(Fully connectioned Neural Network) 형태로 특징 맵(Feature Map)의 모든 픽셀마다 분류 점수(Classification Score)와 4D 벡터(Vector) [l, t, r, b]를 예측하는 것이다. 추가로 겹친 물체에 대하여 학습 및 추론을 하기 위해 FPN(Feature Pyramid Network)을 해결책으로 도입하고, 물체의 중심과 먼 거리에 있는 예측(Prediction)들이 낮은 성능을 가져 물체의 중심에 가까운 예측만을 활용하기 위해 “중심도 분기(Centerness Branch)”를 도입했다.

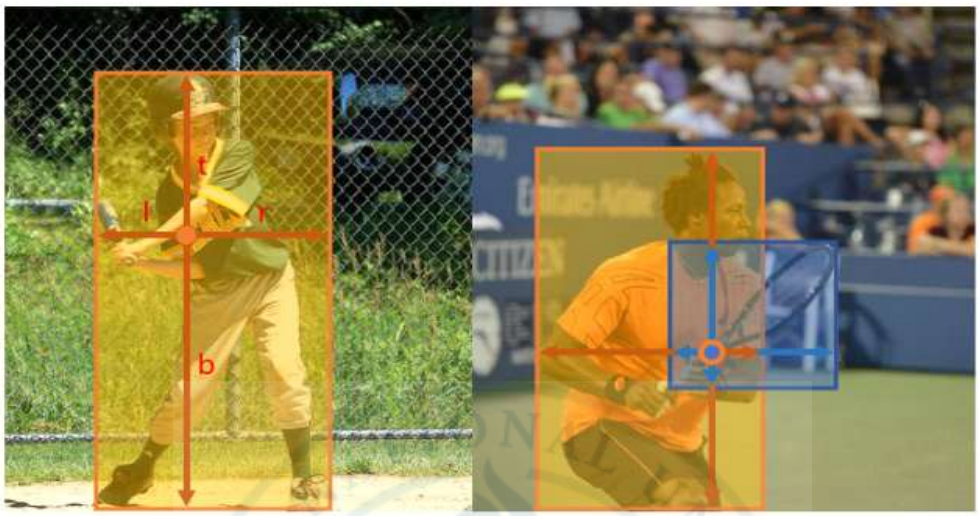

그림 2.9 FCOS 전체적인 개념도[18]

Centerness(중심도)는 그림 2.9처럼 빨간색, 파란색, 기타 색상으로 0~1 사이의 값을 나타내는데 특정 위치(location)에서 해당 위치(location)가 속한 물체(object)의 중심까지의 거리를 정규화된 값이다. 위치가 물체의 중심에서 벗어날수록 1에서 0으로 감소하며 테스트하는 동안 네트워크에서 예측한 중심성에 NMS의 분류 점수(classification score)를 곱하여 품질이 낮은 바운드 박스의 가중치를 낮출 수 있다[19]. 그래서 물체의 바운딩 박스(bounding box)를 결정할 때 물체중심(object center)에서 멀리 떨어진 위치에서 예측된 값들의 영향력을 줄인다.

다른 앵커 프리 방식인 코너넷(CornerNet)은 단일 단계 디텍터(one-stage detector)에서 일반적으로 사용되는 앵커 박스를 설계하지 않고 객체 경계 상자를 좌상단 코너와 우하단 코너의 쌍으로 표현하는 한 쌍의 키포인트로 감지한다[20]. 두 개의 히트맵을 사용하여 좌상단, 하단 코너를 예측하여 객체 바운딩 박스를 생성하여 2019년 당시에는 MS COCO에서 42.2%의 AP를 달성하며 기존의 일단계 디텍터를 능가하는 성능을 보였다. 앵커프리방식은 YOLOv6 ~ YOLOv10 등 최근 YOLO 버전 등에도 앵커 프리 방식이 자주 등장한다.

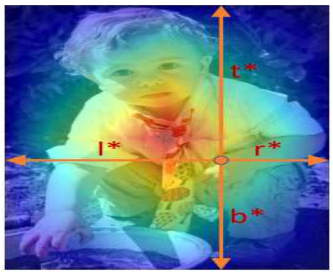

그림 2.10 Centerness 개념[19]

$$Centerness^{*} = \sqrt{\frac{\min(l^{*}, r^{*})}{\max(l^{*}, r^{*})} \times \frac{\min(t^{*}, b^{*})}{\max(t^{*}, b^{*})}}$$

수식 2.1 Centerness 공식[19]

2.5 병목(BottleNeck) 구조

병목(Bottleneck) 구조는 합성곱 신경망(Convolutional Neural Network, CNN)에서 연산 효율을 극대화하기 위해 특정 레이어에서 채널 수를 줄이고 다시 늘리는 방식으로 동작한다. 이 구조는 계산 자원을 절약하면서도 네트워크의 표현력을 유지할 수 있도록 설계되었다. 병목 블록(Bottleneck block)은 '잔차 연결(residual connection)'을 포함하는데, 이는 입력을 직접 출력과 더하는 방식으로, 네트워크가 깊어짐에 따라 발생할 수 있는 그래디언트 소실 문제를 완화한다. 이로 인해 학습이 효율적으로 이루어진다.

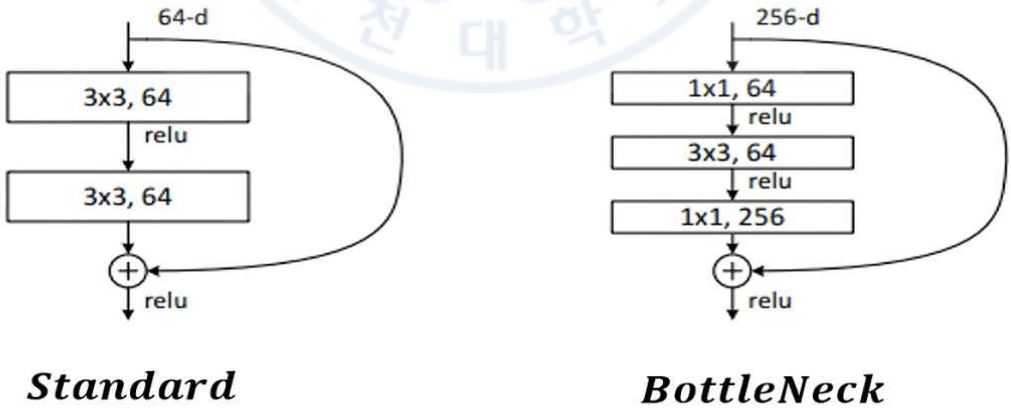

그림 2.11 병목구조 개념[21]

그림 2.11은 병목 구조를 시각적으로 표현한 것이다. 이 구조는 세 가지 주요 컴포넌트로 구성된다. 1x1 합성곱(Convolution)은 입력 채널 수를 축소하여 연산량을 크게 줄여 효율성을 높인다. 3x3 합성곱은 주요 특징 추출을 수행하고 이후 1x1 합성곱은 다시 채널 수를 늘리며, 입력과 출력을 연결하는 잔차 연결이 이뤄진다. 병목(Bottleneck) 구조는 네트워크의 깊이와 복잡성을 증가시키는 동시에 연산 비용을 줄이는 효과적인 방식으로 널리 사용된다.

2.6 공간 피라미드 풀링(Spatial Pyramid Pooling)

SPP[22]는 목(Neck) 일종으로 이미지 크기이나 비율에 상관없이 합성곱(Convolution layer) 의 마지막 특징 맵(feature map)을 고정된 크기의 그리드(Grid)로 분할한 후 평균화하여 고정된 크기의 표현값을(representation)을 생성하고 다음 계층으로 전달하는 방식이다.

R-CNN에서는 완전연결층(fully-connected layer)에서 고정된 크기의 특징맵(feature map)이 필요하기 때문에, 항상 고정된 크기의 이미지를 입력하기 위해 이미지를 왜곡(Wrapping) 또는 자르기(Cropping)하는 과정에서 이미지 변형이나 손실이 발생하여 성능 저하로 이어질 수 있다. R-CNN과 달리, SPP는 입력된 이미지의 크기를 변경하지 않고 그대로 입력받는 대신, 완전 연결층 직전의 특징맵이 달라지더라도, SPP 레이어(Layer)를 통해 고정된 수의 특징을 추출할 수 있다. 이를 통해 인위적인 이미지 변형에 의한 정보 손실을 방지할 수 있었다. 또한, R-CNN은 약 2,000개의 후보 영역에 대해 개별적으로 연산을 수행하기 때문에 속도가 느리다는 단점이 있었으나, SPP는 합성곱 연산을 공유하는 방법을 제안하여 이러한 속도 문제를 해결하였다[23].

그림 2.12의 맨 위 그림은 이미지를 고정된 사이즈로 맞추는 자르기(Cropping), 왜곡(Wrapping) 모습을 보여주고 있으며, 중간 그림은 전통적 CNN 구조, 맨 아래는 SPP 제안하는 구조이다.

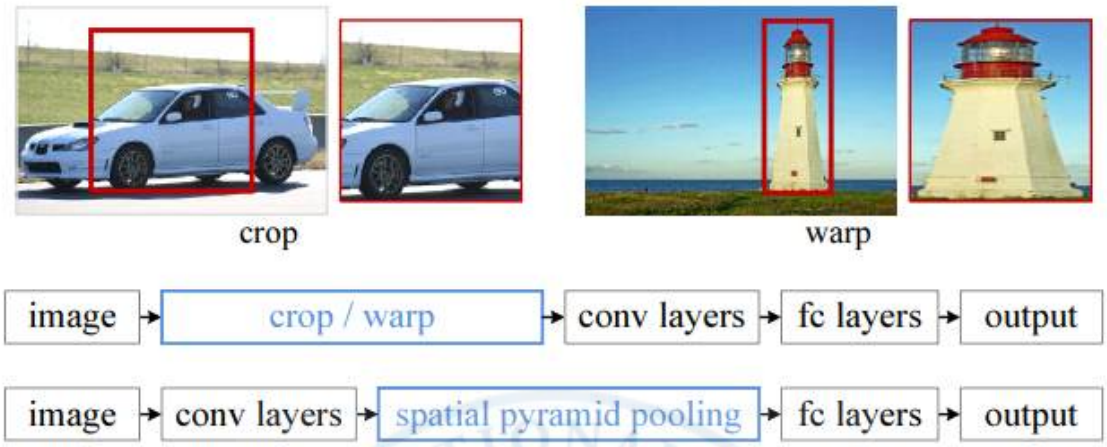

그림 2.12 SPP 네트워크 구조와 전통적인 CNN 구조비교[22]

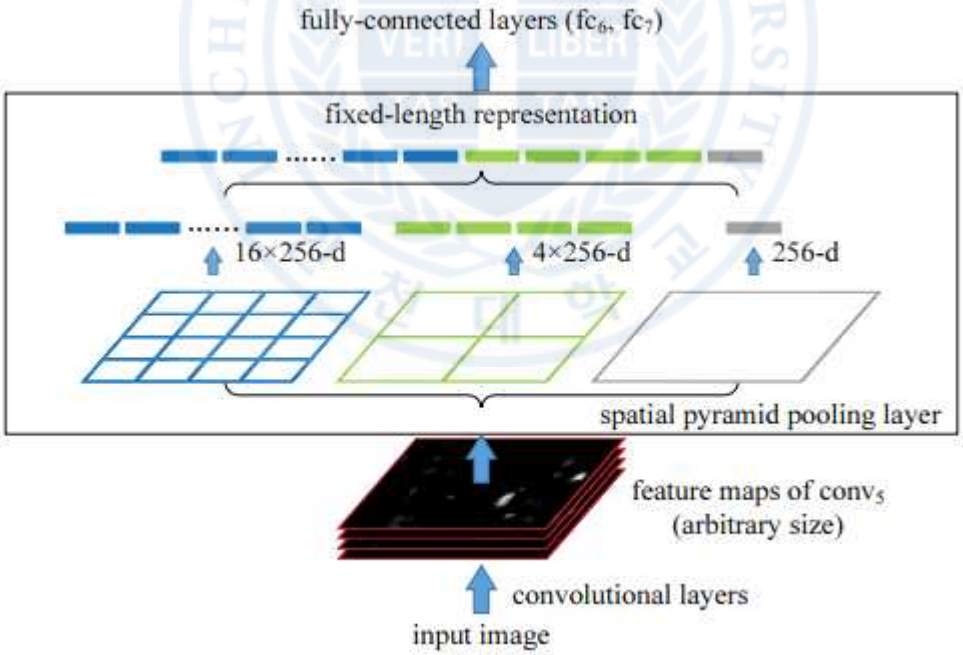

그림 2.13 공간 피라미드 풀링 층 구조를 가진 네트워크[22]

2.7 CSPNet(Cross Stage Partial Network)

CSPNet(Cross Stage Partial Network)은 2019년 11월에 발표된 대만학자들에 의해 제안된 딥러닝 모델 구조로 설계의 주요한 목적은 연산량을 줄이고 정확도를 높이는 것이다. 특징 맵을 두 부분으로 분리하여 한 부분은 고밀도 블록과 전환 레이어 등 여러 계층을 거쳐 전달하고, 나머지 한 부분은 전송된 특징 맵과 결합하여 다음 단계로 이동한다[24]. 특징 맵을 분할하여 처리하기 때문에 기존 CNN 아키텍처보다 계산량이 적고 그래디언트 흐름 개선과 중복 계산 감소를 통해 더 높은 정확도를 달성한다. 그림 2.14은 피처 맵을 두 개의 부분, 부분 밀집 블록 (partial dense block)과 부분 전이 레이어(partial transition layer)로 나누고 일부는 직접 다음 레이어로 전달하고 다른 일부는 여러 레이어를 통과한 후 다시 병합하는 것을 적용한 다이어그램이다.

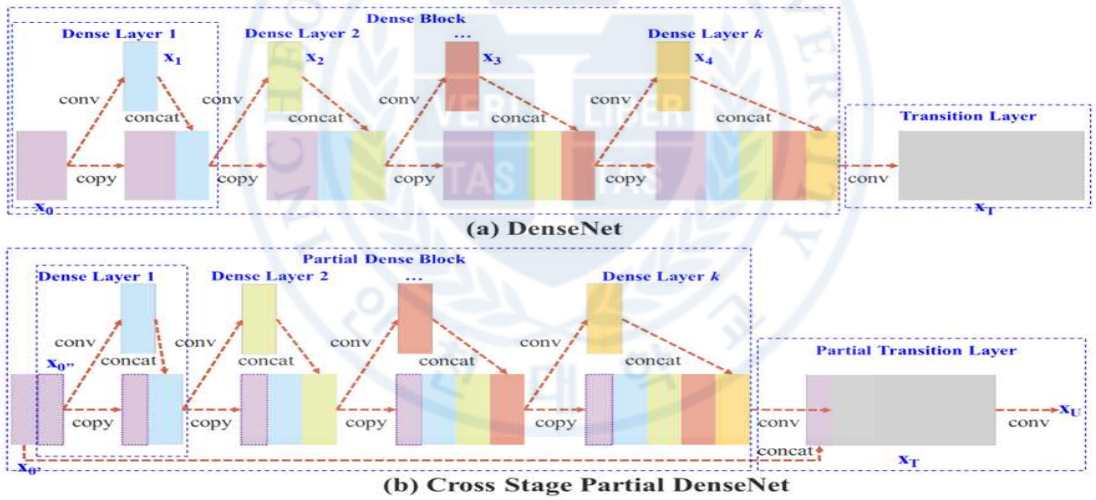

그림 2.14 DenseNet에 적용한 Cross Stage Partial Network [24]

2.8 데이터 증강 기법 (Data Augmentation)

딥러닝에서 데이터 증강은 기존 데이터 세트를 변형하여 새로운 데이터를 생성하는 기술로, 이를 통해 모델의 학습 데이터를 증가시키고 다양한 상황에 대한 일반화 능력을 향상함으로써 과적합(overfitting)을 방지하는 효과를 얻을 수 있다. 또

한, 데이터가 부족한 상황에서 더 많은 훈련 데이터를 생성할 수 있어, 보다 강력한 모델을 구축하는 데 기여한다.

기본적인 데이터 증강 기법으로는 이미지 회전(Rotation), 이동(Translation), 상하좌우 반전(Flipping), 확대/축소(Scaling), 자르기(Cropping), 무작위로 일부분을 잘라내는 랜덤 크롭(Random Cropping) 등이 있다. 색상 변환 기법으로는 밝기(Brightness), 대비(Contrast), 채도(Saturation), 색조(Hue)를 조절하거나, 컬러 지터링(Color Jittering)을 통해 이러한 요소들을 무작위로 조정할 수 있다. 또한, 노이즈 추가 기법으로는 가우시안 분포에 따른 노이즈를 이미지에 무작위로 추가하는 방법, 혹은 이미지에 흰색 또는 검은색 점을 무작위로 추가하는 소금-후추 노이즈(Salt-and-Pepper Noise) 등이 있다.

고급 데이터 증강 기법으로는 두 이미지를 픽셀 단위로 혼합해 새로운 이미지를 생성하는 믹스업(Mixup), 두 이미지의 일부분을 잘라 교환하여 새로운 이미지를 생성하는 컷믹스(CutMix), 이미지의 일부분을 검은색 사각형으로 가리는 컷아웃(Cutout), 무작위로 일부분을 지우는 랜덤 이레이징(Random Erasing) 등이 있다. 또한, 모자이크(Mosaic)는 여러 이미지를 결합해 하나의 이미지를 만드는 기법으로, YOLOv4에서 처음 소개되어 큰 효과를 보였으며 이후 YOLOv5 등 다양한 모델에서 성능을 극대화하는 데 사용되었다.

제 3 장 YOLO

3.1 YOLO 처리 방식

최초 YOLO 기본적인 처리 방식은 다음 그림 3.1과 같이 입력 이미지를 448 X 448 크기로 조정하고 합성곱 망(Convolution Network)으로 통과 처리한 후 NMS (비최대 억제,Non-maxium Suppression) 알고리즘을 이용하여 감지하는 방식이다.

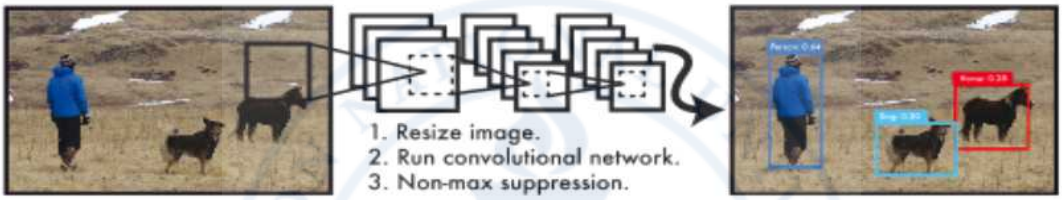

그림 3.1 YOLO 감지 시스템[25]

그림 3.2처럼 입력 이미지를 S x S 그리드 셀로 나눈 다음 각 그리드 셀에서 물체의 중앙에 가까운 셀이 객체를 탐지하는 역할을 맡게 된다. 바운딩 박스(Bounding Box)값은 중심 좌표인 X, Y, 가로(W), 세로(H) 크기 정보와 신뢰도(Confidence Score) 수치를 가지고 있다. 점수(Score)는 바운딩 박스가 물체를 영역으로 제대로 잡고 있는지와 객체의 속성 클래스를 잘 예측하였는지를 나타낸다. 각 그리드 셀은 바운딩 박스와 객체 속성 클래스에 대한 신뢰도(Confidence)를 추정하고 이를 가지고 이미지에서 객체를 찾아내게 된다. 텐서 내 각 그리드 셀의 벡터 내의 30개의 값을 살펴보면 처음 5개와 그 다음 5개의 값은 바운딩 박스의 중앙 x, y값과 바운딩 박스의 높이 h 와 너비 w, 해당 객체에 대한 신뢰도(바운딩 박스 내에 객체가 있을 확률)인 c로 구성되어 있다. 즉, 각 그리드셀 하나당 2개의 바운딩 박스를 예측한다.

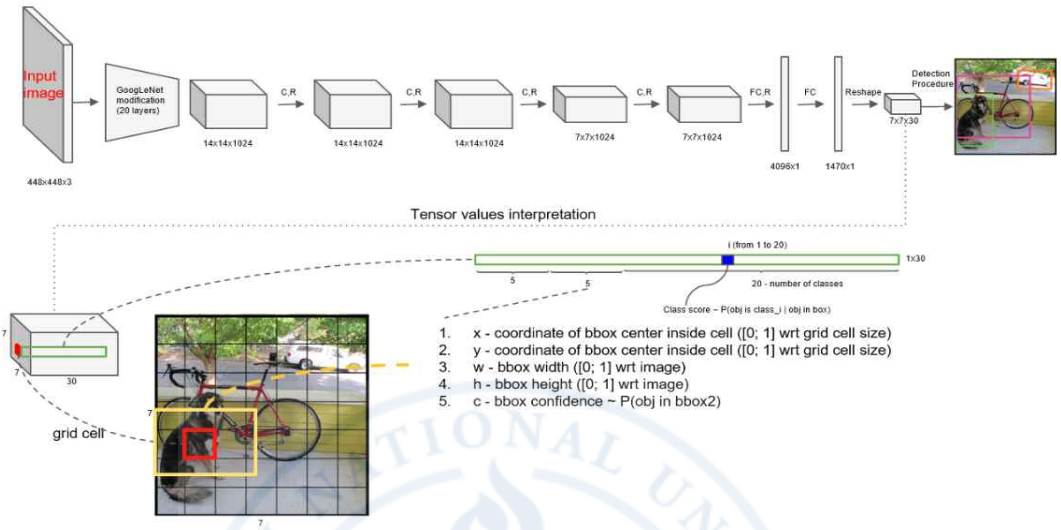

그림 3.2 YOLO 처리방식[26]

예를 들어 그리드 당 2개의 텐서 바운딩 박스를 갖고 클래스 카테고리가 20개인 PASCAL VOC에 적용하면 $B=2, C=20, S=7$으로 $S \times S \times (B*5+C)$ $7 \times 7 \times 30$ 출력 텐서(tensor)를 갖게 된다. 80개 분류를 가진 COCO 데이터 세트 가정한다면 $7 \times 7 \times 90$ 텐서를 갖는다. 이것을 합성곱 층(Convolution Network)을 통해 학습하게 된다.

그런데 각 그리드셀이 2개의 바운딩 박스를 찾는 과정에서 다음과 같이 하나의 객체에 대해 과도한 양의 경계가 감지되는 경우가 발생한다. YOLO에서는 이러한 문제점을 해결하기 위해 NMS을 사용한다. 그림 3.3 이미지에서 보시듯이 각 그리드 셀이 2개의 바운딩 박스들 중에서 겹치는 내용은 모두 제거한 다음 확률이 가장 높은 바운딩 박스를 남김으로써 해당 객체에 대해 바운딩 박스를 올바르게 표시할 수 있도록 해 준다. NMS는 추론 단계의 결과를 개선하고 중복 탐지 수를 줄이는 데 도움이 되는 마지막 단계이다. YOLO 모델은 동일한 객체에 대해 여러 개의 바운딩 박스를 예측할 수 있으며, NMS는 중복을 제거하는 데 도움이 된다[27].

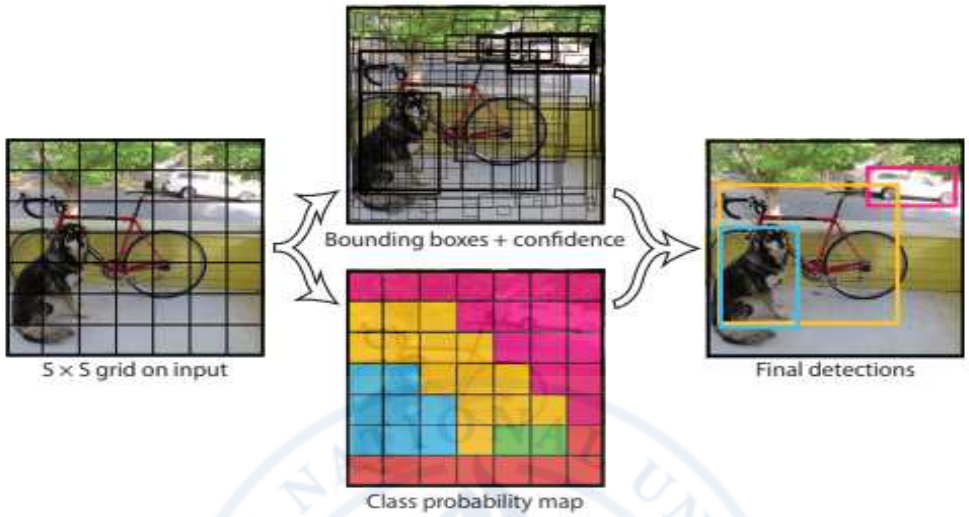

그림 3.3 YOLOv1 감지 방식[25]

3.2 YOLOv1 구조

YOLOv1의 시스템 구조는 컨볼루션 24개, 최대 풀링층 4개, 완전연결층 2개로 구성되는데 그림 3.4과 같다.

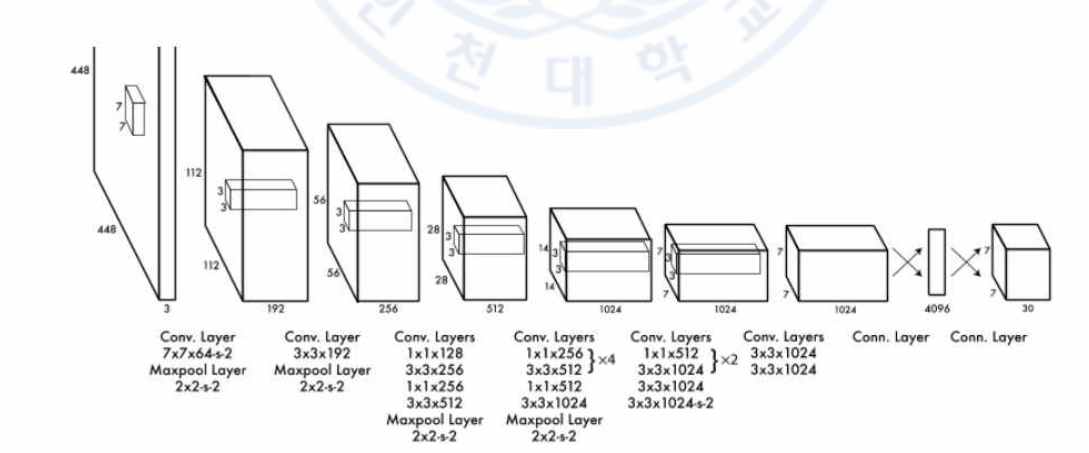

그림 3.4 YOLOv1 구조 [25]

YOLO 구조는 크게 세 부분, 백본(Backbone), 목(Neck), 헤드(Head)의 세 부분으로 구성된다. 백본(Backbone)은 이미지로부터 특징맵을 추출하는 역할을 하며 대표적인 백본으로는 Darknet-19, Darknet-53, CSPDarknet53, EfficientRep 등이 있다. 목(Neck)은 백본에서 추출된 특징을 결합하고 강화하여, 다양한 크기의 객체를 효과적으로 탐지하는 역할을 한다. FPN(Feature Pyramid Network)[14], PANet(Path Aggregation Network) 같은 구조가 목에 속한다. 헤드(Head)는 추출된 특징맵(Feature map)을 바탕으로 물체의 위치를 찾는 부분으로 이 단계에서 특정 영역에 객체가 존재할 확률, 객체의 클래스(카테고리), 그리고 객체의 정확한 위치를 나타내는 바운딩 박스를 예측한다. 대표적인 예로는 YOLO Head (YOLOv5), Efficient Head (YOLOv6), Decoupled Head (YOLOv7), Task-Aligned Head (YOLOv8) 등이 있다.

3.3 손실함수

YOLO의 손실함수는 예측한 바운딩 박스(Bounding Box)가 실제값(Ground Truth)과 얼마나 일치하는지에 대한 오차, 바운딩 박스에 객체가 있는지에 대한 여부, 해당 셀에서 객체 클래스의 신뢰도를 계산하여 오차를 측정합니다. YOLO 저자는 수식 3.2처럼 손실 함수를 5개 항목으로 나누어서 설명한다.

$$
\begin{aligned}
&\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbf{1}_{ij}^{obj}[(x_i-\hat{x_i})^2+(y_i-\hat{y_i})^2] \\
&+\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbf{1}_{ij}^{obj}[(\sqrt{w_i}-\sqrt{\hat{w_i}})^2+(\sqrt{h_i}-\sqrt{\hat{h_i}})^2] \\
&+\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbf{1}_{ij}^{obj}(c_i-\hat{c_i})^2 \\
&+\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbf{1}_{ij}^{obj}(c_i-\hat{c_i})^2 \\
&+\sum_{i=0}^{S^2}\mathbf{1}_{i}^{obj}\sum_{c\in classes}(p_i(c)-\hat{p_i}(c))^2
\end{aligned}
$$

수식 3.2 YOLO의 손실함수[25]

첫 번째 항과 두 번째 항은 각 박스의 위치 오류와 크기 오류를 측정하고 셋째

항은 실제 물체가 존재함에 불구하고 탐지되지 않는 경우 오류, 넷째 항은 물체가 없는데 있다고 했을 때 오류, 다섯째 항은 부류를 나타내는 원-핫 인코딩(One-Hot Encoding) 오류를 측정한다.

3.4 YOLO 역사

3.4.1 YOLOv1

YOLO는 2016년 6월 27일, 미국 라스베이거스에서 열린 컴퓨터비전 및 패턴인식 학회인 CVPR에서 조셉 레드몬(Joseph Redmon)이 논문[28]을 공개 발표하였다.

당시의 Mask R-CNN의 경우 아무리 빨라도 5fps의 성능을 나타나는 데 비해 높은 정확도를 보여주면서 이미지에 해당 객체에 대해 표시까지 해 주며 실시간에 근접하는 45fps의 성능을 보여주었다. GoogLeNet 모델에 영향을 받아 1×1 합성곱 층(Convolution Layer)를 도입하여 채널 수를 줄여 계산량을 줄였다는 특징이 있다.

3.4.2 YOLOv2

2017년 7월 25일, 하외이에서개최된 CVPR2017에서 조셉 레드몬은 YOLO의 두 번째 버전인 YOLO9000을 발표했다. 기존의 20개의 객체를 감지했던 YOLOv1보다 훨씬 많은 9000개의 객체를 감지한다.

YOLOv2의 사용되었던 백본은 Darket-19이라고 부르는 데 19개의 합성곱층(Convolutional layer)와 5개의 풀링 레이어(pooling layer)로 이루어져 있는 네트워크이다. 그림 3.5는 Darket-19의 구조도이다. Darket-19는 전역 평균 풀링(global average pooling)을 사용해서 예측할 뿐만 아니라 1x1 필터(filter)를 사용해서 3x3 합성곱 특징맵(convolution feature map)을 압축하는 방식을 사용하고, 배치 정규화 기법(batch normalization)을 사용해서 정규화(regularization) 한다.

Type	Filters	Size/Stride	Output
Convolutional	32	3 × 3	224 × 224
Maxpool		2 × 2/2	112 × 112
Convolutional	64	3 × 3	112 × 112
Maxpool		2 × 2/2	56 × 56
Convolutional	128	3 × 3	56 × 56
Convolutional	64	1 × 1	56 × 56
Convolutional	128	3 × 3	56 × 56
Maxpool		2 × 2/2	28 × 28
Convolutional	256	3 × 3	28 × 28
Convolutional	128	1 × 1	28 × 28
Convolutional	256	3 × 3	28 × 28
Maxpool		2 × 2/2	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Maxpool		2 × 2/2	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	1000	1 × 1	7 × 7
Avgpool		Global	1000
Softmax			

그림 3.5 DARK-19 구조도[29]

YOLOv2에서는 배치 정규화 기법(Batch Normalization), 고해상도 분류기(High Resolution Classifier), 앵커 박스(Anchor Boxes), Darknet-19 백본(backbone) 사용 등을 통해 성능이 향상되었다 그림 3.6에서는 이러한 성능 향상의 주요 요인 10가지를 추가로 적용한 결과, 모델 성능이 향상됨을 보여준다[30].

	YOLO								YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓	✓
anchor boxes?				✓	✓				
new network?					✓	✓	✓	✓	✓
dimension priors?						✓	✓	✓	✓
location prediction?						✓	✓	✓	✓
passthrough?							✓	✓	✓
multi-scale?								✓	✓
hi-res detector?									✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

그림 3.6 YOLO1과 비교 성능향상 원인[26]

3.4.3 YOLOv3

2018년 YOLOv1, YOLOv2 동일저자인 조셉 레드몬(Joseph Redmon)이 YOLOv3[31]를 발표하였다. 이 모델은 Darknet-53을 백본(backbone)으로 도입하였으며 **다중 스케일 특징 맵(multi-scale feature map)**을 활용하여 다양한 크기의 객체를 효과적으로 검출할 수 있도록 설계되었다. 이러한 구조는 빠른 추론 속도를 제공하는 데 기여하며, 실시간 객체 감지 성능을 더욱 향상했다.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

그림 3.7 DarkNet-53 구조도[32]

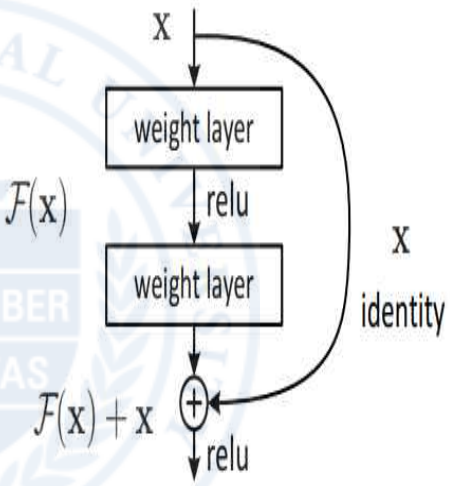

그림 3.8 Residual Block 구조[21]

다중 스케일 특징 맵은 백본 레이어에 52x52, 26x26, 13x13 크기의 세 가지 예측 레이어를 각각 덧붙여 다양의 크기의 물체를 효과적으로 검출하는 것이 특징이다. YOLOv3에서 특징 추출기로 사용되는 백본 네트워크는 Darknet-53으로 구성되어 있으며, 이는 53개의 계층을 통해 고도로 세밀한 특징을 추출하는 데 중점을 둔다.

Darknet-53 구조에는 그림 3.8처럼 잔차 블록(Residual Block)이 반복되어 실행된다. 잔차(Residual) 블록은 마이크로소프트팀이 발표한 Resnet에서 나온 개념으로 가중치 층(weight layer)를 통과한 값과 통과하지 않은 X 자신을 더하는 구조이다. 잔차 블록은 기본적으로 두 가지 주요 경로로 구성된다. 주 경로(Main Path)는 일반적

인 합성곱 연산, 활성화 함수, 배치 정규화(Batch Normalization)을 하며 잔여 경로 (Skip Connection or Shortcut Connection)는 입력을 그대로 출력으로 전달하는 경로이다. 이 경로 덕분에 네트워크의 깊이가 깊어져도 정보가 손실되지 않고 전달될 수 있다. Darknet-53에서의 Residual Block 구성 요소의 1×1 합성곱 층은 채널 수를 줄이거나 확장하는 역할을 하며 3×3 합성곱 층은 공간적인 특징을 추출하는 역할을 한다.

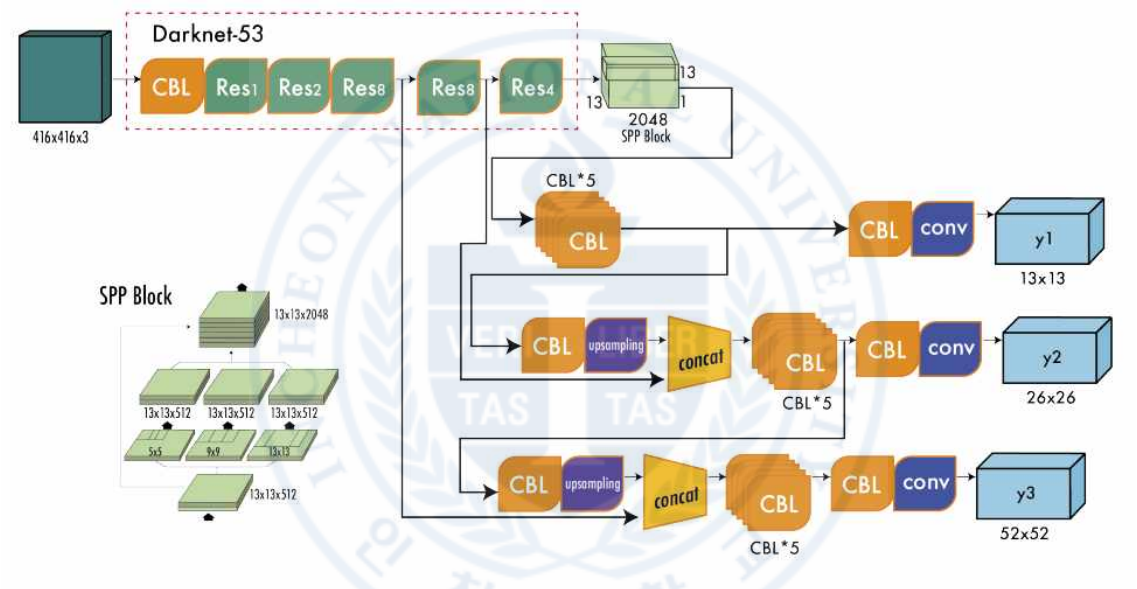

그림 3.9 YOLOV3 다중 스케일 감지 구조 [29]

3.4.4 YOLOv4

2020년 4월 알렉세이 보흐코브스키(Alexey Bochkovsky)가 YOLOv4를 발표하였다 [32]. YOLOv4는 백본(Backbone)으로 CSPDarknet53, 목(Neck)으로 SPP(Spatial Pyramid Pooling), PAN, 헤드(Head)로 YOLOv3 구조를 채택하였다. 또한 싱글 GPU에서 훈련에 적합한 Bag of freebies(공짜 가방), Bag of Specials(스페셜 백) 기법 같은 객체 감지 학습 방법이 도입되었다.

공짜 가방(Bag of freebies)은 훈련 전략을 변경하거나 훈련 비용을 증가시켜 성

능을 개선하는 방법을 의미한다. 공짜 가방의 대표적인 예로 데이터 증강(data augmentation)이 있다. 데이터 증강의 목적은 입력 이미지의 다양성을 높여, 설계된 객체 감지 모델이 다양한 환경에서 수집된 이미지에 대해 더 높은 견고성을 갖추도록 하는 것이다. 이 연구에서는 모자이크(Mosaic)와 자기 적대적 훈련(Self-Adversarial Training) 기법을 도입하여 모델 성능을 향상했다.

스페셜 백(Bag of Specials**)**은 추론 비용은 증가하지만, 객체 감지 정확도를 크게 향상할 수 있는 플러그인 모듈과 후처리 방법을 말한다. Cross-stage partial connections(CSP), Multiinput weighted residual connections (MiWRC), SPP-block 등이 포함된다[32].

3.4.5 YOLOv5

YOLOv5는 Ultralytics(울트라리틱스) 기업에서 개발되었으며 공식 논문 없이 2020년 6월에 공개되었다. YOLOv5 구조는 그림 3.10과 같이 크게 세 부분으로 나눌 수 있다[33]. 백본(Backbone)은 이미지로부터 특징 맵(Feature map)을 추출하는 부분으로, 이전 버전에서 사용된 다크넷 아키텍처를 수정한 CSP-Darknet를 사용한다. CSP-Darknet는 CSPNet, BottleneckCSP, SPP(Spatial Pyramid Pooling)를 결합한 구조이다.

목(Neck)은 백본과 헤드를 연결하는 부분으로 YOLOv5에서는 SPPF 그리고 New CSP-PAN 구조가 사용된다. 헤드(Head)는 최종 출력을 생성하는 부분으로 YOLOv5에서는 YOLOv3 헤드 구조를 사용한다.

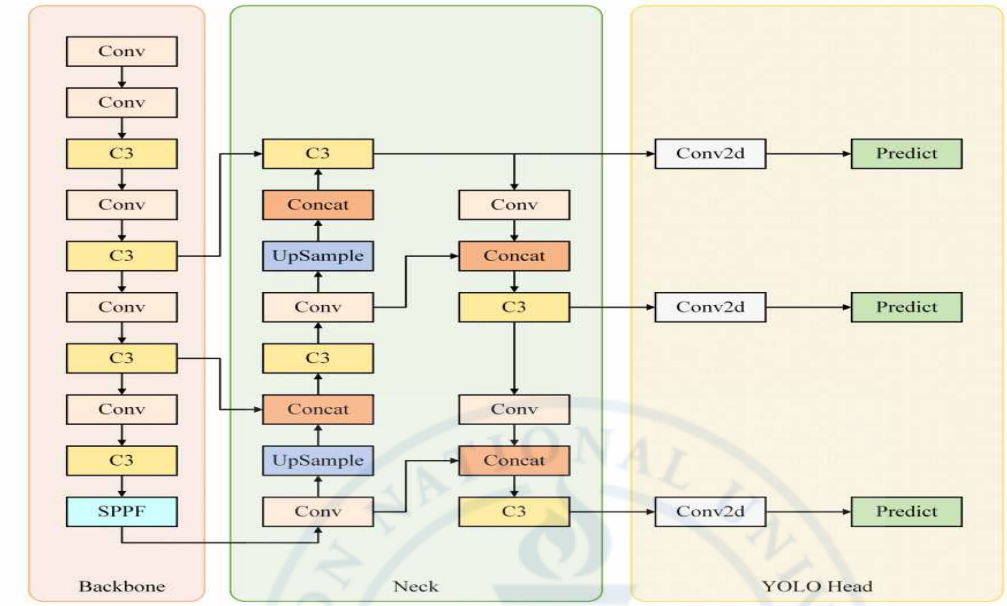

그림 3.10 YOLOv5 구조[33]

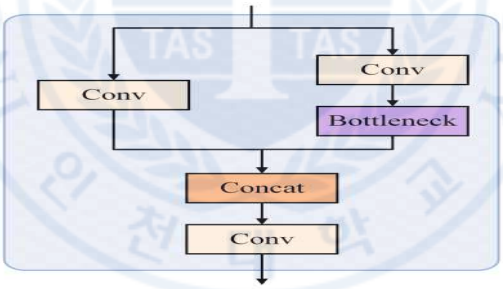

그림 3.11 C3 구조도[33]

3.4.6 YOLOv6(MT-YOLO)

메이투안 비전 AI 연구부서(Meituan Vision AI Department)는 중국의 대형 인터넷 기업인 Meituan(메이투안)의 비전 인공지능(Vision AI) 부서이다, 이 연구팀은 YOLO 저자의 허가를 받아 개발한 객체 탐지 알고리즘을 YOLOv6로 명명하여 2022년 9월 7일에 공개하였다[34]. YOLOv6는 하드웨어 친화적인 효율성과 고성능을 겸비한 산업용 애플리케이션 전용 단일 단계 객체 감지 모델 구조를 갖추고 있다. MS CO

CO Dataset 에서의 벤치마크 결과, YOLOv6는 YOLOv5보다 우수한 성능을 보였다.

초기버전 1.0에서는 그림 3.12과 같이 다양한 크기의 모델을 수용하기 위해 두 개의 확장된 재파라미터화 가능한 백본과 넥, 그리고 하이브리드 채널 전략으로 효율적인 디커플링 헤드를 제안했다[28].

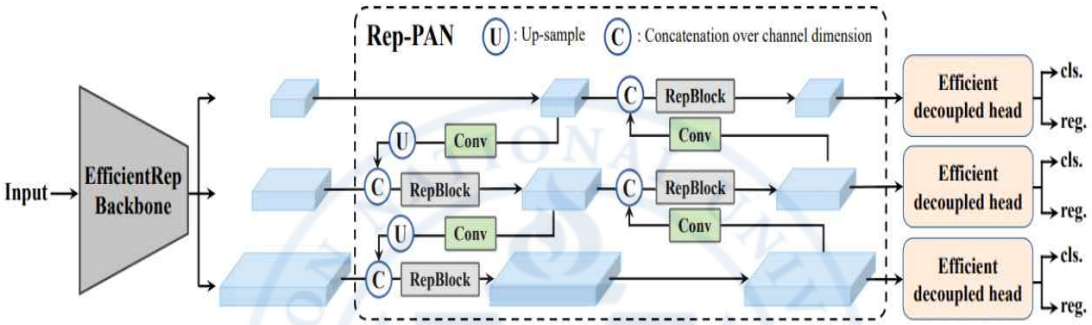

그림 3.12 YOLOv6 모델 구조[28]

2023년에 업데이트된 v3.0 버전에서는 v1.0의 목(Neck) 구조를 대체한 양방향 연결(BiC, Bi-directional Concatenation) 모듈과 앵커 보조 훈련(AAT, Anchor-Aided Training)전략, 전략, 자가 증류(self-distillation) 전략[35]과 같은 아키텍처 개선 사항을 특징으로 하고 있다. 앵커 보조 훈련(AAT) 전략은 추론의 효율성에 영향을 주지 않으면서 앵커기반(Anchor-Based)와 앵커프리(Anchor-Free)의 장점을 모두 가져가기 위한 전략이다. 자가 증류(self-distillation) 전략에서는 더 작은 YOLOv6 모델의 성능을 향상해 훈련 중 보조 회귀 분기를 향상하고 추론 시 이를 제거하여 현저한 속도 저하를 방지했다[36].

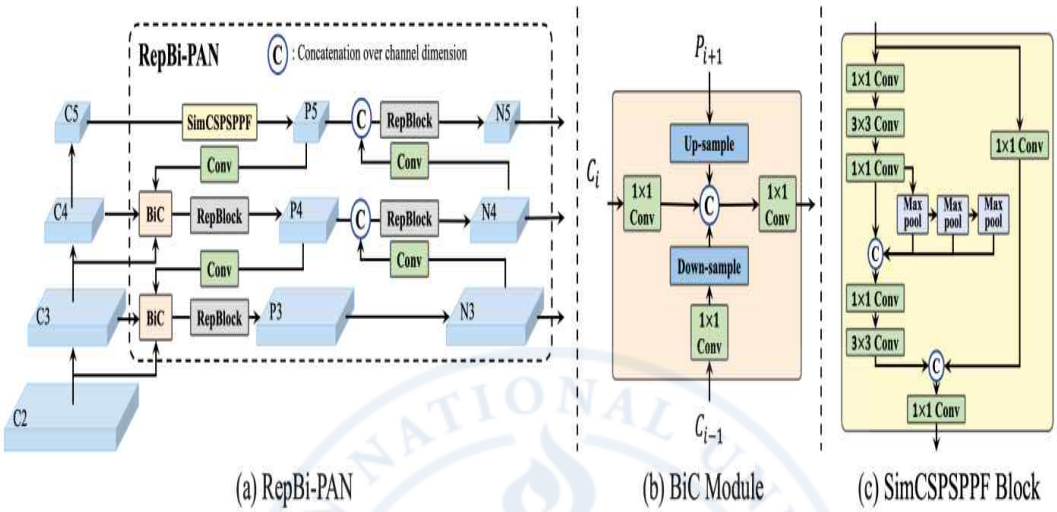

그림 3.13 YOLOv6 v3.0의 목(Neck)[37]

3.4.7 YOLOv7

2022년 7월, “YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-tim e object detectors”[38]라는 제목의 논문이 발표되었다. 이 논문에서는 일반용 GPU, 엣지(Edge) GPU, 클라우드 GPU 환경에 맞춘 다양한 모델 구조인 YOLOv7-X, YOL Ov7-E6/D6, YOLOv7-E6E 등을 소개하고 있다. 또한, 새로운 네트워크 구조인 E-ELA N(Extended Efficient Layer Aggregation Networks)을 도입하여 모델의 학습 능력을 향상했다.

제안된 E-ELAN은 기존 아키텍처의 기울기 전송 경로를 변경하지 않으면서도, 그룹 컨볼루션을 통해 특징의 카디널리티(Cardinality)를 높인다. 이 과정에서 서로 다른 그룹의 특징들을 셔플(Shuffle)하고 병합(Merge)하여 결합하는 방식을 사용하였다. 이러한 방식은 서로 다른 피처 맵에서 학습된 특징을 효과적으로 향상하며, 매개변수와 계산 자원의 효율성을 개선하는 데 기여하였다.

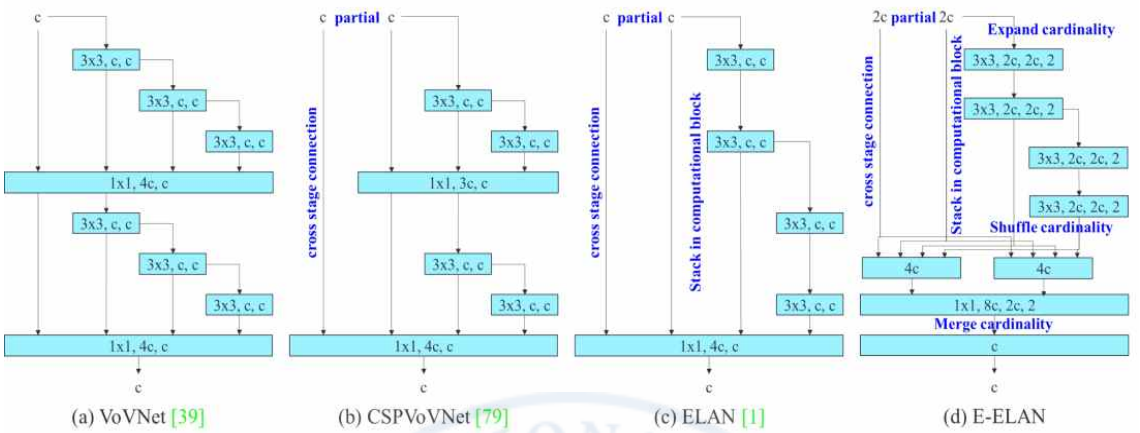

그림 3.14 Extended Efficient Layer Aggregation 구조[38]

이 외에도 캐스케이드 기반 모델 스케일링 방식(Concatenation-based mode)을 채택하여 실제 작업에 적합한 규모의 모델을 생성하여 탐지 요구 사항을 충족할 수 있도록 하였다.

그리고 계획된 재 파라미터화(Planned re-parameterized convolution), 동적 레이블 할당 확장, 거칠고 세밀한 리드 헤드 가이드 라벨 할당(Coarse for auxiliary and fine for lead loss) 등 훈련 가능한 공짜 가방(Trainable bag-of-freebies) 전략 통해 YOLOv7은 모델에 따라 차이가 있지만 비슷한 규모의 YOLOv5-X등과 비교해서 연산량과 파라미터 수 감소에도 불구하고 보다 추론 속도가 빠르거나 더 향상된 AP(Average Precision)을 보여주었다. YOLOv7 다이어그램은 다음과 같다[39].

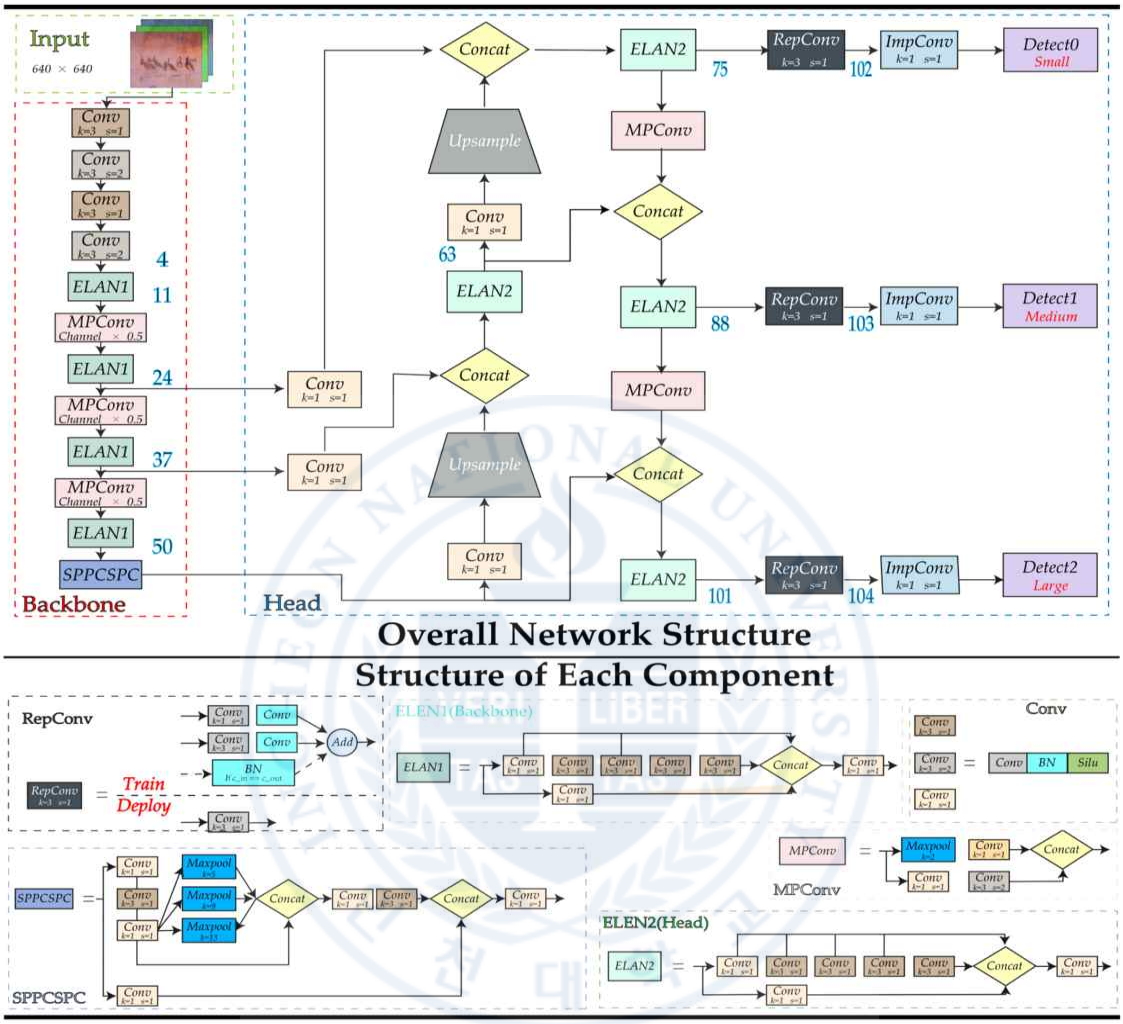

그림 3.15 YOLO7 네트워크 구조[39]

3.4.8 YOLOv8

YOLOv8은 2023년 1월, YOLOv5를 개발한 미국 소재 회사 Ultralytics에서 출시한 모델로, 물체 감지뿐만 아니라 분할(Segmentation), 포즈 추정(Pose Estimation), 추적(Tracking), 분류(Classification) 등 다양한 비전 작업을 지원하는 것이 특징이다. 또한, 플랫폼 형태로 제공되어 사용자의 개발 편의성을 높이고 있다.

YOLOv8의 백본 부분은 기본적으로 YOLOv5의 백본과 동일하며, C3 모듈은 CS

P 아이디어[39]에 기반한 C2f 모듈로 대체되었다. C2f 모듈은 YOLOv7의 ELAN 아이디어에서 학습하여 C3와 ELAN을 결합하여 C2f 모듈을 형성함으로써 YOLOv8이 경량화를 보장하면서 더 풍부한 경사 흐름 정보를 확보할 수 있게 되었다. 그림 3.16은 C3_X(YOLOv5)와 C2f(YOLOv8)의 모듈 간 구조적 차이를 비교하여 보여준다.

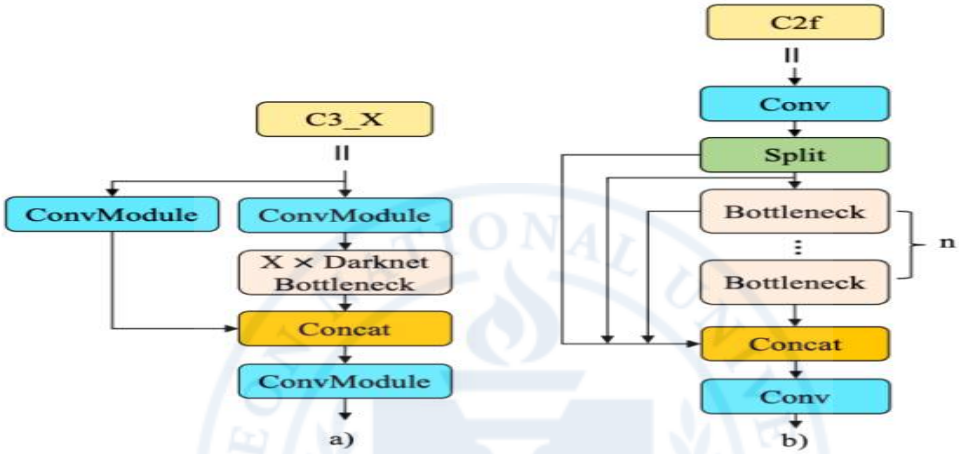

그림 3.16 C3(YOLOv5) 구조와 C2f(YOLOv8) 모듈구조 비교[40]

YOLOv8의 아키텍처는 수정된 CSPDarknet53 백본을 사용하며, 이 구조는 네트워크의 여러 단계 간 정보 흐름을 개선하고 훈련 중 Gradient Flow(그라이언트 흐름)를 강화하는 새로운 CSP(Cross-Stage Partial) 연결을 도입하였다. 그림 3.17는 YOLOv8의 전체 구조를 나타내고 있으며, 해당 구조의 세부 사항은 YOLOv8 아키텍처 링크에서 확인할 수 있다[41].

YOLOv8의 Neck은 백본의 여러 단계에서 추출된 특징 맵을 병합하여 다양한 규모의 정보를 캡처하는 역할을 한다. 기존의 FPN(Feature Pyramid Network) 대신 C2f 모듈을 활용하며, 이 모듈은 높은 수준의 의미론적 특징과 낮은 수준의 공간 정보를 결합하여 작은 물체에 대한 감지 정확도를 특히 향상한다.

또한, YOLOv8의 Head는 각 특징 맵의 그리드 셀에 대해 경계 상자, 객체 점수, 그리고 클래스 확률을 예측하는 여러 감지 모듈을 사용하여 최종 탐지 결과를 산출한다. 이를 통해 YOLOv8은 보다 정확한 객체 감지 성능을 제공한다[42].

MS COCO 2017 데이터 세트를 기준으로 평가한 결과, YOLOv8x는 640픽셀 입력 이미지 크기에서 53.9%의 AP(평균 정밀도)를 기록하여, 동일한 입력 크기에서 YOLOv5의 50.7%와 비교해 더 높은 성능을 보여주었다. 또한, NVIDIA A100과 TensorRT 환경에서 280 FPS의 처리 속도를 달성하였다[33].

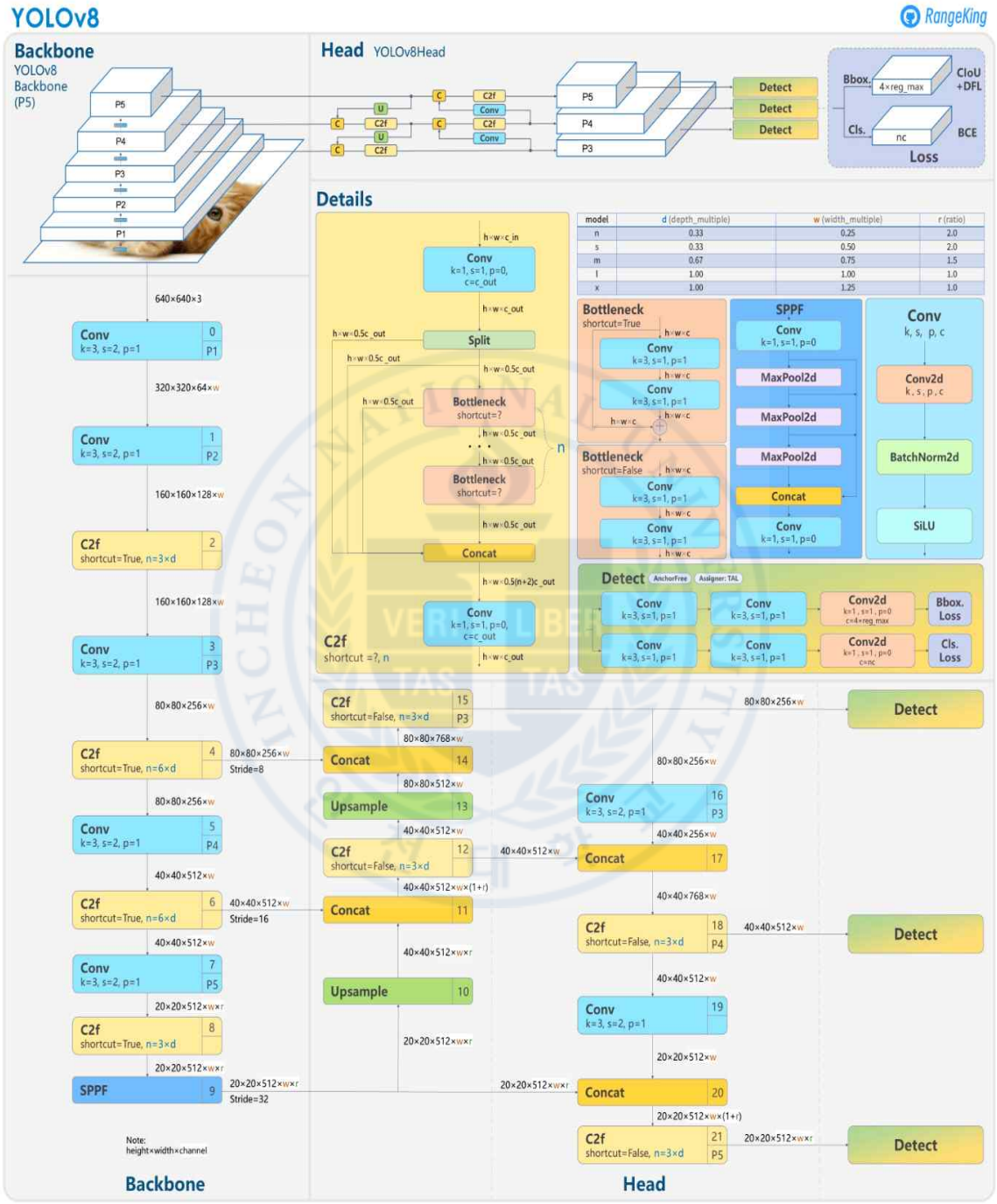

그림 3.17 YOLOv8-P5 네트워크구조[36]

3.4.9 YOLOv9

YOLOv9는 2024년 2월, 대만의 Chien-Yao Wang 등 세 명의 연구자가 발표한 실시간 객체 탐지 모델로, 딥러닝 학습 과정에서 발생하는 정보 손실 문제를 완화하기 위해 프로그래밍 가능한 그래데이션 정보(Programmable Gradient Information) 개념을 도입하였다[43]. PGI는 학습 과정에서 필요한 정보를 효율적으로 보존하고 전달하는 역할을 하며, 이를 통해 모델 성능을 크게 향상했다. 그래디언트는 특정 지점에서 함숫값이 가장 가파르게 증가하는 방향과 그 정도를 나타내는 벡터이며, 딥러닝에서는 이 그래디언트가 모델의 가중치를 업데이트하는 데 중요한 역할을 한다.

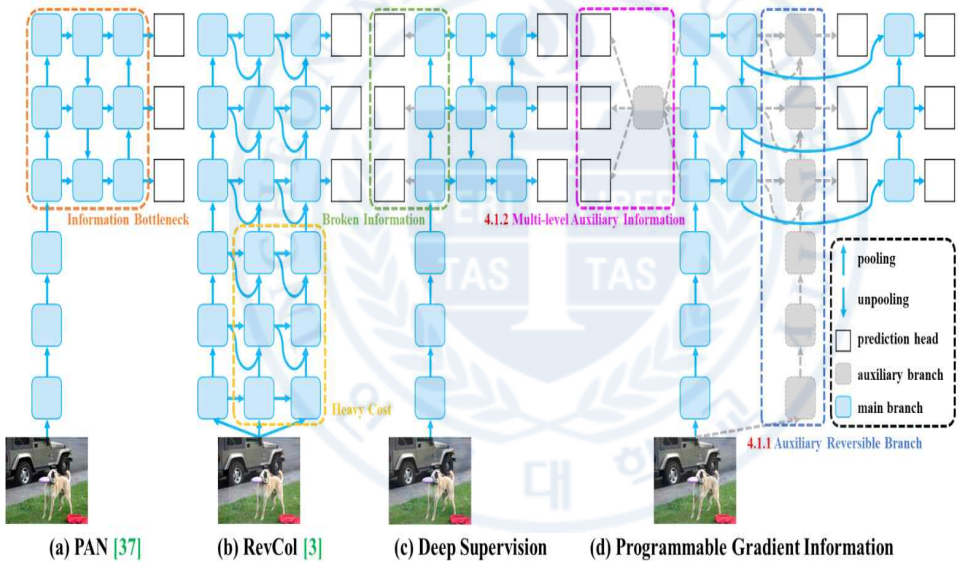

그림 3.18 PGI 와 관련 네트워크 구조, 방법[43]

YOLOv9은 PGI와 함께 Generalized Efficient Layer Aggregation Network (GELAN) 구조를 도입함으로써 실시간 객체 탐지 성능에서 상당한 발전을 이루었다. GELAN은 기존의 Efficient Layer Aggregation Network (ELAN)과 Cross-Stage Partial Network (CSPNet)의 장점을 결합한 효율적인 계층 통합 네트워크로, 경량화, 추론 속도, 그리고 정확도를 모두 고려한 최적화된 구조이다. 특히 GELAN은 ELAN의 그래디언트 경로를 다변화하고, CSPNet의 특징 맵 분할 기법을 결합하여 연산 자원을 줄이면서

도 높은 성능을 유지함으로써 모델의 효율성을 크게 향상했다.

또한, YOLOv9은 MS COCO 데이터 세트에서 새로운 벤치마크를 설정하며, 이전 YOLO 모델들에 비해 더욱 높은 정확성과 효율성을 입증하였다. 특히 YOLOv9c 모델은 YOLOv7 AF에 비해 42% 적은 파라미터와 21% 적은 연산량으로 유사한 수준의 정확도를 달성하였으며, YOLOv9e 모델은 YOLOv8x보다 15% 적은 파라미터와 25% 적은 연산량으로 작동하면서도 AP가 1.7% 향상된 성능을 보였다. 이러한 성과는 YOLOv9이 대형 모델의 새로운 표준을 제시했음을 의미한다.

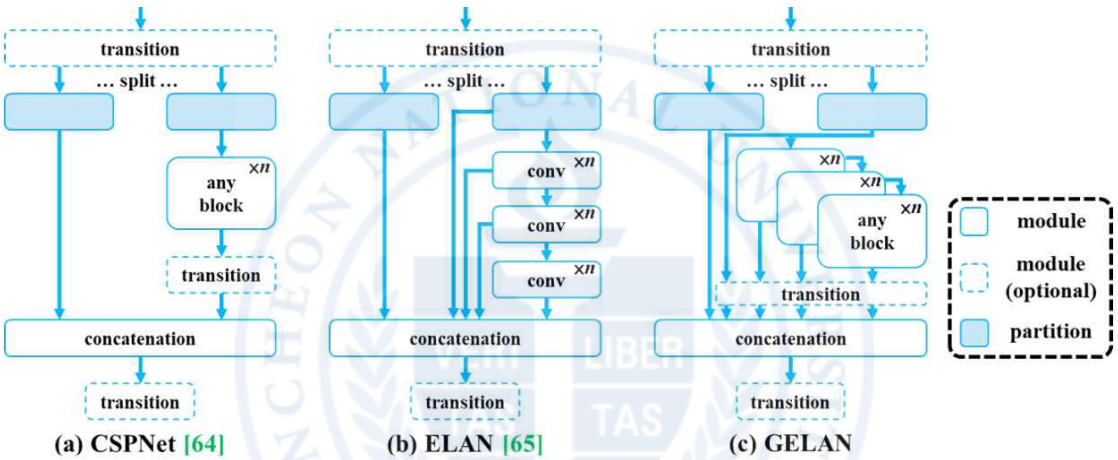

그림 3.19 GELAN 구조와 CSPNet, ELAN 비교[38]

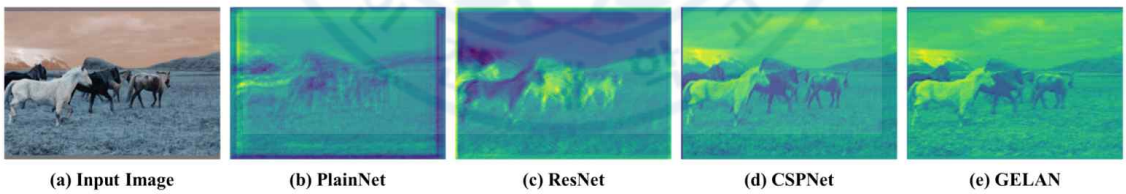

그림 3.20 다양한 네트워크에서 시각화 초기 가중치 결과[43]

3.4.10 YOLOv10

칭화대학교(Tsinghua University) 연구진은 2023년 11월 NeurIPS 2023 콘퍼런스에서 “YOLOv10: Real-Time End-to-End Object Detection”이라는 제목으로 새로운 객체 탐지 모델을 발표하였다[44]. 이 연구에서는 YOLO 탐지 파이프라인 전반에 걸친

후처리 과정과 모델 아키텍처의 주요 개선 사항을 다루고 있다.

YOLOv10의 백본은 CSPNet(Cross Stage Partial Network)의 향상된 버전을 채택하여, 그래디언트 흐름을 개선하고 계산 중복을 줄여 모델의 효율성을 높였다. 또한, 목(Neck) 구조에서는 PAN(Path Aggregation Network) 계층을 포함하여 다중 스케일 특징의 효과적인 융합을 가능하게 하였다. 이를 통해 다양한 크기의 객체를 더 정확하게 탐지할 수 있게 되었다.

모델 아키텍처 측면에서 YOLOv10은 효율성과 정확도 간의 트레이드오프를 개선하기 위해 효율성 중심의 설계 전략을 도입하였다. 이를 위해 **경량 분류 헤드(lightweight classification head), 공간-채널 분리형 다운샘플링(spatial-channel decoupled downsampling), 그리고 랭크-가이드 블록 설계(rank-guided block design)** 등의 기술을 채택하여, 모델의 파라미터 수, FLOPs(Floating Point Operations), 그리고 지연 시간을 줄이는 데 기여하였다. 표 3.1은 이러한 효율성 개선 요인과 그 효과를 수치로 입증한 내용을 포함하고 있다[44].

표 3.1 YOLO10-S/M 모델에서 효율성 개선[44]

#Model	#Param	FLOPs	AP^{val}	Latency
기본(base)	11.2/25.9	28.6/78.9	44.3/50.3	2.44/5.22
+분류 헤드	9.9/23.2	23.5/67.7	44.2/50.2	2.39/5.07
+다운샘플링	8.0/19.7	22.2/65.0	44.4/50.4	2.36/54.97
+블록설계	6.2/14.1	20.8/58.1	44.5/50.4	2.31/4.57

또한 YOLOv10은 NMS-Free 방식과 **일관된 이중 할당(Consistent Dual Assignment)** 전략을 통해 종단형(end-to-end) 실시간 감지를 가능함. 가능하게 하였다. 두 개의 예측 상자를 할당하여 객체의 위치와 크기를 더욱 정확하게 예측하고, NMS(Non-Maximum Suppression) 없이도 겹치는 예측 상자를 제거한다. 이를 통해 객체 탐지 결과를 출력하는 모든 단계를 하나의 통합된 모델로 처리함으로써 즉각적인 결과 출력과 낮은 지연 시간을 달성하였다.

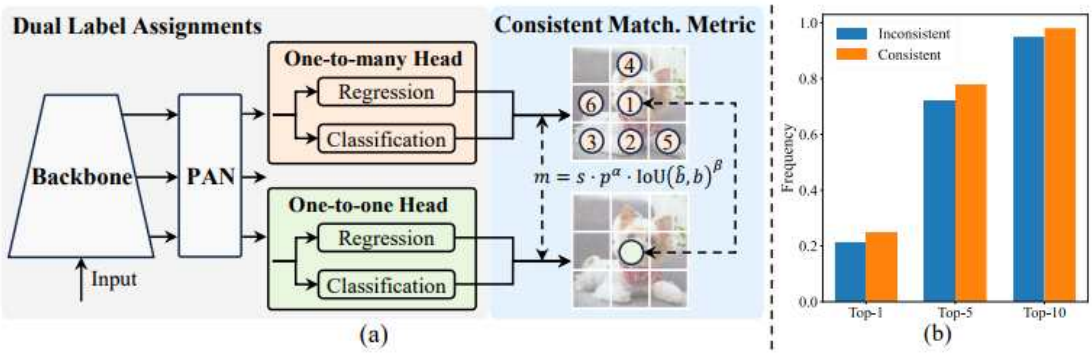

그림 3.21 YOLOv10 일관된 이중 할당[44]

그림 3.21의 (a) 그림은 엔드투엔드(end-to-end) 실시간 탐지를 달성하기 위해 NMS-free 훈련을 위한 일관된 이중 할당을 보여주고 있다. One-to-Many Head는 훈련 중 개체당 여러 예측을 생성하여 풍부한 감독 신호를 제공하고, 학습 정확도를 향상한다. 반면 One-to-One Head는 추론 시 개체당 하나의 최적 예측을 생성하여 NMS 없이도 높은 정확도를 유지하며 지연 시간을 줄이고 모델의 효율성을 크게 개선한다.

3.5 YOLO 버전들의 요약

본 섹션에서는 YOLO의 다양한 버전들에 대해 특징을 정리 요약하였다.

표 3.2 YOLO 버전들 특징 요약[29][45]

버전	연도 / 저자 / 논문(소스)	주요특징
1	2016.06 Joseph Redmon 등 4명 You Only Look Once: Unified, Real-Time Object Detection	• CNN 기반, Feature Map 추출 • Bounding Box 적용 • Darknet-19 모델 적용 • 단일 크기의 객체 감지

버전	연도 / 저자 / 논문(소스)	주요특징
2	2016.12. Joseph Redmon 등 2명 YOLO9000: Better, Faster, Stronger	• 9000 개의 객체 감지 • v1 의 단점 보완 • 배치 정규화(Batch Normalization)기법 도입 • Darknet-53 모델 적용 • 다양한 크기의 객체 감지 • 일부 객체 인식 시 문제 발생
3	2018.4 Joseph Redmon, Ali Farhadi / 미국 워싱턴대학 YOLOv3: An Incremental Improvement	• 백본: DarkNet53 • 목(Neck): FPN 적용 • 작은객체 감지 성능 개선 • 3 개의 바운딩 박스(Bounding Box) 적용 • 위치개선을 위해 SPP-back 적용 • GPU 이용 처리속도 개선(실시간 객체탐지 가능) • 다중 분류(Multilabel Classification)
4	2020.4 Alexey Bochkovskiy,Chien-Yao Wang, Hong-Yuan Mark Liao YOLOv4: Optimal Speed and Accuracy of Object Detection	• v3 의 성능과 처리 속도 개선(CSPNet 적용) • 작은 객체 감지 개선(PANet 적용) • GPU 를 이용한 v3 에 정확도를 높임 • 데이터 증강모델과 모델 규제를 위한 모자이크(Mosaic) 증강 기법 적용
5	2020.5 Ultralytics 회사 논문발표없음. github.com/ultralytics/yolov5	• Ultralytics 에서 PyTotch 기반으로 발표 • v4 대비 속도와 정확도 2~4 배 개선 • 경량화모델 지원(모바일, 엣지) • CPU, GPU 모두 지원 • AutoML 기능 적용 • 백본: CSPDarknet53 • 목(Neck): PANNet

버전	연도 / 저자 / 논문(소스)	주요특징
6	2022.9.7 Chuyi Li 등 중국 메인투안(Meituan)사의 AI 연구부서 YOLOv6 v3.0: A Full-Scale Reloading YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications	• 하드웨어 친화적인 효율적인 설계와 고성능을 갖춘 산업용 애플리케이션 전용 단일 단계 객체 감지 프레임워크 • EfficientRep 백본과 Rep-PAN 넥(Neck)도입 • 백본: RepVGG & CSPRepStack • 목: RepPAN • v5 의 속도와 정확도 증대 • 포즈(Pose) 추정 기능 추가
7	2022.6 Chien-Yao Wang 등 3명 Taiwan YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors	• 백본: EELAN • 목(Neck) : PANNet • v4 를 기반으로 확장 • 실시간 객체 검출을 위한 확장 및 스케일링 제안
8	2023.01 Ultralytics 회사 논문발표없음. github.com/ultralytics/ultralytics	• 최첨단 SOTA 모델 적용 • 감지, 분할, 포즈 추정, 추적기능 고도화 • 비전 AI 작업 지원
9	2023.2 Chien-Yao Wang 외 2인 YOLOv9:Learning What You Want to Learn Using Programmable Gradient Information	• ELAN 과 CSPNet 을 결합한 GELAN 구조 사용 • Programmable Gradient Information (PGI) 개념 도입

버전	연도 / 저자 / 논문(소스)	주요특징
10	2024.5 Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, Guiguang Ding /Tsinghua University YOLOv10: Real-Time End-to-End Object Detection	• NMS-Free 방식과 Consistent Dual Assignment 전략을 통해 end-to-end 실시간 감지를 가능

제 4 장 YOLO 모델 성능 비교

4.1 연구 방법

4.1.1 연구 실험 환경

실험 환경은 Windows 64bit 위에 GPU 사용할 수 있는 H/W 환경과 YOLO 모델을 이용할 수 있는 딥러닝 S/W 도구 및 라이브러리를 구축하였다. 구체적인 실험 환경 및 버전에 대한 요약은 그림 4.1과 표 4.1에 제시하였다. 또한, 실험을 모니터링하고 성능을 비교하기 위해 WandB(Weights & Biases) 플랫폼을 사용하였다. “WandB”는 머신러닝 실험 관리를 위한 도구로, 데이터 실험을 추적하고, 평가하며, 비교할 수 있을 뿐만 아니라, 결과를 시각화하고 쉽게 공유할 수 있는 플랫폼이다[1].

표 4.1 개발 H/W 및 S/W 환경

구 분	사양 및 세부 내용
CPU	Intel i9-9900KF 3.6GHz
RAM	32GB
OS	Windows 11
비디오카드	NVIDIA GeForce RTX 3090
S/W	PyCharm (개발도구,언어) OpenCV, PytQT5 (응용 구현 라이브러리) Python 3.8 /Anaconda (개발언어 및 환경) CUDNN 8.1
개발언어 및 환경	Python-3.10.5 / Anaconda
딥러닝 라이브러리	torch-1.13.1+cu116 CUDA:0

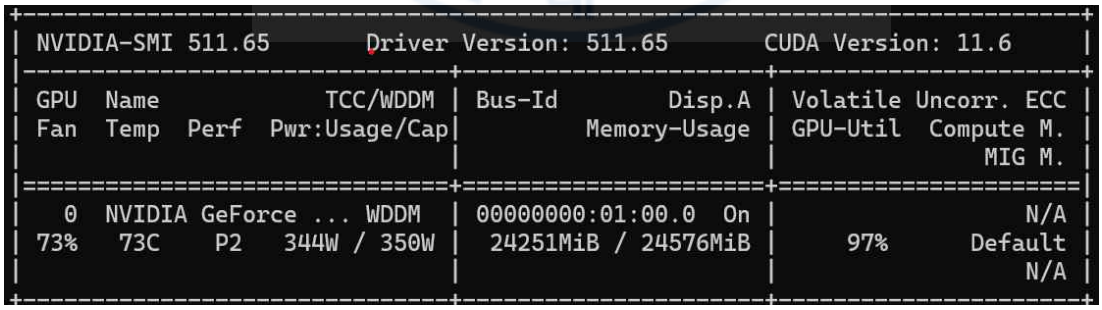

그림 4.1 개발환경 GPU 드라이버 및 CUDA 버전

1) https://wandb.ai/

4.1.2 데이터 세트

흡연 감지를 위한 딥러닝을 학습시키기 위해서 공개 오픈되어 있어 있는 로보플로우(Roboflow) 사이트[2]의 흡연 데이터를 다운하여 활용하였다. 해당 사이트는 다양한 YOLO 버전들의 형식으로 학습 데이터 세트(Dataset)를 편리하게 제공해 주고 있다. 여기서 활용된 데이터 세트는 훈련 데이터(Train Data) 5307장, 검증 데이터(Validation Data) 270, 테스트 데이터(Test Data) 81장으로 구성되어 있고 담배(Cigarette), 사람(Person), 연기(Smoke), 전자담배(Vape) 4가지 클래스로 정의되어 있다[46].

그림 4.2 흡연 데이터 세트 구성

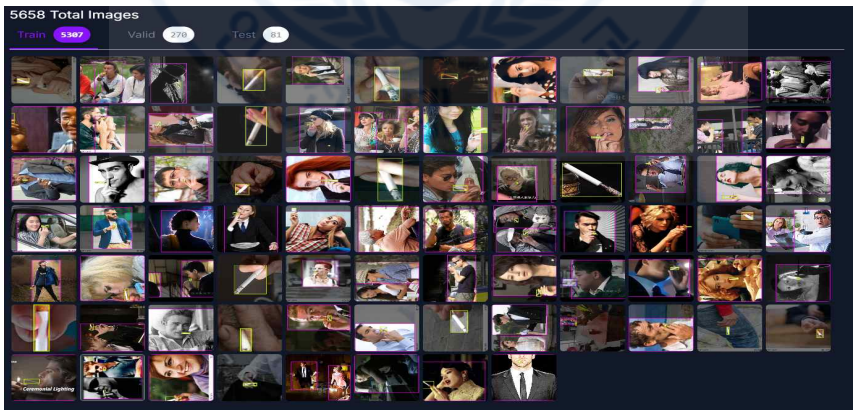

그림 4.3 흡연 데이터 세트의 이미지들

2) https://roboflow.com/

4.2 학습 과정

YOLO 버전별로 성능 차이를 비교 평가하기 위해 동일 흡연 데이터 세트를 가지고 진행하였고 각 모델은 COCO 데이터 세트로 사전 학습된 가중치를 기반으로 전이 학습(Transfer Learning) 수행하였다.

YOLO 버전들의 모델들은 일반적으로 네트워크의 깊이와 넓이에 따라 모델 크기가 나누어지며 버전별로 차이는 있으나 대체로 5가지 종류로 구분된다. 이러한 모델 크기는 학습 파라미터의 양을 결정짓는 중요한 요소로 작용한다. 본 연구에서는 각 YOLO 버전에서 제공되는 5가지 주요 모델을 중심으로 테스트를 수행하였다.

YOLOv5, YOLOv6, YOLOv8, YOLOv10의 경우, 모델 크기는 n, s, m, l, x로 구분되며, 반면, YOLOv9에서는 모델 크기에 따라 T, S, M, C, E와 같은 명칭을 사용하여 구분하였다.

각 모델의 크기는 설정 파일인 yolov{버전번호}{모델명}.yaml에서 정의되며, 이 파일에는 네트워크의 깊이(depth)와 넓이(width)가 포함되어 있다. 깊이(depth)의 값이 높을수록 백본 모듈의 반복 횟수가 증가하며, 넓이(width)의 값이 클수록 합성곱(convolution) 필터 수가 증가하여 학습량이 많아진다. 예를 들어 값이 0.33이면 기본 네트워크의 깊이나 넓이를 1/3로 줄여 조정한 것이고, 0.50이면 절반을 의미한다. YOLOv9 경우는 다른 모델과 다르게 깊이와 넓이 스케일이 아닌 각각 레이어에 직접값을 다르게 지정하고 있다. 다음 표는 YOLO 버전별 모델 크기와 설정값이다.

표 4.2 YOLO 모델 크기 설정값

version	model	depth	width
V5,V6,V8	n	0.33	0.25
	s	0.33	0.50
	m	0.67	0.75
	l	1.00	1.00
V5,V6,V8	x	1.33	1.25
V10	x	1.00	1.25

실제 학습된 파라미터 수는 초깃값 구성값과 비교 시 동일하지 않지만 오차범위 내에서 거의 유사하다. 모델별로 초깃값 기준으로 레이어와 파라미터, 그래디언트,

GFLOP(Giga Floating Point Operations Per Second)은 다음과 같다.

표 4.3 YOLO 버전 레이어, 파라미터, 그래디언트, GFLOPs

version	model	layer	parameters	gradients	GFLOPs
V5	n	262	2,509,244	2,509,228	7.2
	s	262	9,123,740	9,123,724	24.0
	m	339	25,067,452	25,067,436	64.4
	l	416	53,166,428	53,166,412	135.3
	x	493	97,203,260	97,203,244	246.9
V6	n	195	4,238,540	4,238,524	11.9
	s	195	16,306,620	16,306,604	44.2
	m	295	25,858,636	25,858,620	79.1
	x	351	173,024,908	173,024,892	611.2
V7	7	415	37,622,682	37,622,682	105.2
	7x	467	70,835,306	70,835,306	188.9
V8	n	225	3,011,628	3,011,612	8.2
	s	225	11,137,148	11,137,132	28.7
	m	295	25,858,636	25,858,620	79.1
	l	365	43,632,924	43,632,924	165.4
	x	365	68,156,460	68,156,444	258.1
V9	t	917	2,006,188	2,006,172	7.9
	s	917	7,288,956	7,288,940	27.4
	m	603	20,161,212	20,161,196	77.6
	c	618	25,532,316	25,532,300	103.7
	e	1,225	58,147,996	58,147,980	192.0
V10	n	385	2,708,600	2,708,584	8.4
	s	402	8,069,448	8,069,448	24.8
	m	498	16,488,760	16,488,744	64.0
	l	628	25,771,496	25,771,480	127.2
	x	668	31,662,584	31,662,568	171.0

하이퍼파리미터는 버전별 기본적으로 제공해주고 있는 디폴트 값을 사용하였다, 에폭은 사전 랜덤 테스트시 조기종료(Early Stopping) 되는 값이 250회를 넘어가지 않는 것을 확인하여 점검하여 250회로 정하였다. 학습 시 사용되었던 하이퍼파라미터값은 다음과 같다.

표 4.4 YOLO 버전들의 하이퍼파라미터(Hyperparameters) 설정값

Hyperparameters	v5	v6	v7	v8	v9	v10
lr0	0.01	0.01	0.01	0.01	0.01	0.01
lrf	0.1	0.01	0.1	0.01	0.01	0.01
momentum	0.937	0.937	0.937	0.937	0.937	0.937
weight_decay	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
warmup_epochs	3.0	3.0	3.0	3.0	3.0	3.0
warmup_momentum	0.8	0.8	0.8	0.8	0.8	0.8
box	0.05	7.5	0.05	7.5	7.5	7.5
cls	0.5	0.5	0.3	0.5	0.5	0.5
cls_pw	1.0	N	1.0	N	1.0	N
obj	1.0	N	0.2	N	0.7	N
obj_pw	1.0	N	3	N	1	N
iou_t	0.2	N	0.2	N	0.2	N
anchor_t	4.0	N	4.0	N	5.0	N
fl_gamma	0.0	N	0.0	N	0.0	N
hsv_h	0.015	0.015	0.015	0.015	0.015	0.015
hsv_s	0.7	0.7	0.7	0.7	0.7	0.7
hsv_v	0.4	0.4	0.4	0.4	0.4	0.4
degrees	0.0	0.0	0.0	0.0	0.0	0.0
translate	0.1	0.1	0.2	0.1	0.1	0.1
scale	0.5	0.9	0.5	0.9	0.9	0.9
shear	0.0	0.0	0.0	0.0	0.0	0.0
perspective	0.0	0.0	0.0	0.0	0.0	0.0
flipud	0.0	0.0	0.0	0.0	0.0	0.0
fliplr	0.5	0.5	0.5	0.5	0.5	0.5
mosaic	1.0	1.0	1.0	1.0	1.0	1.0
mixup	0.0	0.0	0.0	0.0	0.15	0.0
copy_paste	0.0	0.0	0.0	0.0	0.3	0.0
paste_in	N	N	0	N	N	N
loss_ota	N	N	1	N	N	N
dfl	N	1.5	N	1.5	1.5	1.5

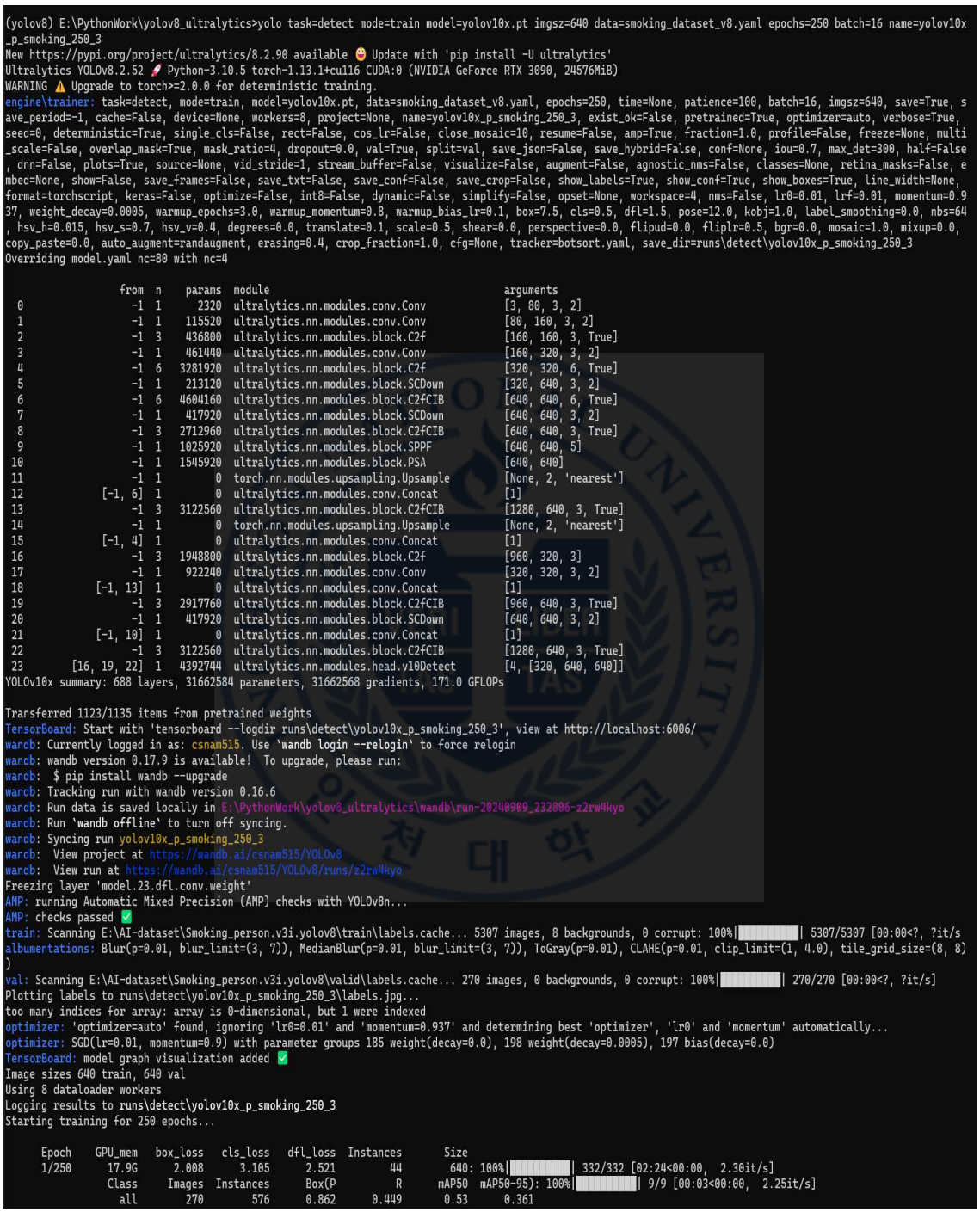
(yolov8) E:\PythonWork\yolov8_ultralytics>yolo task=detect mode=train model=yolov10x.pt imgsz=640 data=smoking_dataset_v8.yaml epochs=250 batch=16 name=yolov10x_p_smoking_250_3
New https://pypi.org/project/ultralytics/8.2.90 available 😃 Update with 'pip install -U ultralytics'
Ultralytics YOLOv8.2.52 🚀 Python-3.10.5 torch-1.13.1+cu116 CUDA:0 (NVIDIA GeForce RTX 3090, 24576MiB)
WARNING ⚠️ Upgrade to torch>=2.0.0 for deterministic training.
engine\trainer: task=detect, mode=train, model=yolov10x.pt, data=smoking_dataset_v8.yaml, epochs=250, time=None, patience=100, batch=16, imgsz=640, save=True, save_period=-1, cache=False, device=None, workers=8, project=None, name=yolov10x_p_smoking_250_3, exist_ok=False, pretrained=True, optimizer=auto, verbose=True, seed=0, deterministic=True, single_cls=False, rect=False, cos_lr=False, close_mosaic=10, resume=False, amp=True, fraction=1.0, profile=False, freeze=None, multi_scale=False, overlap_mask=True, mask_ratio=4, dropout=0.0, val=True, split=val, save_json=False, save_hybrid=False, conf=None, iou=0.7, max_det=300, half=False, dnn=False, plots=True, source=None, vid_stride=1, stream_buffer=False, visualize=False, augment=False, agnostic_nms=False, classes=None, retina_masks=False, embed=None, show=False, save_frames=False, save_txt=False, save_conf=False, save_crop=False, show_labels=True, show_conf=True, show_boxes=True, line_width=None, format=torchscript, keras=False, optimize=False, int8=False, dynamic=False, simplify=False, opset=None, workspace=4, nms=False, lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=7.5, cls=0.5, dfl=1.5, pose=12.0, kobj=1.0, label_smoothing=0.0, nbs=64, hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, degrees=0.0, translate=0.1, scale=0.5, shear=0.0, perspective=0.0, flipud=0.0, fliplr=0.5, bgr=0.0, mosaic=1.0, mixup=0.0, copy_paste=0.0, auto_augment=randaugment, erasing=0.4, crop_fraction=1.0, cfg=None, tracker=botsort.yaml, save_dir=runs\detect\yolov10x_p_smoking_250_3
Overriding model.yaml nc=80 with nc=4
from n params module arguments
0 -1 1 2320 ultralytics.nn.modules.conv.Conv [3, 80, 3, 2]
1 -1 1 115520 ultralytics.nn.modules.conv.Conv [80, 160, 3, 2]
2 -1 3 436800 ultralytics.nn.modules.block.C2f [160, 160, 3, True]
3 -1 1 461440 ultralytics.nn.modules.conv.Conv [160, 320, 3, 2]
4 -1 6 3281920 ultralytics.nn.modules.block.C2f [320, 320, 6, True]
5 -1 1 213120 ultralytics.nn.modules.block.SCDown [320, 640, 3, 2]
6 -1 6 4604160 ultralytics.nn.modules.block.C2fCIB [640, 640, 6, True]
7 -1 1 417920 ultralytics.nn.modules.block.SCDown [640, 640, 3, 2]
8 -1 3 2712960 ultralytics.nn.modules.block.C2fCIB [640, 640, 3, True]
9 -1 1 1025920 ultralytics.nn.modules.block.SPPF [640, 640, 5]
10 -1 1 1545920 ultralytics.nn.modules.block.PSA [640, 640]
11 -1 1 0 torch.nn.modules.upsampling.Upsample [None, 2, 'nearest']
12 [-1, 6] 1 0 ultralytics.nn.modules.conv.Concat [1]
13 -1 3 3122560 ultralytics.nn.modules.block.C2fCIB [1280, 640, 3, True]
14 -1 1 0 torch.nn.modules.upsampling.Upsample [None, 2, 'nearest']
15 [-1, 4] 1 0 ultralytics.nn.modules.conv.Concat [1]
16 -1 3 1948800 ultralytics.nn.modules.block.C2f [960, 320, 3]
17 -1 1 922240 ultralytics.nn.modules.conv.Conv [320, 320, 3, 2]
18 [-1, 13] 1 0 ultralytics.nn.modules.conv.Concat [1]
19 -1 3 2917760 ultralytics.nn.modules.block.C2fCIB [960, 640, 3, True]
20 -1 1 417920 ultralytics.nn.modules.block.SCDown [640, 640, 3, 2]
21 [-1, 10] 1 0 ultralytics.nn.modules.conv.Concat [1]
22 -1 3 3122560 ultralytics.nn.modules.block.C2fCIB [1280, 640, 3, True]
23 [16, 19, 22] 1 4392744 ultralytics.nn.modules.head.v10Detect [4, [320, 640, 640]]
YOLOv10x summary: 688 layers, 31662584 parameters, 31662568 gradients, 171.0 GFLOPs
Transferred 1123/1135 items from pretrained weights
TensorBoard: Start with 'tensorboard --logdir runs\detect\yolov10x_p_smoking_250_3', view at http://localhost:6006/
wandb: Currently logged in as: csnam515. Use `wandb login --relogin` to force relogin
wandb: wandb version 0.17.9 is available! To upgrade, please run:
wandb: $ pip install wandb --upgrade
wandb: Tracking run with wandb version 0.16.6
wandb: Run data is saved locally in E:\PythonWork\yolov8_ultralytics\wandb\run-20240909_232806-z2rw4kyo
wandb: Run `wandb offline` to turn off syncing.
wandb: Syncing run yolov10x_p_smoking_250_3
wandb: View project at https://wandb.ai/csnam515/YOLOv8
wandb: View run at https://wandb.ai/csnam515/YOLOv8/runs/z2rw4kyo
Freezing layer 'model.23.dfl.conv.weight'
AMP: running Automatic Mixed Precision (AMP) checks with YOLOv8n...
AMP: checks passed ✅
train: Scanning E:\AI-dataset\Smoking_person.v3i.yolov8\train\labels.cache... 5307 images, 8 backgrounds, 0 corrupt: 100%| 5307/5307 [00:00<?, ?it/s
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01, blur_limit=(3, 7)), ToGray(p=0.01), CLAHE(p=0.01, clip_limit=(1, 4.0), tile_grid_size=(8, 8))
val: Scanning E:\AI-dataset\Smoking_person.v3i.yolov8\valid\labels.cache... 270 images, 0 backgrounds, 0 corrupt: 100%| 270/270 [00:00<?, ?it/s]
Plotting labels to runs\detect\yolov10x_p_smoking_250_3\labels.jpg...
too many indices for array: array is 0-dimensional, but 1 were indexed
optimizer: 'optimizer=auto' found, ignoring 'lr0=0.01' and 'momentum=0.937' and determining best 'optimizer', 'lr0' and 'momentum' automatically...
optimizer: SGD(lr=0.01, momentum=0.9) with parameter groups 185 weight(decay=0.0), 198 weight(decay=0.0005), 197 bias(decay=0.0)
TensorBoard: model graph visualization added ✅
Image sizes 640 train, 640 val
Using 8 dataloader workers
Logging results to runs\detect\yolov10x_p_smoking_250_3
Starting training for 250 epochs...
Epoch GPU_mem box_loss cls_loss dfl_loss Instances Size
1/250 17.9G 2.008 3.105 2.521 44 640: 100%| 332/332 [02:24<00:00, 2.30it/s]
Class Images Instances Box(P R mAP50 mAP50-95): 100%| 9/9 [00:03<00:00, 2.25it/s]
all 270 576 0.862 0.449 0.53 0.361

그림 4.4 학습 진행 과정

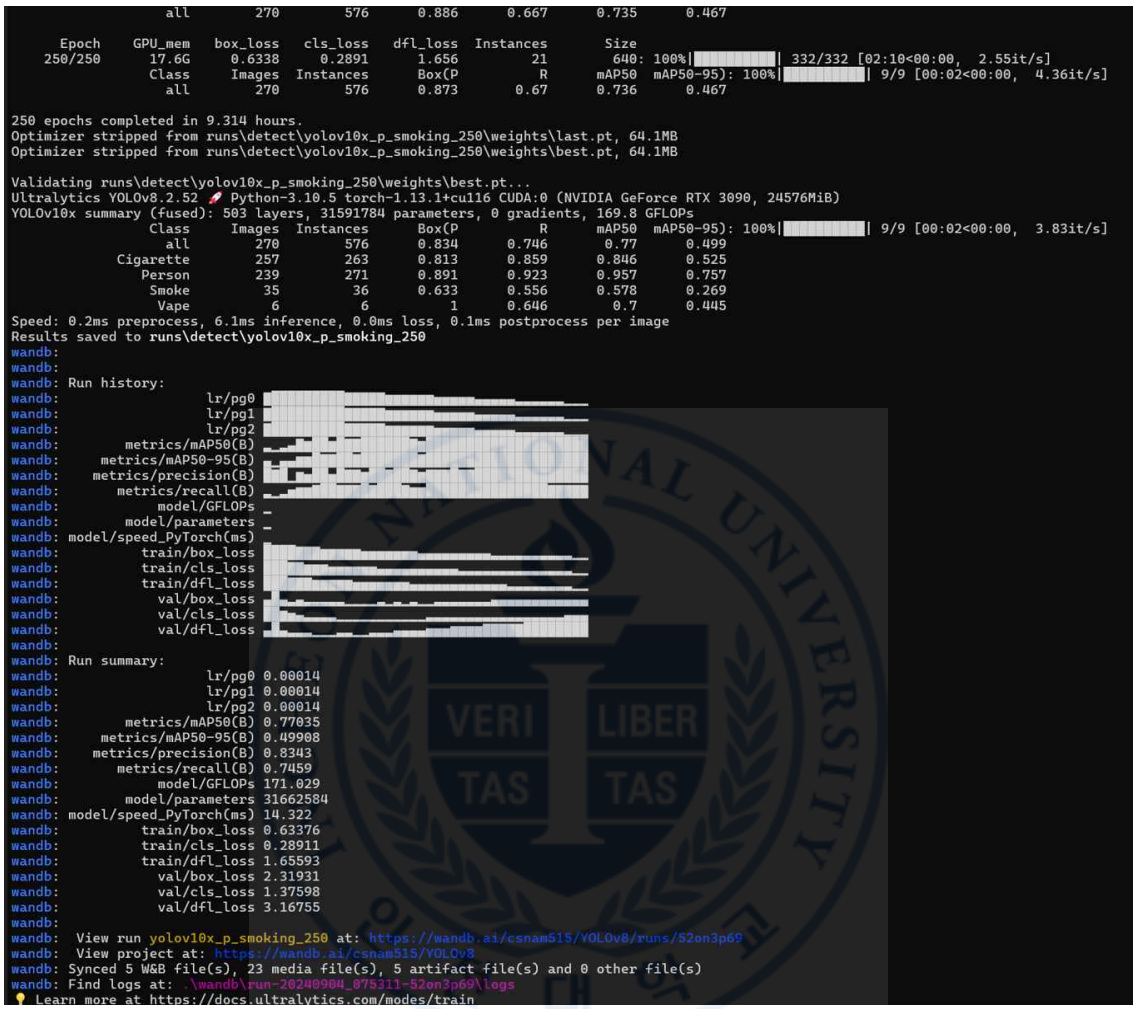

그림 4.5 학습 완료 결과 화면

4.3 실험 결과

4.3.1 실험 평가 도구

4.3.1.1 정밀도(precision), 재현율(recall)

물체 감지(Object Detection) 알고리즘의 성능은 정밀도-재현율 곡선(PR curve, precision-recall)과 평균 정밀도(AP, average precision)로 평가하는 것이 일반적이다. 정밀도(Precision)란 모델이

True라고 예측한 사례 중 실제로 True인 사례의 비율을 의미하며, 재현율은 실제로 True인 사례 중 모델이 True로 예측한 비율을 나타낸다. 두 지표는 수식 4.1 같은 식으로 표현된다.

$$Pecision = \frac{TP}{TP + FP} \quad Recall = \frac{TP}{TP + FN}$$

수식 4.1 정밀도, 재현율 공식

4.3.1.2 F1-점수(score)

F1-점수(score)는 정밀도와 재현율의 가중치를 각각 0.5씩 부여한 조화평균을 의미한다. 좋은 모델을 만들기 위해서는 각각의 지표를 적당히 최적화시키면서 조화를 찾는 게 중요하다. 수식적으로 표현하면 아래와 같다.

$$F1\,Score = 2 \times \frac{\mathrm{Precision} \times Recall}{\mathrm{Precision} + Recall}$$

수식 4.2 F1 Score 수식

4.3.1.3 PR 곡선 (precision-recall Curve)

PR 곡선(Precision-Recall Curve)은 정밀도(Precision)와 재현율(Recall) 사이의 관계를 시각화한 곡선으로, X축에 재현율, Y축에 정밀도를 배치하여 두 지표 간의 트레이드 오프(Trade-off)를 나타낸다. 이 곡선은 모델의 신뢰(confidence) 레벨에 따른 임계값(threshold) 변화를 바탕으로 물체 감지 모델의 성능을 평가하는 데 활용된다.

PR 곡선에서 오른쪽 위 모서리(1, 1) 점에 가까운 모델일수록 좋은 성능을 가진다. 이는 높은 재현율을 유지하면서도 정밀도가 높다는 의미로, 해당 모델이 탐지 누락 없이 정확한 예측을 수행함을 의미한다.

4.3.1.4 AP(Average Precision), mAP(mean Average Precision)

AP(Average Precision)은 Precision-Recall 곡선 아래의 면적을 나타내며, 모델 성능을 평가하는 데 사용되는 대표적인 지표로 모델의 성능을 하나의 숫자로 요약된다. 정밀도(Precision)와 재현율(Recall)은 트레이드 오프(trade-off) 관계이므로, AP는 이 정밀도와 재현율을 두 지표을 종합적으로 모두 고려하여 모델의 성능을 평가한다.

mAP(mean Average Precision)는 여러 클래스에 대한 AP의 평균값을 의미하며, 다중 클래스 분류 및 탐지 모델의 성능 평가에 사용됩니다. 각 클래스에 대한 AP를 계산한 후 그 평균을 취하여 모든 클래스에 걸쳐 모델의 성능을 종합적으로 평가한다.

4.3.1.5 IoU(Intersection over Union)

IoU(Intersection over Union)는 예측 영역 박스(predicted box)와 실제 영역 박스(Ground-truth box)의 겹친(overlapping) 영역에 비율을 말한다. IoU는 사물의 위치를 예측하는 척도이다.

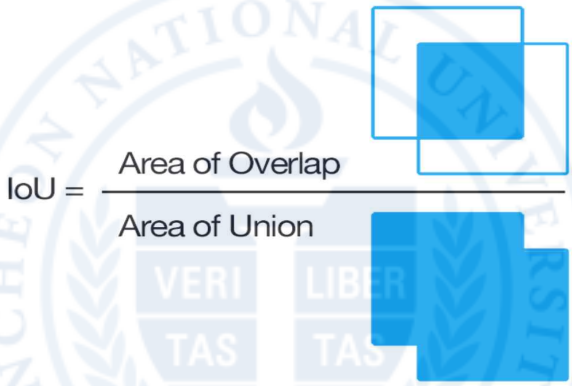

그림 4.6 IoU 개념도[47]

mAP@50은 IoU가 0.5일 때의 mAP를 의미한다. mAP@50:95 다양한 IoU 임계값(0.5부터 0.95까지)에서의 mAP를 평균 내는 방식으로, 이 경우 모델의 성능을 더 엄격하게 평가하게 된다.

4.3.2 실험 진행 방법

각 버전별 모델별 학습 진행에 사용된 자료 소스는 표 4.5에 표시하였다. 각 버전의 YOLO 공식 Github 소스와 YOLOv5, YOLOv8를 공개 발표한 ultralytics 회사에서 제공하는 명령줄 인터페이스(CLI, Command Line Interface)환경과 병행하여 진행하였다.

표 4.5 학습시 사용 소스 코드 위치

YOLO version	소스 위치(source location)
5	https://github.com/ultralytics/yolov5
6	https://github.com/ultralytics/ultralytics
7	https://github.com/WongKinYiu/yolov7.git
8	https://github.com/ultralytics/ultralytics
9	https://github.com/WongKinYiu/yolov9
10	https://github.com/ultralytics/ultralytics

4.3.3 실험 진행 결과표

버전별로 YOLOv5에서 YOLOv10까지, 각 버전 내에서 파라미터가 작은 n 모델부터 x 모델까지 실험 결과표는 다음 표와 같다. YOLOv9의 경우는 모델명이 다르긴 하지만 5단계로 나누어서 정리하였다. 표 4.6~표 4.11까지 내용을 분석하면 버전별 개발자들 사전 정의한 모델들의 파라미터 수과 레이어 수는 동일하지 않지만, 전반적으로 버전 내에서 파라미터 수가 큰 모델일수록 정확도나 재현율이 증가하고 있다.

그러나 YOLOv6 L모델의 경우는 학습 과정 중에 시스템이 다운되는 원인불명의 이유로 결과를 얻지 못하였다. 또한 YOLOv7 모델도 YOLOv7-d6, YOLOv7-e6e,YOLOv-W6 등 여러 모델이 있었으나 원인 불명의 에러가 발생하여 결과표에서는 표시하지 않았다.

표 4.6 YOLOv5 실험 결과

model	n	s	m	l	x
mAP@50	0.668	0.698	0.749	0.767	0.761
mAP@50:95	0.426	0.445	0.47	0.498	0.514
Precision	0.724	0.764	0.784	0.895	0.846
Recall	0.652	0.445	0.734	0.678	0.742
parameters	2,509,244	9,123,740	25,067,452	53,166,428	97,203,260
layers	262	262	339	416	493

표 4.7 YOLOv6 실험 결과

model	n	s	m	l	x
mAP@50	0.557	0.592	0.614	-	0.584
mAP@50:95	0.341	0.358	0.362	-	0.362
Precision	0.759	0.62	0.561	-	0.652
Recall	0.558	0.599	0.677	-	0.574
parameters	4,238,540	16,306,620	51,998,380	-	173,024,908
layers	195	195	273		351

표 4.8 YOLOv7 실험 결과

model	yolov7	yolov7-X
mAP@50	0.691	652
mAP@50:95	0.411	0.403
Precision	0.81	0.792
Recall	0.711	0.655
parameters	415	467
layers	37,622,682	70,835,306

표 4.9 YOLOv8 실험 결과

model	n	s	m	l	x
mAP@50	0.713	0.778	0.772	0.764	0.781
mAP@50:95	0.458	0.477	0.502	0.525	0.515
Precision	0.685	0.804	0.861	0.82	0.842
Recall	0.752	0.766	0.744	0.726	0.787
parameters	3,011,628	11,137,148	25,858,636	43,632,924	68,156,460
layers	225	225	295	365	365

표 4.10 YOLOv9 실험 결과

model	T	S	M	C	E
mAP@50	0.708	0.747	0.767	0.775	0.802
mAP@50:95	0.472	0.507	0.474	0.5	0.528
Precision	0.804	0.87	0.881	0.857	0.848
Recall	0.648	0.652	0.726	0.748	0.754
layers	917	917	603	384	1225
parameters	2,006,188	7,288,956	20,161,212	25,322,332	58,147,996

표 4.11 YOLOv10 실험 결과

model	n	s	m	l	x
mAP@50	0.631	0.647	0.779	0.795	0.770
mAP@50:95	0.418	0.419	0.481	0.509	0.499
Precision	0.709	0.724	0.809	0.873	0.834
Recall	0.574	0.592	0.724	0.754	0.746
parameters	2,708,600	8,069,448	16,488,744	25,722,536	31,662,584
layers	385	402	498	461	688

4.3.4 실험 결과물

4.3.4.1 예측 점수 화면

흡연 감지 결과를 시각적으로 비교하기 위해 성능지표가 가장 좋은 모델과 낮은 군에 속한 모델을 이미지를 추출하여 성능 비교하였다. 그림 4.7은 mAP@50 성능지표가 가장 좋은 YOLOv9 e모델과 지표가 결과가 낮은 YOLOv10 n모델의 이미지 결과이다. 그림 4.7에서 보듯이 YOLOv9 e모델에서는 Cigarette 점수(Score)가 전반적으로 높았으며 Vape 클래스도 찾지만, 비교 모델에서는 찾지 못했다.

그림 4.7 YOLOv9 e모델(위)과 YOLOv10 n모델(아래) 검증 화면

4.3.4.2 mAP@50 비교 그래프

각 버전의 모델들은 파라미터 수가 많을수록 성능이 좋아진다. 즉 파라미터 수에 따라서 평가지표 항목 점수가 비례하는 방향으로 올라간다. 비슷한 파라미터 수를 가진 몇 개의 버전의 모델을 비교 해 본다면 모델의 우수성을 비교할 수 있다. 표 4.12는 학습 파라미터가 비슷한 4가지 모델들의 mAP@50을 비교 결과이다. 비교 모델 경우 YOLOv10 l모델이 성능이 나은 결과가 나왔다.

표 4.12 비교군 모델 간의 파라미터 수, mAP@50 점수

model	parameters	mAP@50
YOLOv5 m	25,047,532	0.749
YOLOv8 m	25,842,076	0.772
YOLOv9 c	25,322,332	0.775
YOLOv10 l	25,722,536	0.795

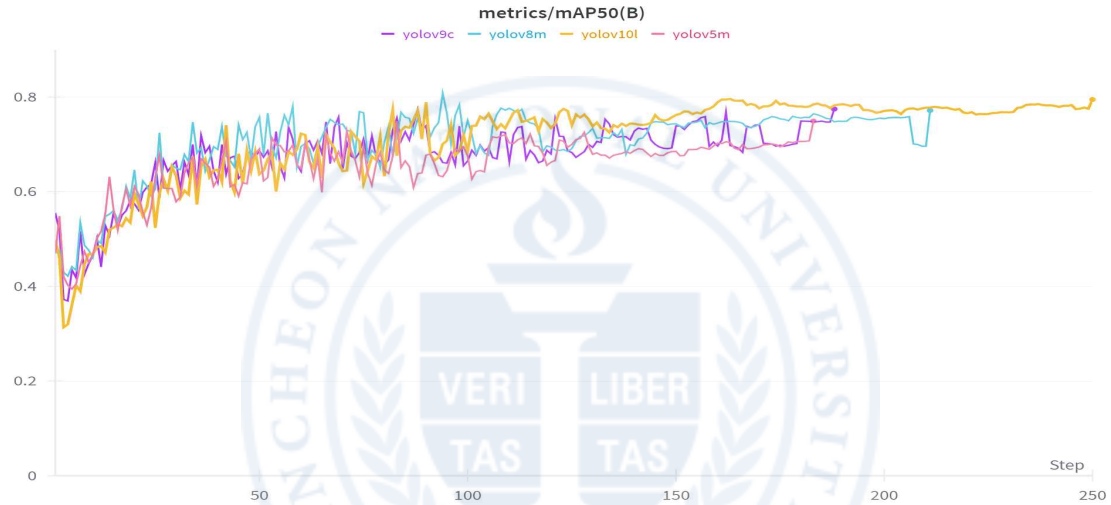

그림 4.8 파라미터수가 비슷한 모델간 mAP@50 비교

4.3.4.3 혼동 행렬(Confusion Matrix)

혼동행렬은 다중 클래스 분류 문제에서 각 클래스별로 모델의 성능을 평가하는데 유리하다. 그림 4.9는 성능지표가 좋은 Yolov9 e모델과 낮은 군에 속한 Yolov10 n모델에 대한 혼동 행렬이다. 두 모델 모두 Person과 Cigarette 클래스에 대해서 높은 성능이 나왔다. Vape와 Smoke 클래스에 대해서 성능이 낮게 관찰되었다

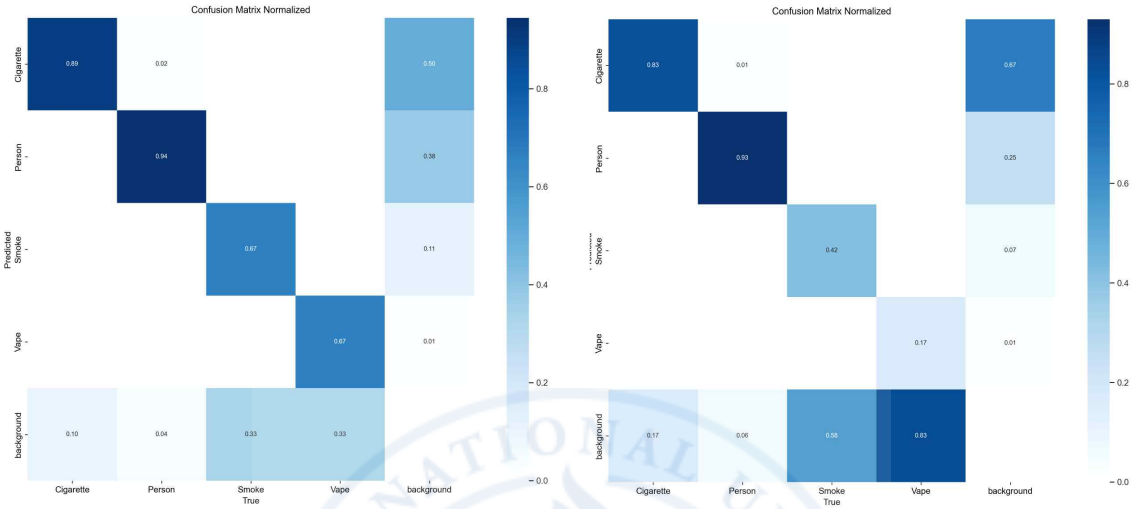

그림 4.9 Yolov9e(좌)과 Yolov10n(우) 혼동 행렬 비교

4.3.4.4 PR(precision-recall) 곡선

YOLOv9e 모델은 Cigarette, Person, Smoke 클래스 등 모든 클래스에 대해 비교적 우수한 성능 지표를 보였다. 반면, YOLOv10n 모델은 Person 클래스에서 YOLOv9e 모델과 유사한 성능을 보였으나, Cigarette와 Vape 클래스에서는 상대적으로 낮은 점수를 기록하였다. 특히, YOLOv10n 모델은 크기가 작은 물체를 감지하는 능력이 부족한 것으로 판단되며, Vape 클래스에 대한 성능은 0.207로 매우 저조한 결과를 나타냈다.

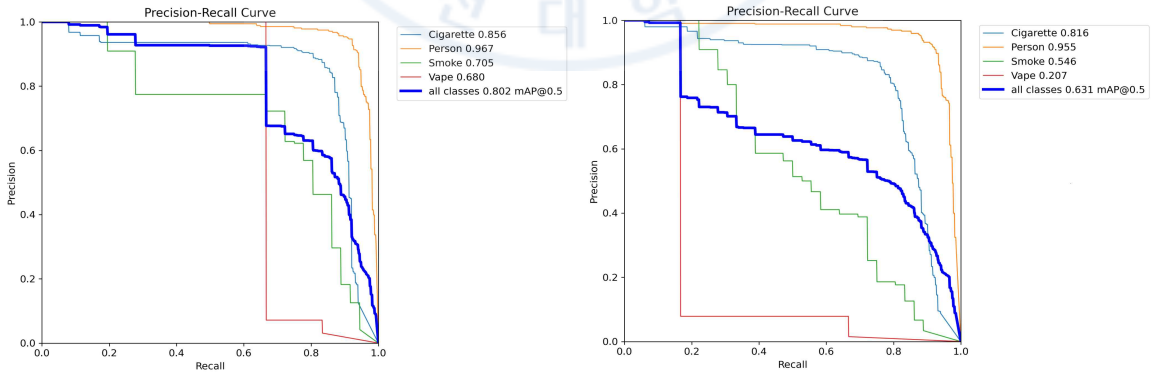

그림 4.10 YOLOv9e(좌)과 Yolov10n(우) PR 곡선 비교

4.4 실험 분석

4.4.1 YOLO 버전들간의 mAP 성능 비교

그림 4.11은 실험한 YOLO 버전 및 모델별 mAP 기준으로 성능을 비교한 그래프이다. 전반적으로 파라미터 수가 큰 모델일수록 성능이 향상되는 경향을 보였다. mAP@50 기준으로 YOLOv9e 모델이 가장 높은 성능을 달성하였으며, 그 뒤를 YOLOv10L 모델과 YOLOv8x 모델이 따랐다. YOLOv6n 모델은 가장 낮은 성능을 기록하였다.

특이할 만한 점은 YOLOv10x 모델이 YOLOv10l 모델보다 더 많은 파라미터를 가지고 있음에도 불구하고, 성능은 상대적으로 낮게 나타났다. 이러한 결과는 YOLOv10 모델에서의 에폭 수가 250으로 충분하지 않거나, 실험에 사용된 학습 데이터에 대한 과적합(Overfitting)으로 인해 발생한 현상으로 추측된다.

version	model	mAP50
YOLOv5	n	0.668
	s	0.698
	m	0.749
	l	0.767
	x	0.761
YOLOv6	n	0.557
	s	0.592
	m	0.614
	x	0.584
YOLOv7	7	0.691
	7x	0.652
YOLOv8	n	0.713
	s	0.778
	m	0.772
	l	0.764
	x	0.781
YOLOv9	T	0.708
	S	0.747
	M	0.767
	C	0.775
	E	0.802
YOLOv10	n	0.631
	s	0.647
	m	0.779
	l	0.795
	x	0.77

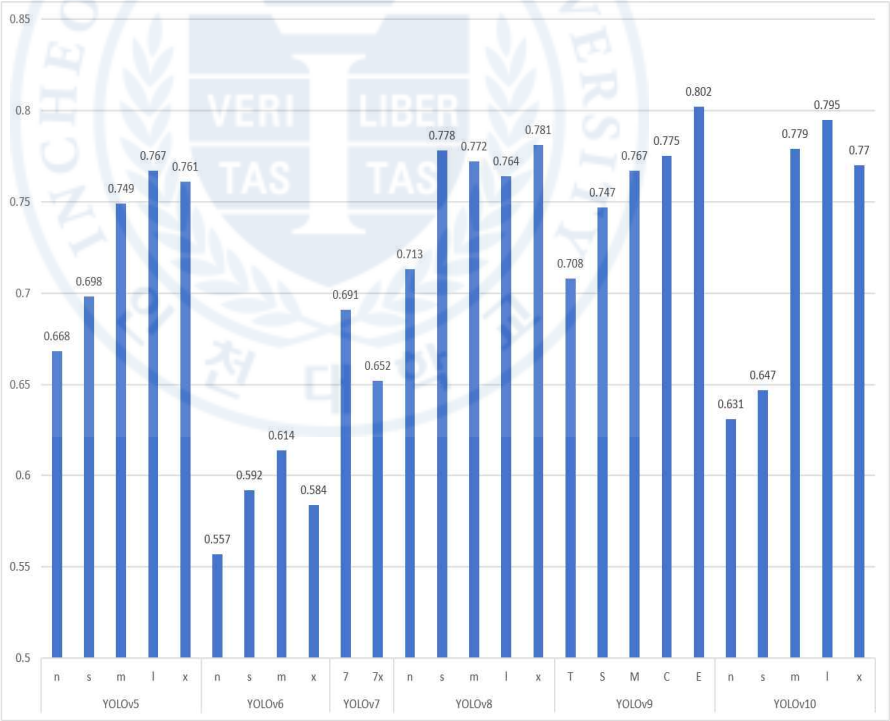

그림 4.11 모델간 mAP@50 성능 비교

4.4.2 YOLO 버전들간의 mAP 파라미터수 비교

그림 4.12는 실험한 YOLO 버전별, 모델별 파라미터 수를 비교한 그림이다. 파라미터수는 모델의 성능과 복잡성에 관련 되어있다. 파라미터 수가 많을수록 성능은 대체로 증가한다. 특이사항은 YOLOv6는 다른 버전에 비해 전반적으로 많은 학습 파라미터를 가지고 있으며, 특히 YOLOv6는 설계될 당시 고성능을 요구하는 산업용 애플리케이션에 최적화되어 단일 GPU 중심의 학습 환경을 고려했기 때문으로 추측된다.

반면, YOLOv10은 효율성 중심의 모델 설계가 특징이며, 이에 따라 학습 파라미터대비 mAP@50 평가지표는 비교군 모델보다 우수한 성능을 보인다. 특히 YOLOv8과 YOLOv10의 m, l 모델을 비교했을 때, YOLOv10 모델은 성능 면에서 유사하거나 더 우수한 결과를 나타냈으나, YOLOv8 모델은 YOLOv10 모델보다 50% 이상 더 많은 파라미터를 가지고 있다. 이는 YOLOv10 모델이 상대적으로 간결한 구조로 되어 있어 학습 시 필요한 연산량과 메모리 소모가 적다는 점을 의미한다.

표 4.13 mAP 성능 대비 파라미터 수 비교

Model	item	YOLOv8	YOLOv10	Compare
m	mAP@50	0.772	0.779	0.007
	parameters	25,858,636	16,488,744	56.8(%)
l	mAP@50	0.764	0.795	0.031
	parameters	43,632,924	25,771,480	69.3(%)

Version	Model	Parameters
YOLOv5	n	2,509,244
	s	9,123,740
	m	25,067,452
	l	53,166,428
	x	97,203,260
YOLOv6	n	4,234,140
	s	16,298,204
	m	51,979,804
	x	172,985,244
YOLOv7	7	37,622,682
	7X	70,835,306
YOLOv8	n	3,011,628
	s	11,137,148
	m	25,858,636
	l	43,632,924
	x	68,156,460
YOLOv9	T	2,006,188
	S	7,288,956
	M	20,161,212
	C	25,322,332
	E	58,147,996
YOLOv10	n	2,708,600
	s	8,069,448
	m	16,488,744
	l	25,722,536
	x	31,662,584

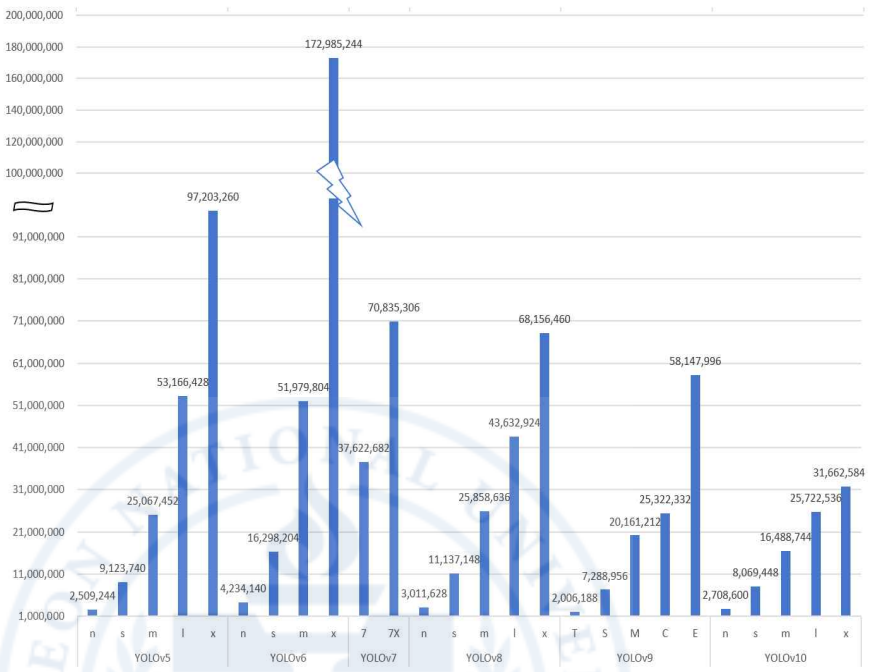

그림 4.12 YOLO 모델 간 파라미터 수 비교

4.4.3 YOLO 버전들간의 추론 시간 비교

본 추론 시간(Inference time) 테스트에서는 동일 데이터에 대해 추론 시간을 수회 측정하였으나 수행 시마다 약 최대 1.5ms 이내로 차이가 있었지만, 모델 간 상대적인 추론 시간의 방향성 같았다. YOLOv5와 YOLOv8의 n 모델이 약 8.4에서 8.6 ms까지의 짧은 추론 시간을 기록하였다. 반면, 추론 시간이 가장 오래 걸리는 모델은 버전별로 가장 큰 모델들이었으며, 이들 모델의 추론 시간은 약 15.1에서 29.6 ms까지로 측정되었다.

이미지 당 추론 시간은 일반적으로 작은 모델에서 큰 모델로 갈수록 증가하는 경향을 보였으나, YOLOv9의 경우 s 모델보다 m 및 c 모델이 더 빠른 추론 속도를 보인 점이 특이했다. YOLOv9e 모델은 본 연구에서 mAP@50이 0.82로 가장 높은 성능을 기록했으며, 가장 많은 파라미터를 사용하였으나 그에 대한 영향으로 추론 시간은 다른 모델에 비해 현저하게 느린 29.6 ms로 측정되었다. YOLOv10의 경우는

파라미터가 큰 모델과 작은 모델 사이에 추론 속도는 약 0.3 ms 정도로 다른 YOLO 버전에 비해 상대적으로 낮게 관찰되었다.

version	model	infer(ms)
Yolov5	n	8.3
	s	8.9
	m	11.4
	l	13.6
	x	17.1
Yolov8	n	8.6
	s	9
	m	10.9
	l	12.2
	x	15.6
Yolov9	t	18.2
	s	21
	m	17.2
	c	17.3
	e	29.6
Yolov10	n	11.3
	s	11.1
	m	13
	l	14.8
	x	15.1

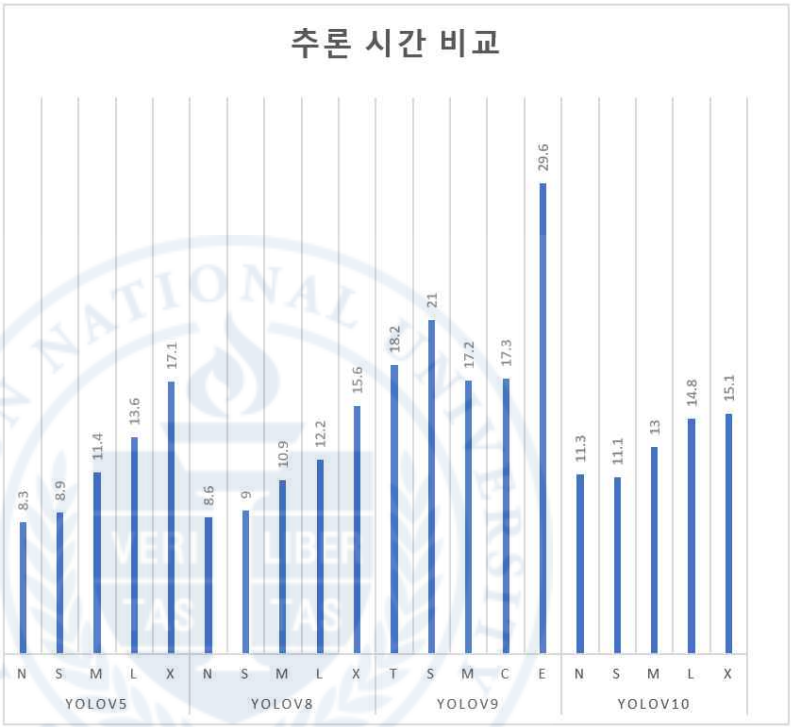

그림 4.13 YOLO 추론 시간 비교

4.4.4 전이 학습과 스크래치 학습과의 비교

그림 4.14는 COCO 데이터 세트에서 학습된 사전 가중치를 이용한 모델과 사전 가중치를 사용하지 않고 처음부터 학습한 스크래치 학습(Training from Scratch)을 진행한 모델 간의 mAP@50 평가지표 비교 결과이다. 사전 학습된 가중치를 활용한 전이 학습(Transfer Learning) 방법의 효율성은 모델에 따라 차이가 있었지만, 스크래치 학습 대비 9%~28% 정도 mAP@50 지표가 높게 나타났다. 전반적으로 전이 학습을 적용한 경우 평균적으로 15.4% 더 나은 성능을 보였다. 이것은 흡연 데이터 수가 적기 때문에, 더 많은 클래스와 학습 데이터를 훈련된 지식을 활용할 경우 물체

감지 능력이 향상될 수 있음을 시사한다.

version	model	scratch lr	tranfer lr	increase(%)
YOLOv5	n	0.61	0.668	9.5
	s	0.622	0.698	12.2
	m	0.641	0.749	16.8
	l	0.636	0.767	20.6
	x	0.664	0.761	14.6
YOLOv8	n	0.61	0.713	16.9
	s	0.622	0.778	25.1
	m	0.641	0.772	20.4
	l	0.636	0.764	20.1
	x	0.664	0.781	17.6
YOLOv9	T	0.648	0.708	9.3
	S	0.682	0.747	9.5
	M	0.716	0.767	7.1
	C	0.695	0.775	11.5
	E	0.74	0.802	8.4
YOLOv10	n	0.557	0.631	13.3
	s	0.582	0.647	11.2
	m	0.651	0.779	19.7
	l	0.62	0.795	28.2
	x	0.665	0.77	15.8

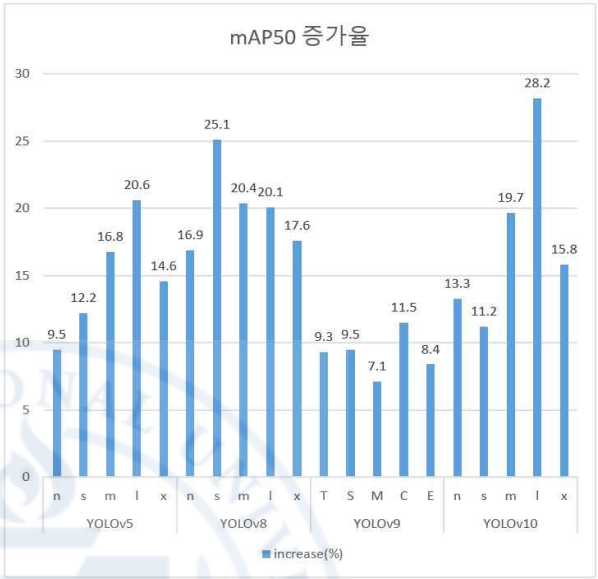

그림 4.14 전이 학습시 AP 증가율

4.4.5 배치 크기 변화에 따른 성능지표 변화

실험 과정 중 원인불명의 이유로 학습 에러가 발생하여 직관적으로 배치사이즈를 조정하여 해결했다. 그러나 배치 크기(batch size) 조정 후 mAP 점수가 달라지는 현상을 통해 배치 크기가 성능에 영향을 줄 수 있음을 확인하였다. 이에 따라 배치 크기 변경이 모델 성능에 미치는 영향을 분석하였다. 특히 YOLOv8의 n 모델을 대상으로 배치 크기 변화에 따른 mAP@50 성능지표를 비교하였다.

실험 결과, 배치 크기와 mAP@50간에는 명확한 비례 관계나 반비례 관계가 나타나지 않았다. 기본 설정값인 배치 크기 16에서 양호한 성능을 보였지만, 가장 우수한 결과는 배치 크기 값이 8일 때 도출되었다. 배치 크기를 증가시키거나 감소시켰을 때, mAP@50 지표는 일부 경우 감소하거나, 일부 경우 증가하는 패턴을 보였다. 이러한 결과는 배치 크기가 고정적인 최적값을 가지기보다는, 반복적인 실험과 점진적인 조정 과정을 통해 최적값을 찾아야 할 필요성을 시사한다.

batch	mAP50
4	0.688
8	0.715
16	0.713
32	0.696
64	0.702

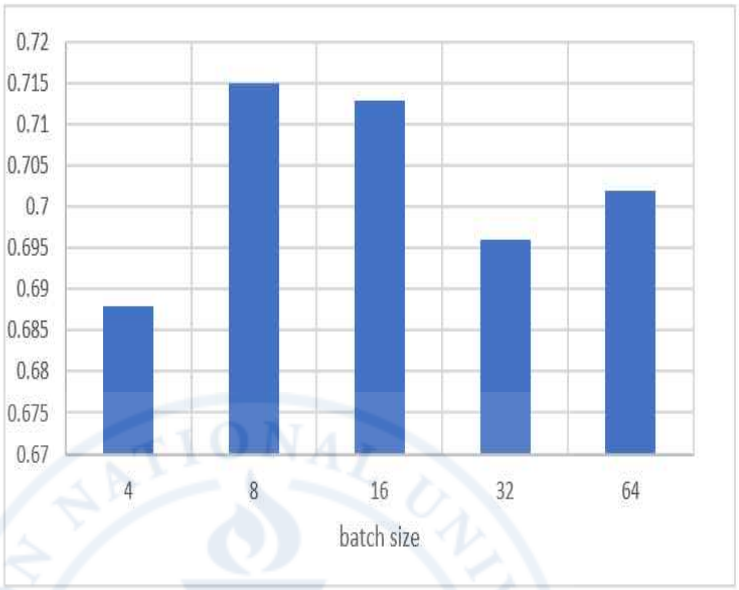

그림 4.15 YOLOv8의 n모델에서 배치 크기 별 mAP@50 변화

4.4.6 데이터 증강 옵션에 따른 성능지표 변화

데이터 증강(data augmentation)은 제한된 데이터로도 효율적인 학습을 가능하게 하기 위한 방법으로 다양한 증강 기법들이 사용된다. 본 연구에서는 모자이크(Mosaic) 증강 기법 적용 옵션에 따른 성능 변화를 분석하였다. 모자이크(Mosaic)는 네 개의 이미지를 크기와 위치를 조정하여 새로운 이미지로 결합하는 방법으로, 증강 강도는 0.0에서 1.0까지의 범위로 설정할 수 있다.

실험 결과, 그림 4.16과 같이 모자이크값을 증감함에 따라 mAP@50 지표의 변화가 비례 관계로 나타나지 않으나 모자이크 데이터 증강 기법을 사용했을 때 성능이 향상되었다. 특히 모자이크값을 최대로 설정하였을 때 약 5% 더 높은 성능이 도출되었다. 이러한 결과는 데이터 증강의 효과가 흡연 데이터에서도 모델 성능을 효과적으로 향상할 수 있음을 보여준다.

mosaic	mAP50
0	0.673
0.3	0.651
0.5	0.703
0.7	0.696
1	0.713

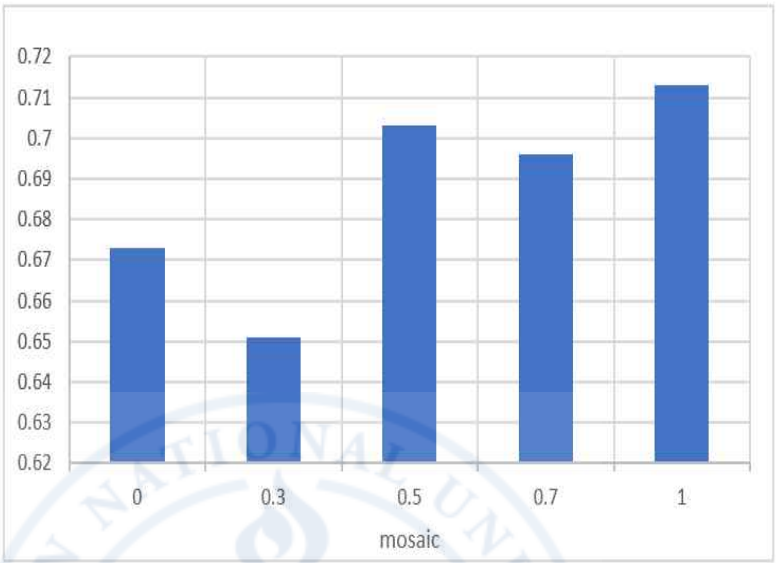

그림 4.16 YOLOv8의 n모델에서 mosaic변화에 따른 mAP@50 변화

제 5 장 결 론

5.1 종합 의견

본 연구는 흡연 감지에 대한 딥러닝 모델의 성능 분석에 중점을 두고 진행되었다. 실시간 담배 및 연기 감지를 위해 관련 이미지 데이터 셋을 수집하고, YOLOv5부터 YOLOv10까지 다양한 버전의 모델을 학습 및 평가하였다. 각 모델의 성능을 검증하기 위해 여러 차례 반복 실험을 수행하였으며, 이를 통해 어떤 모델이 가장 우수한 성능을 보이는지 분석하였다.

실험 결과, YOLOv5 이후에 발표된 최신 YOLO 버전들은 초기 버전에 비해 모두 높은 성능을 기록하였다. 각 버전 내에서 모델 크기에 따른 성능 비교에서는, 모델의 파라미터 수가 증가할수록 평균 정밀도(AP) 값도 함께 증가하는 경향을 보였다. 특히, YOLOv9의 e 모델은 mAP@50 기준으로 비교 모델 중 가장 우수한 성능을 나타냈다.

YOLOv5와 YOLOv8의 성능을 비교한 결과, n, s, m, l, x 모델별로 파라미터 수가 유사한 범위 내에 있다고 간주할 때, 전반적으로 YOLOv8이 더 우수한 AP 평가 지표를 기록하였다. 특히, YOLOv8의 x 모델은 YOLOv5의 x 모델보다 약 1.5배 적은 파라미터 수에도 불구하고, mAP@50에서 더 높은 성능을 보였다. 이는 이전 버전의 C3 모듈 구조를 C2f 모듈로 변경하고, 앵커 프리(Anchor-free) 구조의 헤드(head)를 채택한 점이 작은 데이터 형태 학습에 적합했기 때문으로 판단된다.

최근 발표된 YOLOv8, YOLOv9, YOLOv10 모델 중에 파라미터 수가 약 25만 개로 유사한 YOLOv8의 m 모델, YOLOv9의 c 모델, YOLOv10의 l 모델을 선정하여 비교한 결과, YOLOv10의 l 모델이 가장 높은 성능을 기록하였다.

그러나 제일 작은 경량 모델인 n 모델 성능에서는 YOLOv8이 가장 우수했으며, 그 뒤를 YOLOv9과 YOLOv10이 이었다. 이를 통해, 최신 버전의 모델이 항상 더 우수한 성능을 보장하는 것이 아니라는 점도 확인되었다.

실시간 흡연 감지 시스템은 저전력 및 제한된 메모리 환경에서 작동될 가능성이

높으므로, 파라미터 수가 적은 모델이 유리할 수 있다. 이러한 점에서, 적은 파라미터 수와 높은 성능을 동시에 갖춘 YOLOv9의 t모델은 제한된 하드웨어 자원을 가진 임베디드 환경에 적합한 선택지이다. YOLOv9에서 소개한 Generalized Efficient Layer Aggregation Network (GELAN) 구조는 모델의 적은 수의 파라미터를 가진 경량화와 추론 속도 증가, 정확성 개선에 기여한 것으로 판단된다.

또한, 추론 속도가 중요한 경우에는, 높은 AP값과 빠른 추론 속도를 갖춘 YOLO v8의 n 모델이 실용적인 선택이 될 수 있다. 이것은 Yolov8의 앵커 프리 방식 채택이 계산량이 줄어들어 속도가 증가했기 때문으로 판단된다.

5.2 성과, 한계 및 향후 연구 과제

평가지표 결과는 데이터 세트 보강, 학습률(learning rate), 배치 크기 등의 하이퍼파라미터 조정에 따라 달라질 수 있다. 연구에 활용한 데이터 세트의 클래스는 담배, 연기, 사람, 전자담배로 구성되었으나, 흡연 행동을 감지하기 위해 목적에 따라 다른 클래스(예: 얼굴, 손, 입)가 추가되거나 항목이 삭제될 수 있다.

다른 클래스 구성을 가진 공개 흡연 데이터 세트에서 YOLO 모델의 성능을 비교한 결과, 본 실험의 결과와 일치하지 않는 경우가 있었다. 또한, YOLO 버전별 성능 점수가 크게 나타나지 않는 실험 결과를 통해, 동일한 데이터 세트를 사용하더라도 배치 크기와 같은 하이퍼파라미터의 조정만으로 연구 결과가 달라질 수 있음을 확인하였다.

아울러, 인터넷에서 검증없이 공개된 흡연 행위 데이터 세트의 경우 특정 클래스에 편향되거나 일부 클래스의 데이터가 상대적으로 부족한 경우 등 데이터 불균형 문제가 존재할 가능성이 있다. 보다 정확하고 객관적인 결론을 도출하기 위해서는 다양한 데이터 세트를 활용한 실험과 다양한 조건을 검증하여 통계 자료를 추가하고 보완할 필요가 있다.

학습 데이터의 질, 클래스 간 데이터 분포, 하이퍼파라미터 조정, 모델 구조 등은 학습 결과에 영향을 미치는 다양한 요소들이다. 딥러닝은 수학적 알고리즘에 기반하지만, 결과를 해석하거나 성능을 최적화하는 과정에서 명확히 정의된 이론만으

로 문제를 해결하기 어려운 경우가 많다. 이로 인해 특정 영역에서의 물체 감지 성능을 향상시키는 작업은 연구자의 경험과 반복 실험에 크게 의존하게 된다. 이러한 특성은 최근 설명 가능한 AI(XAI)에 대한 연구가 활발히 이루어지고 있는 이유 중 하나로 판단된다.

실험 과정에서는 담배 모양의 볼펜과 같은 유사한 물체를 담배로 오인하는 사례가 발견되었으며, 이는 모델의 성능 향상을 위한 추가적인 대책이 필요함을 시사하였다. 특히, 비 오는 날이나 야간과 같은 악조건의 데이터를 학습시켜 탐지 성능을 개선한다면 다양한 실제 환경에서 나은 정확한 흡연 감지에 기여할 것으로 보인다.

또한, 흡연 감지 성능을 더 높이기 위해서는 영상 내 인물의 위치 및 자세를 담배의 위치와 연계하여 분석하는 후처리 기법을 도입하는 것도 고려할 만하다. 실생활에서 활용할 수 있는 보다 정확한 탐지 결과를 도출할 수 있을 것으로 예상되며, 실시간 흡연 감지 시스템의 실효성을 더욱 높일 수 있을 것이다.

본 실험 결과와 분석은 데이터 세트와 다양한 하이퍼파라미터 조정에 따라 상이한 결과가 나올 수 있다. 그럼에도 불구하구 흡연 행위 탐지와 더불어 물체 인식 감지 분야에 참고 자료 활용되어, 향후 영상 딥러닝 기술의 발전에 기여되기를 바란다.

참 고 문 헌

[1] Zhang, D., Jiao, C. and Wang, S. (2018). "Smoking Image Detection Based on Convolutional Neural Networks." 2018 IEEE 4th International Conference on Computer and Communications (ICCC), Computer and Communications (ICCC), 2018 IEEE 4th International Conference on, pp. 1509 – 1515.

[2] Macalisang, J. R., Merencilla, N. E., Ligayo, M. A. D., Melegrito, M. P. and Tejada, R. R. (2020). "Eye-Smoker: A Machine Vision-Based Nose Inference System of Cigarette Smoking Detection using Convolutional Neural Network." 2020 IEEE 7th International Conference on Engineering Technologies and Applied Sciences (ICETAS), Engineering Technologies and Applied Sciences (ICETAS) 2020 IEEE 7th International Conference on, pp. 1 – 5.

[3] Tang, Liu, Zheng, Zhang, Wang and Yang, 2021-07-09, "Smoking Behavior Detection Based On Improved YOLOv5s Algorithm." , pp. 1.

[4] Zhang, Chen, Xiao and Li (2021). "Research on Smoking Detection Based on Deep Learning." Journal of Physics: Conference Series, 2024(1).

[5] Ma, Y., Yang, J., Li, Z. and Ma, Z., 2022, "YOLO-Cigarette: An effective YOLO Network for outdoor smoking Real-time Object Detection." , pp. 121 – 126.

[6] Li, J. (2022). "Study on Smoking and Telephone Behaviour Detection Based on Convolutional Neural Network." 2022 International Conference on Machine Learning and Intelligent Systems Engineering (MLISE), Machine Learning and Intelligent Systems Engineering (MLISE), 2022 International Conference on, MLISE, pp. 415 – 418.

[7] C. Wang, T. Zheng, F. Sun and H. Liu, 2023, "A Smoking Detection Algorithm Based on Improved YOLOV5." 2023 IEEE 3rd International Conference on Power, Electro

nics and Computer Applications (ICPECA), pp. 1038–1043.

[8] Peng, J., Wang, C., Li, Y. and Chen, H. (2023). "Substation Personnel Smoking Detection Based On GhostNetV2-YOLOv5." 2023 6th International Symposium on Autonomous Systems (ISAS), Autonomous Systems (ISAS), 2023 6th International Symposium on, pp. 1–6.

[9] Aditya, Gudpati, Reddy, Shu and Karampudi (2023). "Smoking Detection using Deep Learning." International Journal of Computer Trends and Technology, 71(02), pp. 8.

[10] Lakatos, R., Pollner, P., Hajdu, A. and Joo, T. (2023). "A multimodal deep learning architecture for smoking detection with a small data approach." .

[11] 김동준, 최유진, 박경민, et al. (2023). "딥러닝 기술을 이용한 영상에서 흡연행위 검출." 한국인터넷방송통신학회 논문지, 23(4), pp. 107–113.

[12] Wang, Z., Liu, Y., Lei, L. and Shi, P. (2024). "Smoking-YOLOv8: a novel smoking detection algorithm for chemical plant personnel." Pattern Analysis and Applications, 27(3).

[13] Zou, Z., Chen, K., Shi, Z., Guo, Y. and Ye, J."Object Detection in 20 Years: A Survey." .

[14] Lin, T., Dollar, P., Girshick, R., He, K., Hariharan, B. and Belongie, S. (2017). "Feature Pyramid Networks for Object Detection." 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, CVPR, pp. 936–944.

[15] Liu, S., Qi, L., Qin, H., Shi, J. and Jia, J. (2018). "Path Aggregation Network for Instance Segmentation." 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Computer Vision and Pattern Recognition (CVPR), 2018 IEEE/CVF Conference on, CVPR, pp. 8759–8768.

[16] Wang, Z., Xie, K., Zhang, X., Chen, H., Wen, C. and He, J. (2021). "Small-Objec

t Detection Based on YOLO and Dense Block via Image Super-Resolution." IEEE Acce ss, Access, IEEE, 9, pp. 56416 – 56429.

[17] MathWorks "Anchor Boxes for Object Detection." https://www.mathworks.com/help/v ision/ug/anchor-boxes-for-object-detection.html.

[18] Tian, Z., Shen, C., Chen, H. and He, T. (2019). "FCOS: Fully Convolutional One-Stage Object Detection." 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Computer Vision (ICCV), 2019 IEEE/CVF International Conference on, pp. 962 6 – 9635.

[19] Tian, Z., Shen, C., Chen, H. and He, T. (2022). "FCOS: A Simple and Strong An chor-Free Object Detector." IEEE Transactions on Pattern Analysis and Machine Intellige nce, Pattern Analysis and Machine Intelligence, IEEE Transactions on, IEEE Trans.Patter n Anal.Mach.Intell., 44(4), pp. 1922 – 1933.

[20] Law, H. and Deng, J. (2018). "CornerNet: Detecting Objects as Paired Keypoints." .

[21] He, K., Zhang, X., Ren, S. and Sun, J. (2016). "Deep Residual Learning for Image Recognition." 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVP R), Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on, pp. 7 70 – 778.

[22] He, K., Zhang, X., Ren, S. and Sun, J. (2014). "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition." .

[23] 고명진, 박민주, 여지호, Go, M., Park, M. and Yeo, J. (2022). "Faster R-CNN을 이용한 갓길 차로 위반 차량 검출." 한국ITS학회논문지, 21(1), pp. 105 – 122.

[24] Wang, C., Mark Liao, H., Wu, Y., Chen, P., Hsieh, J. and Yeh, I. (2020). "CSPN et: A New Backbone that can Enhance Learning Capability of CNN." 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Comput

er Vision and Pattern Recognition Workshops (CVPRW),2020 IEEE/CVF Conference on, pp. 1571 – 1580.

[25] Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. (2016). "You Only Look Once: Unified, Real-Time Object Detection." 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on, pp. 779 – 788.

[26] deepsystems.io "yolo review." https://docs.google.com/presentation/d/1aeRvtKG21KHdD5lg6Hgyhx5rPq_ZOsGjG5rJ1HP7BbA/pub?start=false&loop=false&delayms=3000&slide=id.p.

[27] Horvat, M., Jelečević, L. and Gledec, G. (2023). "Comparative Analysis of YOLOv5 and YOLOv6 Models Performance for Object Classification on Open Infrastructure: Insights and Recommendations." Central European Conference on Information & Intelligent Systems, pp. 317 – 324.

[28] Yusof, N. ‘ M., Sophian, A., Zaki, H. F. M., Embong, A. H., Bawono, A. A. and Ashraf, A. (2024). "Assessing the performance of YOLOv5, YOLOv6, and YOLOv7 in road defect detection and classification: a comparative study." Bulletin of Electrical Engineering and Informatics, 13(1), pp. 350.

[29] Terven, J. and Cordova-Esparza, D. (2023). "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS." .

[30] Redmon, J. and Farhadi, A. (2017). "YOLO9000: Better, Faster, Stronger." 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, CVPR, pp. 6517 – 6525.

[31] Redmon, J. and Farhadi, A. (2018). "YOLOv3: An Incremental Improvement." .

[32] Bochkovskiy, A., Wang, C. and Liao, H. M. (2020). "YOLOv4: Optimal Speed and Accuracy of Object Detection." .

[33] Zhou, F., Deng, H., Xu, Q. and Lan, X. (2023). "CNTR-YOLO: Improved YOLOv 5 Based on ConvNext and Transformer for Aircraft Detection in Remote Sensing Image s." Electronics (Switzerland), 12(12).

[34] Li, C., Li, L., Jiang, H., et al. (2022). "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications." .

[35] Xu, T. -. and Liu, C. -. (2019). "Data-distortion guided self-distillation for deep ne ural networks." 33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Inno vative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, pp. 5565.

[36] Ultralytics "Meituan YOLOv6." https://docs.ultralytics.com/models/yolov6/#overview.

[37] Li, C., Li, L., Geng, Y., et al. (2023). "YOLOv6 v3.0: A Full-Scale Reloading." .

[38] Wang, C., Bochkovskiy, A. and Liao, H. M., 2022-07-06, "YOLOv7: Trainable bag -of-freebies sets new state-of-the-art for real-time object detectors." .

[39] Chen, X., Pu, H., He, Y., et al. (2023). "An Efficient Method for Monitoring Bird s Based on Object Detection and Multi-Object Tracking Networks." Animals (2076-261 5), 13(10), pp. 1713–1739.

[40] Ngoc, H. T., Nguyen, K. H., Hua, H. K., Nguyen, H. V. N. and Quach, L. -. (20 23). "Optimizing YOLO Performance for Traffic Light Detection and End-to-End Steerin g Control for Autonomous Vehicles in Gazebo-ROS2." International Journal of Advanced Computer Science and Applications, 14(7), pp. 475.

[41] OpenMMLab "YOLOv8." https://github.com/open-mmlab/mmyolo/tree/main/configs/yol ov8.

[42] Anonymous "yolov8 architecture." https://yolov8.org/yolov8-architecture/.

[43] Wang, C., Yeh, I. and Liao, H. M. (2024). "YOLOv9: Learning What You Want t o Learn Using Programmable Gradient Information." .

[44] Wang, A., Chen, H., Liu, L., et al. (2024). "YOLOv10: Real-Time End-to-End Obj ect Detection." .

[45] Alif, M. A. R. and Hussain, M. (2024). "YOLOv1 to YOLOv10: A comprehensive review of YOLO variants and their application in the agricultural domain." .

[46] visionwork (2022). "smoking_person_dataset." Roboflow Universe, Open Source Dat aset, https://universe.roboflow.com/visionwork/smoking_person.

[47] Adrian Rosebrock (2016). "Intersection over Union (IoU) for object detection." http s://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/.

ABSTRACT

A Comparative Study on Smoking Detection Performance of YOLO-based Deep Learning Models

Nam, Chun Shik
Department of Computer Engineering
Incheon National University

In modern society, many public places are designated as non-smoking areas, but due to the lack of enforcement personnel and the uncooperative attitude of smokers, it is oft en difficult to effectively control smoking behavior in non-smoking areas. To solve this problem, a real-time smoking detection system using deep learning-based image processi ng models can be developed to efficiently detect smoking behavior, provide warning me ssages to prevent smoking in non-smoking areas, and contribute to preventing damage a nd accidents caused by smoking in dangerous facilities.

Among many object detection deep learning models, the YOLO series of models has been actively studied by researchers and industry practitioners due to its fast real-time d etection and high accuracy. In this study, we compare and validate the performance of various recent models from YOLOv5 to YOLOv10 on smoking data to select the best model according to the application situation. The experimental results show that the e m odel of YOLOv9 has the highest performance in terms of mAP@50, and the x model o f YOLOv8 has the best performance among the compared models despite having a smal

l number of parameters. In addition, the t model of YOLOv9 is evaluated as a good ch oice for lightweight models such as embedded environments because it simultaneously sa tisfies a small number of parameters and high performance.

However, it was found that smoking detection performance depends not only on the model structure but also on hyperparameter settings such as dataset composition, learning rate, and batch size. Therefore, further experiments on large-scale, high-quality datasets a nd under various conditions are needed to improve performance and draw objective conc lusions.

We hope that the results of this research will contribute to solving social problems t hrough the development of a smoking detection system and contribute to the developme nt of image deep learning technology by providing useful data in the field of object det ection.

Key Words : Deep Learning, Image Processing, Real-Time Object Recognition, Smoking Detection, YOLO

감사의 글

늦은 나이에 새로운 학업에 도전할 용기를 북돋아 주고 따뜻한 응원과 격려로 언제나 제 곁을 지켜준 사랑하는 아내 O.J에게 깊은 감사를 전합니다. 당신의 지지와 응원이 없었다면 끝까지 마칠 수 없었을 것입니다.

또한 연구 방향을 함께 고민해 주시고 초안을 세밀히 검토하며 조언을 아낌없이 나눠 주신 지도교수님과 심사위원 교수님들께 감사드립니다. 세심한 지도는 제 연구를 완성하는 데 있어 많은 도움이 되었습니다. 그리고 무엇보다, 포기의 유혹을 떨쳐내고 이 여정을 끝까지 걸어온 저 자신에게도 조심스레 칭찬을 덧붙이고 싶습니다.

학위 취득을 기뻐하실 부모님과 가족들, 배움의 시작과 끝을 함께 하며 이끌어 주시고 도움을 주신 모든 분을 한 분씩 떠올리며 감사의 마음을 전합니다. 앞으로도 여러분의 기대와 응원에 부응하며, 계속 성장하고 행복한 삶을 살아가도록 하겠습니다.

2024년 12월 24일

남 춘 식 올림