증상 검출 및 질병 분류 작업의 결합을 통한 효율적인 넙치 상태 진단을 위한 다중작업학습 접근법

김수영, 이명철*

충남대학교, *한국전자통신연구원

jellyfish319@o.cnu.ac.kr, *mclee@etri.re.kr

A Multi-Task Learning Approach for Efficient Flatfish Status Diagnosis through Joint Symptom Detection and Disease Classification

Kim Sooyoung, Myungcheol Lee*

Chungnam National Univ., *Electronics and Telecommunications Research Institute

요 약

넙치 양식 산업은 질병으로 인해 연간 막대한 경제적 손실을 겪고 있으며, 특히 출혈, 염증과 같은 증상은 육안으로 식별하기 어려워 신속하고 정확한 진단에 한계가 있다. 본 연구에서는 이러한 문제를 해결하기 위해, 단일모형으로 넙치의 증상을 탐지함과 동시에 질병을 분류하는 다중작업학습(Multi-Task Learning) 기반의 딥러닝 모델을 구현한다. 모형은 객체 탐지에 우수한 성능을 보이는 YOLO 구조를 기반으로 두 가지 작업을 동시에 처리하는 모형을 설계하고 학습시켰다. 최종적으로 얻은 다중작업모형은 정확도 측면에서 기존 단일작업모형과 대비하여 검출에서는 비슷한 성능을, 분류에서는 7% 정도의 성능 하락이 발생하였으나, 전체 필요한 모델 크기를 12% 감소하고, 추론 시간을 28% 감소하는 효과를 보였다. 본 다중작업모델을 통해 스마트폰 등 저사양 단말에 탑재함으로써 양식 현장에서 신속한 통합 질병 진단 시스템의 가능성을 제시한다.

Ⅰ. 서론

양식 현장에서는 출혈성 질병이나 염증 등 육안으로 식별이 어려운 증상으로 인해 전문가 없이 신속한 질병 진단이 어렵고, 이로 인한 넙치 폐사로 연간 약 1,300억 원의 손실이 발생하고 있다.[1]

이를 해결하기 위해 2022년 제주도에서는 넙치 질병 탐지·분류·예측을 위한 AI 모델과 데이터를 공개하였으나[2], 작업별로 개별 모델을 사용하는 기존 방식은 추론 시간이 길고, 경량 기기에서의 활용에 제약이 있다. 본 논문에서는 이러한 한계를 극복하기 위해, YOLO 모델을 백본 모델로 하여 다중작업학습(Multi-Task Learning, MTL)[3]을 적용한 하나의 경량 모델로 증상 탐지와 질병 분류를 동시에 수행하는 방법을 제안한다.

Ⅱ. 본론

가. Dataset 구성

기존 AI허브에서 제공하는 증상 탐지의 라벨링 형식은 COCO형식이었기에 백본 모형으로 사용할 YOLO 포맷으로의 추가적인 변환을 수행하였다. AI허브에서 활용한 원본 라이브러리에서는 객체 탐지와 증상 분류 각각에 대한 학습만 제공하기에, 본 연구에서는 객체 탐지 관련 데이터로더는 기존 API를 사용하였고, 증상 분류는 커스텀 로더를 만들어 YOLO에서 사용하는 ImageNet 형식의 라벨 분류가 아닌 자체적인 라벨 형식을 정의하여 사용하였다.

나. Class 병합

사용할 데이터셋의 라벨의 경우 증상은 31종, 질병은 21종이며 학습용 데이터 4만 8천개와 검증용 데이터 7천개가 존재한다. 비브리오병은 각 병원체가 일으키는 질병이나 증상의 차이간 명확히 밝혀진 부분이 없기에 하나의 클래스로 통합하였다.[4] 한 개체가 복수의 질병을 앓는 경우를 모두 포함하면 70개 이상의 질병 조합이 나오므로 복수의 질병은 새로운 카테고리를 만들어 통합하는 형식으로 클래스를 병합하였다. 증상은 31가지로 클래스 간 불균형이 심하여 주요 증상인 출혈, 부식, 안구 증상, 궤양, 종양으로 나누고 나머지 증상들은 희소하게 존재하여 기타로 분류하였다.

다. Model 구조 및 구현

본 실험에 사용한 모형은 YOLO(You Only Look Once) 모형으로 객체 탐지, 분할, 분류에 대해 좋은 성능을 보여준다. 따라서 연구의 목표인 증상 탐지, 질병 분류 두 가지 작업에 대해 좋은 성능을 보여주는 YOLO 모델을 백본 모델로 선택하였다. 기존 YOLO 모형의 백본 네트워크로부터 추출한 특징을 스케일링하는 Neck으로부터 출력값을 입력으로 하는 분류 Head를 추가로 붙이는 방식으로 모형을 설계하였다.[그림 1] 넥의 구조를 보면 feature를 여러 방식으로 scaling한 layer들이 존재한다. 최종적으로 마지막에 존재하는 layer에서의 feature를 분류 헤드에 넘겨주는 것이 제일 좋은 성능을 보여주었기에 다음과 같은 구조로 모형을 설정하였다. 다양한 성능 평가 비교를 위해 백본 네트워크로 활용한 기본 모델은 YOLO11[5], YOLOv8 두 개의 모델을 사용하였다.

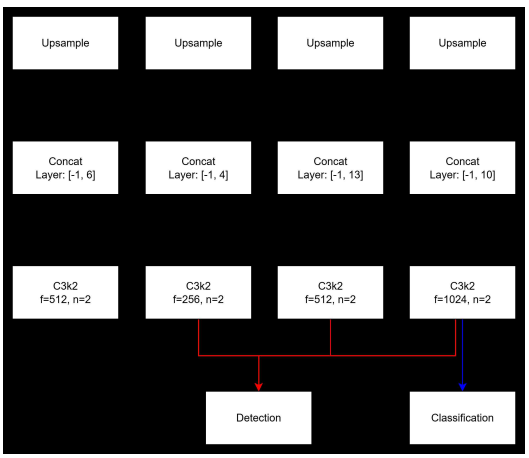

그림 1 YOLO11 MTL의 Neck과 Head 도식화 그림

YOLO 학습을 지원하는 ultralytics 라이브러리에는 다중작업학습을 따로 지원하지 않고 있기에 앞서 정의한 모형을 바탕으로 Pytorch 라이브러리와 ultralytics 라이브러리를 활용하여 구현하였다.

라. 손실함수와 평가지표

본 논문에서는 학습 효율성과 모델의 일반성을 확보하기 위해 작업별로 동일한 가중치를 부여하는 Equal Weighting 전략을 사용하였다. 본 실험은 YOLO 백본 아키텍처에 따른 MTL의 성능과 효율성 비교에 있으므로, 가장 보편적이고 표준적인 Equal Weighting을 통해 손실함수라는 변수를 통제하여 오로지 아키텍처의 영향을 분석하고자 하였다.

$$L_{\text{total}} = L_{\text{det}} + L_{\text{cls}}$$

평가지표는 Recall, Precision, mAP50, mAP95, F1 Score, Top1 Accuracy, Top5 Accuracy, 모델 크기, 추론 시간을 측정하여 고려하기로 하였다.

마. 실험 결과

YOLO 프레임워크가 기본적으로 제공하는 단일작업학습과 YOLO를 기반으로 하는 다양한 백본 네트워크를 바탕으로 다중작업학습을 진행한 결과이다.

평가지표 / 모델	Detection				Classification	
	mAP 50	mAP 50-95	Recall	Precision	Top1 Acc	Top5 Acc
yolo11n-det (merged)	0.389	0.225	0.393	0.489	-	-
yolo11n-det (original)	0.244	0.152	0.261	0.394	-	-
yolo11n-cls (merged)	-	-	-	-	0.727	0.947
yolo11n-cls (original)	-	-	-	-	0.693	0.886

표 1 Class 통합에 따른 YOLO 단일작업학습 실험 결과

평가지표 / 모델	Detection				Classification		
	mAP 50	mAP 50-95	Recall	Precision	F1 Score	Top1 Acc	Top5 Acc
yolo11n -mtl	0.385	0.214	0.388	**0.514**	0.414	0.651	**0.924**
yolo11s -mtl	0.377	0.211	0.387	0.49	**0.418**	**0.657**	0.92
yolo8n -mtl	0.372	0.207	0.391	0.476	0.381	0.622	0.923
yolo8s -mtl	**0.411**	**0.238**	**0.408**	0.513	0.204	0.489	0.915

표 2 YOLO 다중작업학습 실험 결과

평가지표 / 모델	Inference Time	Size
yolo11n - det	0.4ms	2.6M
yolo11n - cls	0.3ms	1.55M
yolo11n - mtl	0.5ms	3.65M

표 3 YOLO11n의 작업 별 추론 시간과 모델 크기

실험 결과를 해석해보면 [표 1]은 앞서 설정하였던 Class 병합의 효과를 보여준다. Class 병합을 수행하지 않은 데이터셋보다 병합을 한 데이터셋에서 각각의 단일작업에 대한 성능이 높음을 볼 수 있다. [표 2]에서는 이렇게 병합한 데이터셋을 바탕으로 여러 모형에 대한 다중작업학습을 진행하였을 때의 결과이다. 탐지에 대한 성능은 yolo8s가 가장 높으며 yolo11 모델이 질병 분류 작업에 더 좋은 성능을 냄을 확인할 수 있다. MTL 환경에서 모델 아키텍처에 따라 탐지와 분류 성능 간 Trade-off가 발생하며 yolo8s는 탐지에, yolo11n/s는 분류에 더 강점을 보인다. yolo8s의 경우 탐지 성능은 제일 좋으나 분류의 성능은 굉장히 떨어지기에 전반적인 성능을 고려해보면 yolo11 구조가 더 효율적이다. yolo11n의 증상 탐지 성능은 yolo11s 모델보다 더 좋으며 질병 분류 부분에서는 두 모델 간 성능 차이가 거의 없기에 최종적으로 모델의 크기가 더 작고 탐지에서 더 좋은 성능을 보여주는 yolo11n을 최적의 모델로 선정하였다.

yolo11n과 단일작업모델 간 성능 비교를 해보면, 객체 탐지 부분에서는 mAP 점수는 대등한 수준이며, 정밀도(Precision)는 MTL 모델이 더 높게 나왔다. 이는 MTL 모델이 다른 작업(분류)을 함께 학습하면서 공유 표현(Shared Representation)를 학습하였다는 점을 시사한다. 분류 성능에서는 단일작업모델이 우세하다. Top-1 Accuracy에서 약 7.6%p의 차이가 발생하며, 이는 모델이 탐지와 분류라는 두 가지 작업을 동시에 처리하도록 학습되면서 분류 작업에 대한 학습이 다소 희석된다고 해석된다. 마지막으로 MTL 모델의 추론 시간은 탐지 모델과 분류 모델을 각각 실행한 시간(0.7ms)보다 28% 더 빠른 0.5ms가 걸리며, 모델 크기 역시 각 모델을 합한 것(4.15M)보다 12% 더 작은 3.65M개의 파라미터를 갖는다.[표 3]

Ⅲ. 결론

본 연구에서는 단일모델로 객체 탐지와 질병 분류를 동시에 수행하는 다중작업학습 모델을 구축하고 그 효율성을 검증하였다. 이를 위해, 분류와 탐지를 동시에 수행하기 위한 커스텀 데이터 로더를 만들었고, 분류 헤드를 백본 네트워크에 최적으로 연결하는 방안을 고려하여 설계하였으며, 각 태스크의 학습 효율성과 일반성을 확보할 수 있는 손실함수를 사용하였다. 여러 YOLO 아키텍처를 비교 분석한 결과, yolo11n 구조가 탐지와 분류 성능 간의 균형이 제일 뛰어나기에 최적 모델로 선정하였다.

최종적으로 제시하는 yolo11n-mtl 모델은 전문 탐지 모델과 대등한 수준의 탐지 성능을 유지하면서도, 두 개의 단일모델을 사용하는 것보다 추론 속도는 빠르고 모델 크기는 작은 효율성을 보였다. 비록 분류 성능에서는 약간의 성능 저하가 발생했으나, 이는 효율성을 극대화하는 과정에서 발생하는 합리적인 트레이드오프(Trade-off)로 판단된다.

결론적으로, 본 연구에서 구현한 MTL 모델은 제한된 하드웨어 리소스 환경에서 성능 저하를 최소화하며 시스템의 효율성을 극대화할 수 있는 실용적인 해결책임을 입증하였다. 이는 실시간 처리가 요구되는 다양한 산업 현장에 효과적으로 적용될 수 있을 것으로 기대된다.

ACKNOWLEDGMENT

이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No. 2021-0-00136, 다양한 산업 분야 활용성 증대를 위한 대규모/대용량 블록체인 데이터 고확장성 분산 저장 기술 개발)

참 고 문 헌

[1] https://www.boka.co.kr/news/articleView.html?idxno=4437

[2] AI Hub - 넙치 질병 데이터 https://aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&dataSetSn=71345

[3] Caruana, R. (1997). “Multi-task learning”. 《Machine Learning》 28: 41 - 75

[4] Sohn, H., Kim, J., Jin, C. et al. Identification of Vibrio species isolated from cultured olive flounder (Paralichthys olivaceus) in Jeju Island, South Korea. Fish Aquatic Sci 22, 14 (2019). Standard(AES),"FIPS PUB ZZZ, 2001

[5] Rahima Khanam, Muhammad Hussain, arXiv:2410.17725v1 [cs.CV] 23 Oct 2024