라즈베리파이에 적합한 실시간 객체탐지 모델 연구

박승우[1], 박영진[2], 최혜원[3], 하승헌[4], 도유성[5]
[1, 3, 5]한국공학대학교 전자공학과 학부생
[2]롯데케미칼 정보보호담당 리더
[4]한국공학대학교 임베디드공학과 학부생
winnerpark17@tukorea.ac.kr, parkyoungjin@lotte.net, chchlgpdnjs@naver.com

Real-time Object Detection Model for Raspberry Pi

Seung-woo Park[1], Young-jin Park[2], Hye-won Choi[3], Seung-heon Ha[4], Yu-seong Do[5]
[1, 3, 5]Dept. of Electronic Engineering, Tech University of Korea
[2]Dept. of Information Protection Leader, Lotte Chemical
[4]Dept. of Embedded Engineering, Tech University of Korea

요 약

본 연구는 라즈베리파이4와 같은 저전력, 저비용 임베디드 시스템에서 효과적으로 작동할 수 있는 실시간 객체 탐지 모델을 제안한다. 우리는 대중적으로 유명한 YOLO를 사용하지 않고 MobileNetV1-SSDLite를 기반으로 한 경량화된 모델을 설계하고, 이를 라즈베리파이4에 최적화하여 구현하였다. 실험 결과, 제안된 모델은 평균 5 FPS의 처리 속도로 mAP 0.68을 달성하여, 기존의 모델들과 비교했을 때 라즈베리파이4에서 더 효율적인 성능을 보여주었다. 또한, Jetson Nano와의 비교를 통해 각 플랫폼의 특성을 분석하여 라즈베리파이4에 경량화된 모델이 충분하다라는 것을 소개한다.

1. 서론

실시간 객체 탐지(Object Detection)는 컴퓨터 비전 분야에서 중요한 과제 중 하나이다. 최근 딥러닝 기술의 발전으로 높은 정확도를 가진 객체 탐지 모델들이 개발되었지만, 이러한 모델들은 대부분 고성능 GPU를 필요로 한다. 그러나 대부분의 공학자들이 주로 라즈베리파이 그리고 Jetson Nano가 실시간 객체 탐지 디바이스로 선정하여 수요가 증가하고 있다. 이때, 사용자가 YOLO를 통해 실시간 객체탐지모델을 만든다면, 라즈베리파이의 경우 작동이 수월하지 않을 뿐만 아니라 이를 해결하기 위해서 상대적으로 비싸고 무게가 6배 차이가 나는 Jetson Nano를 해결책으로 많이 사용하는 경향을 보였다. 따라서 본 연구에서는 라즈베리파이4의 제한된 계산 능력을 고려하면서도 효과적인 실시간 객체 탐지를 수행할 수 있는 모델을 제안하고자 한다.

2. 제안 모델

본 연구에서는 MobileNetV1-SSDLite를 기반으로 한 경량화 모델을 제안한다.

3. 실험 및 결과

3.1 실험 환경

<표 1> 실험환경

하드웨어	라즈베리파이4 Model B (4GB RAM)
운영체제	Raspberry Pi OS (64-bit)
딥러닝 프레임워크	TensorFlow Lite, YOLOv8, YOLOv5
데이터셋	COCO 데이터셋의 서브셋을 사용하여 학습 및 평가를 진행하였다. 총 3개 클래스, 20,000장의 이미지로 구성되었다.

3.2 성능 평가

제안된 모델을 YOLOv5, YOLOv8, SSD MobileNet V1과 비교하였다. 평가 지표로는 mAP(mean Average Precision)와 FPS(Frames Per Second)를 사용하였다.

<표 2> 라즈베리파이4에서 YOLO모델과 SSD MobileNetV1(제안된 모델) 평가지표

모델	디바이스	mAP	FPS	전력 소비 (W)
YOLOv5	라즈베리파이4	0.37	0.8	3.7
YOLOv8	라즈베리파이4	0.42	0.6	3.9
SSD MobileNet V1(제안 모델)	라즈베리파이4	0.21	4.2	3.1

<표 2>에서 YOLO모델을 경량화하지 않고 사용했을 때, 라즈베리파이4 디바이스에서 실행시킨다면 상대적으로 mAP는 높게 평가되지만 FPS(Frames Per Second) 초당 프레임이 현저히 느려지고 소비 전력 또한 높게 나오는 것을 확인할 수 있다. 하지만 실시간 객체탐지 모델을 사용하는 사용자 입장에서 궁극적으로 우리가 이를 구현하려는 목적은 대부분이 실시간으로 모니터링하는 것이다. 그렇다면 높은 mAP와 낮은 소비전력도 중요하지만 높은 FPS가 가장 중요하다. FPS가 실시간 모니터링에 큰 영향을 미치므로 SSD MobileNet V1이 임베디드 디바이스인 라즈베리파이에서 구동하기 적합하다.

3.3 라즈베리파이4와 Jetson Nano 비교 분석

Jetson Nano의 공개된 벤치마크 결과와 하드웨어 사양을 바탕으로 다음과 같은 비교 분석을 수행하였다.

1. 전력 효율성:

- 라즈베리파이4는 일반적으로 더 낮은 전력 소비를 보여, 배터리 구동 장치나 저전력 응용 분야에 적합하다. 반면에 Jetson Nano는 GPU를 활용한 높은 성능을 제공하지만, 상대적으로 더 많은 전력을 소비할 것으로 예상된다.

2. 가격:

- 라즈베리파이4는 Jetson Nano보다 낮은 가격으로 제공되어, 비용 효율적인 솔루션이 필요한 프로젝트에 적합하다.

3. 무게:

- 라즈베리파이4에 비해서 Jetson Nano보다 6배 더 가볍다. 라즈베리파이4는 46g, Jetson Nano 261g으로 조사 결과가 나왔다. 따라서 무게 경량화가 필요한 프로젝트는 이를 참고하면 좋을거 같다. 그리고 추가적으로 두 기기 모두 칩 외부 물리적인 보호 또는 발열로 인해 예방조치를 위한 방열판 또는 쿨링팬을 설치한다면 무게가 이 보다는 좀더 나온다는 것을 고려해야 한다.

4. 결론

본 연구에서는 자율주행 비행에 필요한 경량 디바이스의 예로 라즈베리파이에 적합한 실시간 객체탐지모델을 제안하였으며, 객체탐지 모델이 구동되는 라즈베리파이와 Jetson Nano 두 개의 디바이스 비교 분석을 통해 각 플랫폼의 장단점과 객체모델 구도에 대한 예측을 파악하였다.

MobileNetV1-SSDLite를 기반으로 한 경량화 모델과 하드웨어 최적화 기법을 통해, 제한된 계산 능력에서도 효과적인 객체 탐지가 가능함을 보였다. 제안된 모델(SSD MobileNet V1)은 YOLOv5, YOLOv8과 비교했을 때 매우 높은 FPS를 달성하였다. 따라서 비용적인 측면, 무게적인 측면, 그리고 MCU가 이동하는 것을 고려했을 때 제안된 모델(SSD MobileNet V1) 또는 그 이상의 모델을 사용하는 것을 권장한다. 향후 연구에서는 모델의 정확도를 더욱 향상시키면서도 전력 소비를 최소화하는 방안을 탐구할 예정이다. 그리고 MobileNet V1 뿐만 아니라 SSD MobileNet V2, SSD MobileNet V3 버전으로 도약해서 사용해도 전력소비가 높을 뿐 성능은 훨씬 높을 것이고 이에 대해 연구할 예정이다. 또한, Jetson Nano에서의 직접적인 성능 평가와 최적화 연구를 통해 더 포괄적인 임베디드 플랫폼 비교 분석을 수행할 계획이다.

※본 논문은 과학기술정보통신부
대학디지털교육역량강화 사업의 지원을 통해
수행한ICT멘토링 프로젝트 결과물입니다.

참고문헌

[1] Howard, A. G., et al. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv preprint arXiv:1704.04861.
[2] Liu, W., et al. (2016). SSD: Single Shot MultiBox Detector. In European Conference on Computer Vision (pp. 21-37). Springer, Cham.
[3] Jocher, G., et al. (2021). YOLOv5. https://github.com/ultralytics/yolov5
[4] Jocher, G., et al. (2023). YOLOv8. https://github.com/ultralytics/ultralytics
[5] Sandler, M., et al. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 4510-4520).
[6] Lin, T. Y., et al. (2014). Microsoft COCO: Common Objects in Context. In European Conference on Computer Vision (pp. 740-755). Springer, Cham.
[7] NVIDIA. (2021). Jetson Nano Developer Kit. https://developer.nvidia.com/embedded/jetson-nano-developer-kit
[8] Raspberry Pi Foundation. (2021). Raspberry Pi 4 Model B. https://www.raspberrypi.org/products/raspberry-pi-4-model-b/