JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 29 No. 8, pp. 157-164, August 2024
https://doi.org/10.9708/jksci.2024.29.08.157

Object Detection Performance Analysis between On-GPU and On-Board Analysis for Military Domain Images

Du-Hwan Hur*, Dae-Hyeon Park**, Deok-Woong Kim*, Jae-Yong Baek**,
Jun-Hyeong Bak***, Seung-Hwan Bae****

*M. S. candidate, Vision & Learning Lab, Inha University, Incheon, Korea
**Ph.D. candidate, Vision & Learning Lab, Inha University, Incheon, Korea
***Researcher, PGM Seeker R&D, LIG Nex1 Co., Ltd., Yongin, Korea
****Associate Professor, Vision & Learning Lab, Dept. of Electrical and Computer Engineering, Inha University, Incheon, Korea

[Abstract]

In this paper, we propose a discussion that the feasibility of deploying a deep learning-based detector on the resource-limited board. Although many studies evaluate the detector on machines with high-performed GPUs, evaluation on the board with limited computation resources is still insufficient. Therefore, in this work, we implement the deep-learning detectors and deploy them on the compact board by parsing and optimizing a detector. To figure out the performance of deep learning based detectors on limited resources, we monitor the performance of several detectors with different H/W resource. On COCO detection datasets, we compare and analyze the evaluation results of detection model in On-Board and the detection model in On-GPU in terms of several metrics with mAP, power consumption, and execution speed (FPS). To demonstrate the effect of applying our detector for the military area, we evaluate them on our dataset consisting of thermal images considering the flight battle scenarios. As a results, we investigate the strength of deep learning-based on-board detector, and show that deep learning-based vision models can contribute in the flight battle scenarios.

▸ **Key words**: Object detection, On-GPU Neural Detector, On-Board Neural Detector, Quantization, Military aircraft detection, Model Deployment

• First Author: Du-Hwan Hur, Dae-Hyeon Park, Deok-Woong Kim, Jae-Yong Baek,
Corresponding Author: Seung-Hwan Bae
*Du-Hwan Hur (gjenghks@inha.edu), Vision & Learning Lab, Inha University
**Dae-Hyeon Park (saintPalite2221@inha.edu), Vision & Learning Lab, Inha University
*Deok-Woong Kim (k5000plus@inha.edu), Vision & Learning Lab, Inha University
**Jae-Yong Baek (jy1213@inha.edu), Vision & Learning Lab, Inha University
***Jun-Hyeong Bak (junhyeong.bak@lignex1.com), PGM Seeker R&D, LIG Nex1 Co., Ltd.
****Seung-Hwan Bae (shbae@inha.ac.kr), Vision & Learning Lab, Dept. of Electrical and Computer Engineering, Inha University
• Received: 2024. 06. 18, Revised: 2024. 07. 05, Accepted: 2024. 07. 09.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

[요 약]

본 논문에서는 제한된 자원을 가진 보드에서 딥러닝 기반 검출기 구축에 대한 실현 가능성에 대해 논의한다. 많은 연구에서 고성능 GPU 환경에서 검출기를 평가하지만, 제한된 연산 자원을 가진 보드에서의 평가는 여전히 미비하다. 따라서 본 연구에서는 검출기를 파싱하고 최적화하는 것으로 보드에 딥러닝 기반 검출기를 구현하고 구축한다. 제한된 자원에서의 딥러닝 기반 검출기의 성능을 확인하기 위해, 여러 검출기를 다양한 하드웨어 자원에서 모니터링하고, COCO 검출 데이터 셋에서 On-Board에서의 검출 모델과 On-GPU의 검출 모델을 mAP, 전력 소모량, 실행 속도(FPS) 관점으로 비교 및 분석한다. 그리고 군사 분야에 검출기를 적용한 효과를 고려하기 위해 항공 전투 시나리오를 고려할 수 있는 열화상 이미지로 구성된 자체 데이터 셋에서 검출기를 평가한다. 결과적으로 우리는 본 연구를 통해 On-Board에서 모델을 실행하는 딥러닝 기반 검출기의 강점을 조사하고, 전장 상황에서 딥러닝 기반 검출기가 기여할 수 있음을 보인다.

▸ **주제어**: 객체 검출, 온-GPU 딥러닝 기반 검출기, 온-보드 딥러닝 기반 검출기, 양자화, 군용기 검출, 모델 구축

I. Introduction

객체 검출 기술은 딥러닝 모델의 도입으로 큰 성공을 거두고 비약적으로 성능이 향상되었다. 하지만, 딥러닝 기반 객체 인식 모델의 성능 평가는 대부분 GPU (Graphics Processing Unit) 환경(이하 On-GPU)에서 테스트 되었다. 실제 환경에서 고성능 모델을 적용하는 것은 높은 전력 소모량, 물리적 크기와 무게, 높은 가격 등의 이유로 많은 문제점에 직면한다.

이 문제를 해결하기 위해 상대적으로 낮은 가격, 낮은 전력 소모량을 가진 On-Board에서 모델을 추론하는 것이 해결 방법 중, 하나가 될 수 있다. 하지만 낮은 연산 성능으로 인해 실시간 추론을 하기에는 많은 성능적 제약이 따른다. 따라서 On-Board 딥러닝 모델을 실시간으로 추론하기 위해서는 딥러닝 모델에 가지치기(Pruning), 양자화(Quantization) 등의 최적화(Optimization) 기술 [1, 2, 3]을 딥러닝 모델에 적용함으로써 최대한 딥러닝 모델을 경량화 시켜야 한다. 하지만 이러한 최적화 기법은 딥러닝 모델의 기존 가중치 정보를 감소시키거나 제거하기 때문에 모델 정확도를 감소시킬 수 있다. 따라서 On-GPU에서의 모델 추론과 On-Board에서의 모델 추론 간의 성능 분석이 필요하다. 하지만 최신 연구에서는 On-GPU에서의 모델 추론과 On-Board에서의 모델 추론 간의 분석이 매우 미진하다. 비록 [4, 5]와 같이 일부 On-Board에서 평가된 딥러닝 모델 분석 결과가 있으나 높은 연산량과 높은 전력 소모량은 실제 환경과 다소 차이가 있다.

특히, 실시간 전장 상황은 전력 공급이 제한될 수 있으며, 통신이 끊길 수 있을 뿐만 아니라 예상하기 어려운 주변 환경 변화가 일어날 수 있다. 그 중에서도 주/야간을 가리지 않고 작전을 실행해야 될 수 있기 때문에 높은 조도 변화에 대응할 수 있어야 한다. 열화상 이미지(Infrared Thermal Image)는 열화상 카메라를 통해 생성되며 조도 환경 변화에 강건한 특성을 가진다. 가시광 이미지(RGB Image)가 가시광선을 인식하는 한계상 저조도 환경에서 객체를 식별하는데 제한적이지만, 열화상 이미지는 적외선을 인식하기 때문에 저조도 환경에서도 객체를 충분히 식별할 수 있다. [6] 따라서 열화상 이미지는 조도 환경 변화에 대응이 필요한 군사 분야 등에서 활용된다. 예를 들어 F-15K의 경우 표적 탐지 및 추적을 위해 열화상 카메라가 사용된다. [7]

앞서 언급했듯, 딥러닝 모델이 전장 상황에서 제한적인 전력 공급 상황과 통신 문제, 그리고 효율적인 전술 수행 능력을 갖추기 위해서는 모델이 독립적이고 실시간으로 작동될 필요가 있다. 따라서 실시간으로 딥러닝 모델이 독립적으로 실행되어야 하는 경우, 가격, 전력 공급 문제로 인해 On-Board 에서 실행하는 것을 고려해야 한다. 따라서 이러한 제한된 환경에서 최근 딥러닝 검출 기술을 평가하기 위해, 본 연구에서는 다음과 같이 검출기를 구현하고 실험 및 평가를 하였다:

(1) On-GPU 딥러닝 모델과 On-Board 딥러닝 모델 간의 성능 차이를 확인하기 위해 COCO 데이터 셋을 기반으로 객체 검출 모델을 평가하고 분석한다.

(2) 실제 전장 상황과 유사한 열화상 이미지로 구성된 항공 데이터 셋 검출 실험을 통해 온 보드에서 최근 객체 검출 기술의 성능 값과 전장 환경에 대한 적용 가능성을 논한다.

II. Related Works

본 장에서는 On-GPU 및 On-Board에 탑재할 딥러닝 기반 객체 검출 기술에 대해 소개하고, On-Board와 On-GPU 간의 차이에 대해 사전 연구된 내용에 대해 소개한다.

1. Object Detection

객체 검출(Object Detection)은 이미지가 주어질 때, 객체의 클래스를 분류(Classification)하고 객체의 위치(Localization)를 추론하는 작업을 말한다. 객체 검출을 실행할 때, 객체의 비율을 사전정의하여 객체 가능성이 있는 후보(Proposal)를 생성한다. 이것을 앵커 박스(Anchor box)라고 하고, 앵커를 사용하는 검출기를 앵커 기반 검출기(Anchor-based Detector)라고 한다. 검출기는 앵커 박스를 기반으로 객체의 클래스와 경계 상자를 추론한다. 예로써, Faster R-CNN [8]은 Region Proposal Network를 통해 객체 후보 영역을 예측하고, 예측된 영역에서 앵커 박스를 이용해 객체의 클래스 판별 및 객체 상자를 조정한다. YOLOv3 [9]는 이미지를 일정 크기의 그리드(Grid)로 나눈 후, 각 그리드마다 앵커 박스를 사용해 객체 클래스 판별 및 객체 상자를 조정한다.

하지만 앵커 기반 검출기의 경우, 객체에 적합한 앵커(Positive sample)는 소수이며, 그 외는 Negative sample이기 때문에 학습 시 클래스 불균형 문제가 발생할 수 있다. 또한, 객체 별로 적합한 앵커를 생성할 수 없기 때문에 학습 및 추론 시 정확도 저하를 일으킬 수 있고, 또한 앵커 개수가 많아질수록 모델의 복잡도가 증가된다.

이 문제를 해결하기 위한 앵커 프리 검출기(Anchor-Free Detector)는 고정된 앵커 박스를 사용하지 않는 대신 특징맵으로부터 직접 객체의 클래스의 경계 상자를 예측하는 방법이다. 이를 통해 앵커를 사전정의하는 대신 학습을 통해 직접 모델이 박스를 생성하기 때문에 기존 앵커 기반 검출기의 단점을 해소한다. 본 연구에서는 앵커 프리 검출기인 YOLOX [10]를 분석 대상으로 설정하고, On-GPU와 On-Board 간의 비교 분석을 수행한다.

III. Methods

본 장에서는 YOLOX 검출기에 대해 검토하고, YOLOX를 On-Board에 탑재 및 실행하기 위해 연산을 수행할 NPU인 Hailo-8 [11]에 대해 소개한다. 그리고 Hailo-8 NPU에서 실행하기 위한 모델 변환 방법에 대해 소개한다.

1. Revisiting the YOLOX Detector

본 절에서는 실험 및 분석에 사용하기 위한 YOLOX 검출기에 대해 검토한다. YOLOX는 실시간 객체 검출을 위한 앵커 기반 검출기 YOLOv3를 개선한 검출기로, 특징맵(Feature map)을 추출하기 위한 백본(Backbone) 네트워크, 특징맵을 개선할 수 있는 넥(Neck) 네트워크, 객체의 박스와 클래스를 추론할 수 있는 헤더(Header) 네트워크로 구성된다. 모델 파라미터 크기의 경중에 따라 순서대로 YOLOX-X, YOLOX-L, YOLOX-M, YOLOX-S, YOLOX-Tiny, 그리고 YOLOX-Nano로 구성된다. YOLOX는 검출기의 실시간성을 유지하면서도 정확도를 향상시키기 위해 헤더 네트워크 개선, 데이터 증강 방법 추가, 앵커 프리 검출기로 설계된다. YOLOX는 기존 YOLOv3와 비교해 헤더 네트워크 내의 분류/회귀 모델이 단일 헤더로 구성되어 추론하는 대신 분류, 회귀 헤더로 별도로 분리한다. 이는 분류 및 회귀 작업 간의 충돌을 완화하고, 종단 간 학습(End-to-End Learning)에서 발생하는 성능 저하를 최소화한다. 그리고 모델의 일반화 성능을 향상시키기 위해, 학습 데이터의 다양성을 확보할 수 있는 강력한 데이터 증강 기법인 MixUp [12]과 Mosaic으로 모델을 학습시킨다. 마지막으로 객체의 위치, 높이, 너비에 대해 직접 예측하도록 재설계함으로써 앵커 프리 검출기로 모델링한다. 이는 다양한 크기 및 비율의 객체 박스에 대해 적응적으로 결정할 수 있게 만든다.

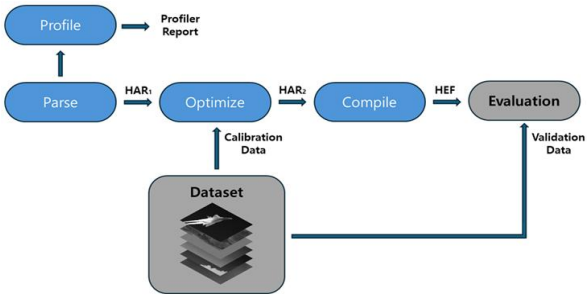

Fig. 1. Model Converting Process

2. Optimizing the model for NPU

On-Board 딥러닝 모델을 실행하기 위해서는 딥러닝 모델을 연산하기 위한 NPU (Neural Processing Unit)가 필요하다. NPU는 딥러닝 연산에 최적화된 프로세서로, GPU와 비교해 비교적 낮은 전력 소모량으로 설계된다. 하지만 GPU보다 낮은 연산 속도를 가지고 있어 모델은 상대적으로 낮은 속도로 추론된다. 그럼에도 불구하고, 실시간 전장 상황 등의 어플리케이션에 딥러닝 모델을 적용하기 위해서는 실시간으로 모델이 추론되어야 하는 문제에 직면한다. 이를 해결하기 위해 NPU에서 경량화

(Light-weight)된 딥러닝 모델을 사용해 추론 속도를 향상시킬 필요가 있다. 모델 경량화 기법은 주로 양자화(Quantization), 가지치기(Pruning) 등이 사용된다. 양자화는 딥러닝 기반 모델의 연산 표현을 고비트에서 저비트로 감소시키는 방식으로, 완전 정밀도 표현 방식인 32비트 연산 방식(FLOAT32)를 정수 표현 방식인 8비트(INT8) 등으로 압축시킨다. 가지치기는 딥러닝 모델의 불필요한 가중치를 제거하는 방식이다. 두 가지 방법 모두 연산량을 감소시키기 때문에 모델의 추론 속도를 향상시킬 수 있으나, 모델의 정확도가 저하되기 때문에 최대한 모델 정확도 감소를 최소화시켜야 하는 문제점이 있다.

본 연구에서 사용할 NPU인 Hailo-8은 이스라엘의 Hailo사에서 제공하는 NPU로 26 TOPS (Tera Operations Per Second)의 연산 성능을 갖고 있으며 M.2 폼팩터를 지원한다. Hailo-8에서 모델을 실행하기 위해서는 기존 가중치 파일을 Hailo-8에서 실행 가능한 파일인 HEF 파일로 변환해야 실행할 수 있다. Fig. 1에 나와있듯이, HEF 모델 변환 방법은 다양한 형태의 딥러닝 모델에 대응하기 위해 기존 가중치 모델을 그래프 구조로 변환하는 Parsing, 최적화 기법을 적용해 모델을 경량화하는 Optimization, 마지막으로 Hailo-8에서 실행 가능한 HEF 파일로 변환하는 Compile로 구성된다. 그리고 생성된 HEF 파일을 통해 모델을 평가(Evaulation)할 수 있다. Parsing 단계는 다양한 딥러닝 모델을 처리하기 위해 Hailo 호환 표현 방법(Hailo-compatible representation)으로 변환한다. 이를 위해 가중치 모델을 그래프 구조로 변환한다. 이 작업을 위해서는 기존 가중치 파일의 입력 노드와 출력 노드(예. 헤더 네트워크의 마지막 레이어)의 레이어 이름을 입력하여 설정해야 한다. Fig. 3에서 확인할 수 있듯, Parsing 변환 전후에서 네트워크 연결 구조가 선형 구조에서 그래프 구조로 변경된 것을 확인할 수 있다. 이를 통해 HAR1 (Hailo Archive) 파일을 생성한다. 또한, 이 과정에서 사전에 모델의 예상 정확도를 파악할 수 있는 Profiler Report를 생성할 수 있다.

Optimization 단계는 실시간 모델 추론을 위해 딥러닝 모델을 최적화하고 경량화하는 작업인 양자화을 수행한다. Optimization은 두 가지 단계인 완전 정밀도 최적화와 양자화 최적화로 구성된다. 완전 정밀도 최적화는 부동 소수점 정밀도 모델로부터 양자화하기 전에 모델을 사전에 최적화한다. 양자화로 인해 발생하는 성능 저하를 최소화하기 위해 양자화 노이즈(Quantization noise)가 작은 레이어의 채널의 가중치를 증폭하는 Equalization 알고리즘 [14], 가지치기 기법 등이 적용된다.

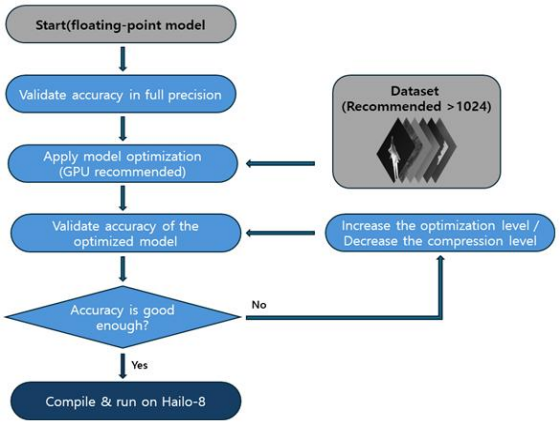

Fig. 2. Optimization Process

양자화 최적화는 기존 가중치 모델의 부동 소수점 표현 방식을 정수 표현으로 모델을 압축하기 위해 QAT (Quantization-aware Training) 알고리즘을 통해 양자화를 수행한다. 양자화를 학습 과정에 통합하는 것으로, 딥러닝 모델을 양자화 환경에서 학습시킴으로써 기존 부동소수점 값으로 구성된 가중치와 양자화된 값으로 구성된 가중치 간의 양자화 노이즈에 대응함으로써 양자화 상태에서도 높은 정확도를 유지하도록 한다. 추가적으로 정확도 저하를 최소화하기 위해, 레이어 내 채널의 편향값(Bias)에 상수를 추가하여 기존 학습된 분포가 이동(Shift)되는 현상을 최소화하는 IBC [15], 양자화에 적합한 미세조정 방법인 QFT [16], 양자화 수행 시 적응적으로 반올림을 수행하는 AdaRound [17] 등을 통해 양자화를 수행한다. 추가적으로 모델 정확도를 향상시키기 위해 기존 가중치 모델을 선생 모델(Teacher model)로 하고, 양자화 모델을 학생 모델(Student model)로 해 선생 모델이 생성하는 특징맵을 학생 모델이 유사하게 생성하도록 학습하는 지식 증류(Knowledge Distillation)을 적용할 수 있다. 결과적으로 Optimization 후 HAR2 파일을 생성한다. 마지막으로 Fig. 2에 나와 있듯이, 최적화된 모델의 정확도를 검증하고, 정확도가 불충분할 경우 최적화/압축 수준을 다른 구성으로 반복한다. 마지막으로, Compile 단계에서는 HAR2 파일을 Hailo NPU에서 동작 가능한 바이너리 파일을 생성한다. 이를 통해 HEF (Hailo Executable Format) 파일을 생성한다. 이 HEF 파일을 통해 양자화된 모델을 Hailo-8 NPU에서 실행할 수 있다.

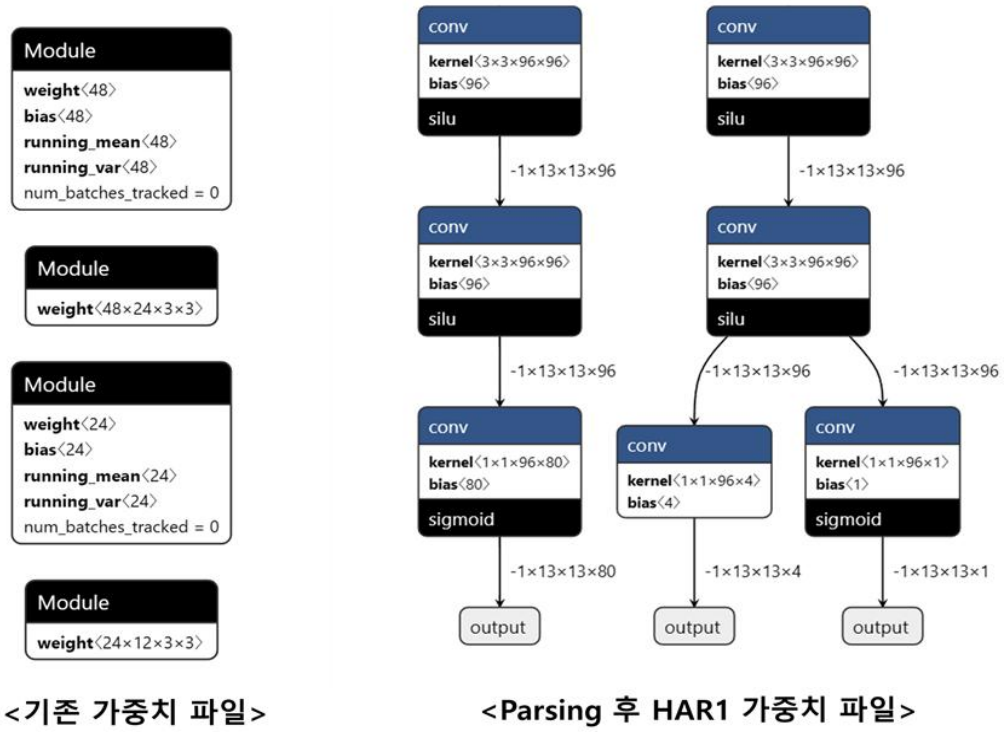

Fig. 3. Comparison before and after applying parsing in [13]

IV. Experiments

본 장에서는 YOLOX 검출기를 사용해 On-Board AI와 On-GPU AI 간의 성능을 제공하고 두 모델 간의 차이를 분석한다. 그리고 열화상 이미지 기반 항공 데이터 셋을 On-GPU에서 YOLOX 검출기로 평가해 딥러닝 기반 검출기의 전장 상황 적용 가능성을 확인한다.

1. Implementation Detailed

On-Board 딥러닝 모델과 On-GPU 딥러닝 모델 간의 성능 차이를 분석하기 위해, Table 2에 제시된 두 가지의 하드웨어 환경에서 YOLOX 검출기를 사용해 실험한다. 학습 및 평가 모델은 YOLOX-M, YOLOX-S, YOLOX-Tiny를 사용한다. COCO 2017 [18] 데이터 셋에서의 성능 비교를 위해 COCO 학습 데이터 셋으로 학습된 가중치 모델 사용하여 이미지 크기 640x640으로 평가한다. 이 때 학습 파라미터는 기존 YOLOX 실험 환경과 동일하게 셋팅한다. 또한 On-Board에서 YOLOX를 COCO 검증 데이터 셋에서 평가하기 위해 우리는 On-GPU 딥러닝 모델의 가중치 파일을 3.2절에 따라 HEF 파일을 생성한다. 이 때 32비트에서 8비트로 모델을 양자화한다. 그리고 임베디드 보드 Solidrun ClearFog LX2에 Hailo-8 NPU를 장착한 뒤, 양자화된 HEF 파일을 실행한다. 또한 항공 평가 데이터 셋을 평가하기 위해, COCO 학습 데이터 셋으로 사전학습된 YOLOX-Tiny 모델을 기반으로 항공 학습 데이터 셋으로 미세조정(Fine-tuning)을 수행한다.

Table 1. Detailed comparison On-GPU YOLOX with On-Board YOLOX on COCO validation set

	Model	Parameters	mAP (%)↑	Power Consumption(W)↓	FPS (Hz)↑
On GPU YOLOX	YOLOX-Tiny	5.1M	34.8	108	45.5
	YOLOX-S	9.0M	39.2	127	47.0
	YOLOX-M	25.3M	46.2	208	43.8
On Board YOLOX	YOLOX-Tiny	5.1M	32.0 (-2.8%↓)	2.24 (-97.9%↓)	82.9
	YOLOX-S	9.0M	38.6 (-0.6%↓)	2.79 (-97.8%↓)	75.5
	YOLOX-M	25.3M	45.5 (-0.7%↓)	3.30 (-98.4%↓)	45.0

2. Datasets

본 실험은 COCO 데이터 셋과 항공 전투 시나리오 학습 데이터 셋을 통해 모델을 평가한다. COCO 데이터 셋은 객체 검출 작업에 사용할 수 있는 데이터 셋으로 80개의 클래스로 구성된다. 학습 데이터 셋은 약 118K장, 검증 데이터 셋은 약 5K장, 테스트 데이터 셋은 약 41K장으로 구성된다. 우리는On-Board 딥러닝 모델과 On-GPU 딥러닝 모델 간의 비교하기 위해 COCO 검증 데이터 셋을 사용하여 평가한다. 또한, 항공 전투 시나리오를 고려하기 위한 열화상 이미지 기반 항공 학습 데이터 셋을 On-GPU에서 평가한다. 해당 데이터 셋은 열화상 이미지로 구성되며, 16비트의 원본 데이터 셋을 8비트로 변환하기 위해 Min-Max 정규화 기법을 적용한다. 클래스는 항공기와 회전익기 2개로 구성된다.

3. Metric

mAP는 클래스 별 AP를 평균낸 값으로, AP는 Precision-Recall의 그래프가 주어질 때 그래프의 면적 값을 계산해 모델의 정확도로 사용한다. 전력 소모량은 GPU 혹은 NPU에서 소모된 초 당 평균 전력 소모량을 계산한다. FPS (Frame Per Second)는 초 당 처리되는 프레임의 수로 모델의 추론 속도를 판단한다.

4. Performance Analysis between On-GPU YOLOX and On-Board YOLOX on COCO val.

On-Board YOLOX와 On-GPU YOLOX 간의 모델 성능 비교를 위해, COCO 검증 데이터 셋을 사용하여 평가를 수행한다. 비교를 위해 우리는 각 On-Board/On-GPU 별로 YOLOX-M, YOLOX-S, YOLOX-Tiny 모델을 사용해 평가한다. Table. 1에서 확인할 수 있듯이, On-GPU YOLOX과 비교해 On-Board YOLOX는 YOLOX-M, YOLOX-S, YOLOX-Tiny 모델 각각 mAP가 2.8%, 0.6%, 0.7% 감소하였다. 모든 모델에서 검출 정확도는 감소하였으나 전력 소모량은 97.9%, 97.8%, 98.4% 감소하였으며 FPS 또한 On-GPU YOLOX와 비교해 On-Board YOLOX가 모두 더 높은 것을 확인하였다. 이는 모델 최적화를 통해 비교적 적은 모델 정확도 손실만으로 낮은 연산 속도를 가진 NPU에서 모델 추론 속도를 가속 할 수 있다는 것을 보여준다.

Table 2. Hardware Specification

	Type	Hardware Specification
On-GPU YOLOX	CPU	Intel Xeon Scalable Gold 6330 2ea
	RAM	DDR4-3200 1TB
	GPU	NVIDIA RTX A6000 48GB
On-Board YOLOX	CPU	Arm Cortex A72 2Ghz
	RAM	DDR4-3200 ECC/REG 64GB
	NPU	Hailo-8 (26 TOPS)

5. Analysis on Fighter Plane Battle Scenarios

실제 전장 상황을 고려하기 위해 열화상 이미지 기반 데이터 셋에서 모델 On-GPU YOLOX를 평가한다. 평가를 위해 항공 평가 데이터 셋을 사용해 평가한다. 비교를 위해 우리는 On-GPU에서 YOLOX-Tiny 모델을 사용해 평가한다. Table. 3에서 확인할 수 있듯이, 입력 이미지 크기가 256x256 때와 비교해, 입력 이미지 크기가 416x416 일 때 mAP가 23.5% 더 크게 증가하는 것을 확인할 수 있지만, FPS는 4.7% 감소하였다. 이는 입력 이미지의 크기가 검출 정확도에 큰 영향력을 끼친다는 것을 나타낸다. 그럼에도 불구하고, 256x256의 크기 이미지에서도 작은 크기의 객체를 검출할 수 있음을 Fig. 4에서 보일 뿐만 아니라 60 프레임 이상의 빠른 추론 속도를 가지고 있어 실제 전장 상황에서 딥러닝 기반 검출기가 효과적이고 실시간 모델 구축 실현 가능성이 높을 수 있음을 나타낸다.

Table 3. Evaluation Results of On-GPU YOLOX on Fighter Plane Battle Scenarios Test Datasets

Model	Test Size	mAP ↑	FPS ↑
YOLOX-Tiny	416	42.8	61.5
	256	19.3	64.6

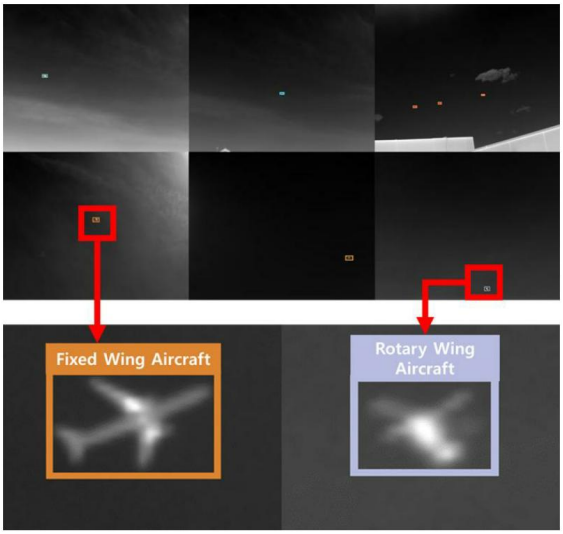

Fig. 4. Qualitative Results on Our Flight Test Datasets

V. Conclusion

본 연구에서 우리는 YOLOX 검출기를 사용해 On-GPU 딥러닝 모델과 On-Board 딥러닝 모델 간의 비교 분석 실험을 수행하고, 이를 통해 모델 최적화를 통해 딥러닝 모델이 상대적으로 적은 정확도 손실만으로 On-Board에서 실시간으로 동작할 수 있음을 확인하였다. 또한, 실제 전장 상황에 적용할 수 있을 열화상 이미지 기반 항공 평가 데이터 셋에서 실험을 수행하였으며, 실현 가능성을 가진 검출 정확도와 추론 속도가 나올 수 있음을 확인하였다. 본 연구를 통해 우리는 On-Board 딥러닝 모델의 강점을 확인하고, 추후 이 연구를 통해 전장 상황에서 딥러닝 모델이 도입되는 것에 기여할 수 있을 것으로 기대한다.

ACKNOWLEDGEMENT

This work was supported by Korea Research Institute for defense Technology planning and advancement(KRIT) grant funded by the Korea government(DAPA(Defense Acquisition Program Administration)) (No. 21-107-F00-015(KRIT-CT-22-024-02), 2023)

REFERENCES

[1] Blalock, Davis, et al. “What is the state of neural network pruning?.” Proceedings of the Third Conference on Machine Learning and Systems, pp.129-146, Austin, TX, USA, March 2020. DOI: https://doi.org/10.48550/arXiv.2003.03033

[2] Nagel, Markus, et al. "A white paper on neural network quantization." arXiv preprint arXiv:2106.08295, June 2021. DOI: https://doi.org/10.48550/arXiv.2106.08295

[3] Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean.“Distilling the knowledge in a neural network.” arXiv preprint arXiv:1503.02531, Aug. 2015. DOI: https://doi.org/10.48550/arXiv.1503.02531

[4] Wang, Xinjiang, Liu, Zeyu, Hu, Yu, Xi, Wei, Yu, Wenxian, & Zou, Danping. "Featurebooster: Boosting feature descriptors with a lightweight neural network." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7630-7639, Vancouver, BC, Canada, June 2023. DOI: 10.1109/CVPR52729.2023.00737

[5] Cao, Ziang, Chen, Ke, Zhu, Yi, & Zhang, Tao. "Tctrack: Temporal contexts for aerial tracking." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 456-460, LA. USA, June 2022. DOI: 10.1109/CVPR52688.2022.01438

[6] Agrawal, Kshitij, and Anbumani Subramanian. "Enhancing object detection in adverse conditions using thermal imaging." arXiv preprint arXiv:1909.13551, Oct. 2019. DOI: https://doi.org/10.48550/arXiv.1909.13551

[7] Boeing. “F-15K 슬램이글(Slam Eagle).” URL: www.boeing.co.kr/products-and-services/defense-space-and-security/f-15k-slam-eagle.

[8] Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." Advances in neural information processing systems 28, pp. 91-99, Quebec, Canada, Dec. 2015. DOI: https://doi.org/10.1109/tpami.2016.2577031

[9] Redmon, Joseph, & Farhadi, Ali. "Yolov3: An incremental improvement." arXiv preprint arXiv:1804.02767, Aug. 2018. DOI: https://doi.org/10.48550/arXiv.1804.02767

[10] Ge, Zheng, et al. "Yolox: Exceeding yolo series in 2021." arXiv preprint arXiv:2107.08430, April 2021. DOI: https://doi.org/10.48550/arXiv.2107.08430

[11] S. Ward-Foxton, “Details of hailo ai edge accelerator emerge,”8 2019.[Online]. URL: https://www.eetimes.com/details-of-hailoai-edge-accelerator-emerge/

[12] Zhang, Hongyi, Cisse, Moustapha, Dauphin, Yann N., & Lopez-Paz, David. "mixup: Beyond empirical risk minimization." arXiv preprint arXiv:1710.09412, Aug. 2017. DOI: https://doi.org/10.48550/arXiv.1710.09412

[13] Netron. "Lutz Roeder’s Netron." URL: https://netron.app/

[14] Meller, Eldad, Almog, Uri, & Grobman, Mark. "Same, same but different: Recovering neural network quantization error through weight factorization." International Conference on Machine Learning, PMLR, pp. 4486-4495, Long Beach, California, USA, June 2019. DOI: https://doi.org/10.48550/arXiv.1902.01917

[15] Finkelstein, Alexander, Almog, Uri, & Grobman, Mark. "Fighting quantization bias with bias." arXiv preprint arXiv:1906.03193, June 2019. DOI: https://doi.org/10.48550/arXiv.1906.03193

[16] McKinstry, Jeffrey L., Andersen, David, Wang, Xinjiang, & Chattopadhyay, Aniruddha. "Discovering low-precision networks close to full-precision networks for efficient embedded inference." arXiv preprint arXiv:1809.04191, Oct. 2018. DOI: https://doi.org/10.48550/arXiv.1809.04191

[17] Nagel, Markus, Amjad, Rabab A., Blankevoort, Tijmen, & Welling, Max. "Up or down? adaptive rounding for post-training quantization." International Conference on Machine Learning, PMLR, July 2020. DOI: 10.48550/arXiv.2004.10568

[18] Lin, Tsung-Yi, Maire, Michael, Belongie, Serge, Hays, James, Perona, Pietro, Ramanan, Deva, Dollar, Piotr, & Zitnick, C. Lawrence. "Microsoft coco: Common objects in context." Computer Vision–ECCV 2014: 13th European Conference, pp. 740-755, Zurich, Switzerland, Sep. 2014. DOI: https://doi.org/10.1007/978-3-319-10602-1_48

Authors

Du-Hwan Hur received the B.S. degree in Computer Engineering from Hanbat National University in 2021, and is currently pursuing the M.S. - Ph.D. integrated course degree with the Department of Electrical and Computer Engineering at Inha University, Korea. His current research interest is objection detection, multi-object tracking, and on-board AI.

Dae-Hyeon Park received the B.S degree in Computer Engineering from Inha University in 2020 and the M.S degree with the Department of Electrical and Computer Engineering at Inha University in 2023. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering at Inha University, Korea and His current research interests are single/multi-object tracking, multi-modal learning, real-time system and self-attention mechanism.

Deok-Woong Kim received the B.S. degree with Department of Industrial engineering, Public administration, Inha University, South Korea. He is currently pursuing the M.S. - Ph.D. integrated course degree. His research interests are knowledge distillation, quantization, data-free and on-board AI.

Jae-Yong Baek received the M.S degree in Computer Science and Engineering from Incheon Nation University, and worked as a software engineer that develop the software for object detection, lane detection and semantic segmentation in autonomous vehicle startup. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering at Inha University, Korea. His current research interests are object detection, generative model, quantization and optimization for edge AI.

Jun-Hyeong Bak received his B.S. degree in Electronic Engineering from Hannam University in 2020, and M.S. degree in Electrical and Computer Engineering from Inha University in 2022. He is currently a research engineer at PGM Seeker R&D, LIG Nex1 Co., Ltd. He is interested in fast visual object detection and tracking on low-power hardware.

Seung-Hwan Bae received the BS degree in information and communication engineering from Chungbuk National University, in 2009 and the MS and PhD degrees in information and communications from the Gwangju Institute of Science and Technology (GIST), in 2010 and 2015, respectively. He was a senior researcher at Electronics and Telecommunications Research Institute (ETRI) in Korea from 2015 to 2017. He was an assistant professor in the Department of Computer Science and Engineering at Incheon National University, Korea from 2017 to 2020. He is currently an Associate Professor with the Department of Electrical and Computer Engineering at Inha University, His research interests include object tracking, object detection, generative model learning, continual learning, on-device ML, etc.