https://doi.org/10.7840/kics.2025.50.11.1771

엣지 컴퓨팅 환경에서 딥러닝 객체 검출 및 객체 추적 기반 산업 현장 모니터링 시스템 구현

강 민 성◆, 임 영 철°

Implementation of an Edge Computing-Based Industrial Site Monitoring System Using Deep Learning for Object Detection and Tracking

Minsung Kang◆, Youngchul Lim°

요 약

본 논문에서는 YOLOv7 기반의 객체 검출 및 특징 임베딩 통합 네트워크를 엣지 컴퓨팅 환경에 최적화하고, 다중 객체 추적 기술을 적용하여 산업 현장을 모니터링할 수 있는 시스템을 제안한다. 다양한 산업 현장의 요구에 맞게 객체 검출 통합 네트워크를 세 가지 복잡도와 세 가지 해상도로 설계하였다. 총 9개의 스케일을 갖는 모델을 서버에서 학습하고, ONNX 모델 변환 및 TensorRT의 FP32, FP16, INT8 양자화 기반 최적화를 통해 Jetson Xavier 및 Nano 보드에 적용하였다. 객체 검출 박스의 IOU 유사도와 특징 임베딩 유사도를 기반으로 다중 객체 추적 기법을 개발하였다. 실험 결과, 객체 검출 기법은 Xavier 보드에서 INT8 연산 기준 100 FPS 이상의 실시간 추론이 가능했으며, Nano 보드에서도 최대 약 70 FPS의 성능을 나타냈다. 정확도 측면에서도 INT8 최적화 적용 후 평균 1% 이내의 성능 저하만 발생하였다. 다중 객체 추적은 MOTA 52.73% 성능을 나타냈으며, 추론 속도는 평균 1.93ms로 측정되었다. 다양한 스케일의 모델과 객체 검출 및 다중 객체 추적 기법을 기반으로, 산업 현장의 다양한 요구사항에 맞게 적용 가능한 모니터링 시스템을 개발하였다.

Key Words : Edge Computing, Object Detection, Multiple Object Tracking, Industrial Monitoring, Deep Learning Optimization

ABSTRACT

This paper proposes a monitoring system for industrial sites by optimizing a YOLOv7-based integrated network for object detection and feature embedding in edge computing environments and applying multi-object tracking techniques. The object detection network was designed with three levels of complexity and three resolutions to meet the diverse requirements of industrial environments. A total of nine scale models were trained on the server and deployed on Jetson Xavier and Nano boards after ONNX conversion and TensorRT-based optimization using FP32, FP16, and INT8 quantization. A multi-object tracking method was developed based on the IOU similarity of detection boxes and the similarity of feature embeddings. Experimental results showed that the object detection model achieved real-time inference at over 100 FPS on

※ 본 연구는 과학기술정보통신부에서 지원하는 DGIST 연구과제(25-IT-01) 및 연구과제(25-SENS2-05)의 지원으로 수행되었습니다.
◆ First Author : Division of Mobility Technology, DGIST, mskang@dgist.ac.kr, 정회원
° Corresponding Author : Division of Mobility Technology, DGIST, ninolyc@dgist.ac.kr, 정회원
논문번호 : 202504-099-A-RN, Received April 29, 2025; Revised June 12, 2025; Accepted June 14, 2025

the Xavier board with INT8 operations and up to approximately 70 FPS on the Nano board. In terms of accuracy, INT8 optimization led to an average performance degradation of less than 1%. The multi-object tracking achieved a MOTA of 52.73% with an average inference time of 1.93 ms. Based on models of various scales and object detection and tracking techniques, the proposed system is adaptable to the diverse monitoring needs of industrial environments.

Ⅰ. 서 론

최근 CCTV 산업은 인공지능 영상 감시 기술과 융합되며 고도화되고 있다. 특히 중대재해처벌법 시행 이후 산업 현장에서의 안전사고 예방이 주요 과제로 떠오르면서, 현장 내 CCTV 설치가 급증하고 있다. 기존의 단순 저장형 CCTV 시스템은 사고 발생 이후의 사후 분석에만 활용되었으나, 최근에는 실시간으로 위험을 감지하고 경보를 제공할 수 있는 지능형 CCTV로 진화하고 있다[1]. 이러한 변화는 객체 인식, 이상행동 탐지, 사고 예측 기능 등을 포함한 AI 기반 영상 분석 기술의 발전과 함께 이뤄지고 있으며, 이에 따라 산업 현장의 요구에 부합하는 고정밀 영상 분석 솔루션에 대한 수요도 지속적으로 증가하고 있다. 특히 다양한 각도의 다채널 카메라를 통해 넓은 공간을 동시에 모니터링하고, 이를 중앙 서버에서 통합적으로 관리함으로써 즉각적인 상황 대응이 가능해지고 있다. 이러한 시스템은 제조업뿐만 아니라 건설 현장, 플랜트, 항만, 공공 인프라 등 다양한 산업 및 기반 시설로 확산되고 있으며, 그 중요성은 점차 확대되고 있다.

그러나 영상 데이터를 모두 중앙 서버에서 처리하는 기존 방식은 카메라 채널 수가 증가할수록 네트워크 과부하, 분석 지연, 병목 현상 등의 문제가 발생하게 된다. 특히 이상 상황 발생 시 빠른 판단과 조치가 필요한 산업 현장의 특성상, 중앙 집중형 구조는 즉각적인 대응에 한계를 가지며, 시스템 전체의 신뢰성과 확장성을 저해할 수 있다. 이러한 문제를 해결하기 위한 대안으로 엣지 컴퓨팅(Edge Computing) 기술이 부상하고 있다. 엣지 컴퓨팅은 영상 수집 장치 자체에서 데이터를 선별·처리하여 실시간으로 반응함으로써, 네트워크 부담을 줄이고 분석 속도 및 응답성을 크게 향상시킨다. 예를 들어, 현장에서 이상행동이나 위험 요소가 탐지될 경우 즉시 알람을 송출하고, 필요한 정보만 중앙 서버로 전송함으로써 효율적 자원 운영이 가능하다. 최근에는 NVIDIA Jetson Xavier와 Nano와 같은 저전력 엣지 디바이스의 보급으로 인해[3], 영상 분석 시스템의 분산 구조 설계가 더욱 유연하고 정교하게 구성되고 있다. 이로 인해 기존 대비 빠르고 민첩한 반응 체계를 구축할 수 있게 되었으며, 감시 시스템의 전체적인 실용성과 확장성 또한 한층 강화되고 있다.

이러한 지능형 영상 감시 시스템의 핵심은 딥러닝 기반 객체 검출 및 객체 추적 기술이다. 특히 YOLO 계열은 단일 단계 추론 구조를 기반으로 하여 연산 속도가 빠르고, 경량화가 용이하며, 다양한 복잡도에 따라 산업 환경에 유연하게 적용할 수 있다는 장점이 있다[2,6,11]. YOLO와 같이 검증된 네트워크를 선택함으로써 하드웨어 호환성과 실시간성, 안정성을 확보할 수 있으며, 특히 연산 자원이 제한된 엣지 환경에 적합한 특성을 보인다. 객체 검출과 더불어 딥러닝 기반의 다중 객체 추적 기술 역시 산업 현장에서 중요성이 커지고 있으며, 실시간 추적 정확도를 유지하기 위해 다양한 경량화 기법과 후처리 알고리즘이 함께 적용되고 있다. 이러한 기술들은 모두 고정밀 연산을 필요로 하므로, 연산 자원이 제한된 환경에서는 효율적인 모델 최적화 및 컴파일이 반드시 필요하다.

본 논문에서는 고사양 PC 환경에서 학습된 YOLOv7[2] 기반의 객체 검출 및 특징 임베딩 통합 네트워크[12]를 복잡도와 해상도를 조절하여 9개의 스케일을 갖도록 설계하였다. ONNX(Open Neural Network Exchange)[4] 형식으로 모델을 변환하고, TensorRT[5]의 FP32, FP16, INT8 양자화 기반으로 최적화하여 NVIDIA Jetson Xavier 및 Nano 보드에 실시간 추론이 가능하도록 개발하였다. TensorRT는 NVIDIA에서 제공하는 고성능 추론 최적화 프레임워크로, 모델 구조를 단순화하고 연산 경로를 최적화함으로써 메모리 사용량을 줄이고 처리 속도를 크게 향상시킬 수 있다. 객체 검출 이후의 연속적인 대상 식별 및 행동 분석 등을 위해 객체 추적이 필수적이며, 실시간 처리가 가능한 객체 추적 기술을 개발하였다. 객체 검출 박스의 IOU 유사도[9]를 기반으로 객체 추적 매칭을 하였고, 추적 중 사라졌다가 다시 나타나는 객체에 대해서는 특징 임베딩 유사도[10]을 기반으로 재식별(Re-Identification)을 수행하였다. 산업 현장에서는 사용 환경, 해상도, 처리 속도 등의 조건이 다양하기 때문에, 상황에 맞는 최적의 모델을 선택하는 것이 중요하다. 이에 본 논문에서는 다양한 객체 검출 및 추적 모델에 대해 성능을 정량

그림 1. 엣지 컴퓨팅 환경에서 산업 현장 모니터링 시스템
Fig. 1. Industrial Site Monitoring System Based on Edge Computing.

적으로 비교·분석하고, 이를 바탕으로 그림 1과 같이 산업 현장 모니터링을 위한 적합한 모델을 제시한다.

Ⅱ. 본 론

2.1 다양한 스케일의 객체 검출 통합 네트워크

본 논문에서는 산업 현장 모니터링을 위하여 YOLOv7[2] 기반의 객체 검출 및 특징 임베딩 통합 네트워크[13]를 사용하였다. 통합 네트워크는 경량화 backbone을 기반으로 기존의 객체 검출 네트워크에 특징 임베딩 추출 서브 네트워크를 추가하여, 하나의 네트워크로 객체 검출과 특징 임베딩 추출이 가능한 one-stage 네트워크이다. 특징 임베딩은 검출된 박스 당 하나의 64채널 또는 128채널의 특징을 포함하고 있고, 이 특징을 기반으로 객체의 ID를 구분할 수 있다. 통합 네트워크[13]를 기반으로 산업 현장의 다양한 요구 사항에 대응하기 위하여 여러 스케일을 갖는 모델을 설계하였다. 기본 모델(Medium)[13]를 기반으로 각 레이어의 채널수를 절반으로 줄인 Small 모델, 그리고 채널수를 2배로 확장한 Large 모델을 설계하였다. 그리고 입력 해상도를 320x320, 480x480, 640x640로 설정하여 총 9개의 스케일의 갖는 통합 모델을 설계하였다. 통합 네트워크의 기존 Darknet 프레임워크기반 학습 방식[6]은 검출 대상이 아닌 모든 샘플을 음성(negative) 샘플로 간주하여 클래스 불균형 문제를 유발하고, 학습 안정화에도 시간이 오래 걸린다. 이를 해결하기 위해, 본 논문에서는 online hard negative mining 기법[7]을 적용하여 양성 샘플 비율에 맞춰 어려운 음성 샘플(hard negatives)만 선택해 학습에 사용함으로써 정확도 향상과 수렴 속도 개선을 달성하였다. 이를 통해 산업 현장에서 객체 검출과 식별을 통합적으로 처리하는 실시간 시스템 구현이 가능하다.

2.2 통합 네트워크 ONNX 모델 변환

객체 검출 및 특징 임베딩 통합 네트워크는 Darknet 프레임워크에서 학습되었으며, Windows 환경에서 C/C++ 기반으로 개발·테스트되었다. 그림 2와 같이 엣지 컴퓨팅 환경에서 실시간 실행하기 위해 TensorRT 최적화를 위한 ONNX 형식으로 변환이 필요하다. Linux 기반 Pytorch 환경에서 ONNX로 변환하기 위해, 기존 프레임워크의 모델 구조 텍스트 파일과 weight 이진 파일을 변환하였다. 이를 위해 개발된 모델 컨버터는 Darknet 텍스트 구조를 읽어 PyTorch nn 모듈 함수(예: Conv2d, MaxPool2d, Leaky ReLU 등)로 변환하였다. Darknet의 route layer는 torch.cat 함수를 사용해 feature map을 결합하였고, YOLO head도 Python 기반 모듈로 변환하였다. Conv 계층의 weight, bias, mean, var 값은 binary weight 파일에서 읽어와 PyTorch 형식에 맞게 적용하였다. 변환이 완료된 PyTorch 모델은 ONNX 형식으로 내보냈으며, 입력/출력 형식은 PyTorch 모델과 동일하게 유지하였고, opset_version은 11을 사용하였다. 최종적으로, 복잡도 3종(S, M, L)과 입력 해상도 3종(320, 480, 640)을 조합하여 총 9개의 ONNX 모델을 생성하였다.

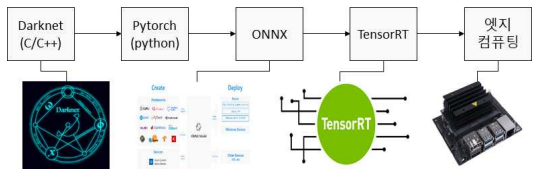

그림 2. 엣지 컴퓨팅 환경에서 딥러닝 모델 변환 과정
Fig. 2. Deep Learning Model Conversion Workflow for Edge Computing.

2.3 임베디드 보드에서 TensorRT 모델 생성

엣지 컴퓨팅 환경으로 NVIDIA Jetson Xavier와 Nano 보드를 사용하였다. 이들 보드는 일반 PC 대비 GPU 성능이 수십에서 수백 배 낮기 때문에, ONNX 모델을 TensorRT로 최적화하여 실시간 처리를 가능하게 하였다. 모든 모델은 FP32, FP16, INT8 정밀도로 변환하였으며, 특히 INT8은 정확도 하락을 방지하기 위해 calibration 과정을 적용하였다. COCO 데이터셋[8]과 Polygraphy Calibrator를 사용하여 수백 장의 이미지로 보정된 TensorRT 엔진을 생성하였다. Jetson Xavier는 1,000장 단위로 cache를 생성했고, Nano 보드는 메모리 제한으로 100장씩 분할 보정을 수행하였다. 결과적으로, 그림 3과 같이 9개의 ONNX 모델에 대해 FP32, FP16, INT8 변환을 적용하여 총 27개의 TensorRT 엔진을 생성하였다. 객체 검출 정확도와 추

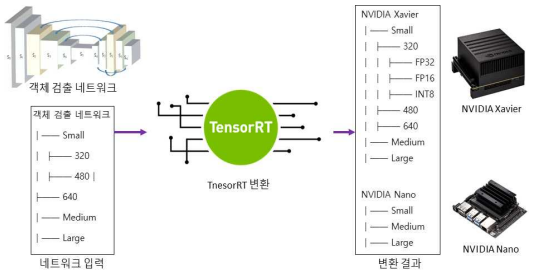

그림 3. 객체 검출 모델을 엣지 컴퓨팅 환경에 변환 결과
Fig. 3. Results of Deploying Object Detection Model on Edge Device.

론 속도를 기준으로, 산업 현장에서 사용 환경에 맞는 최적의 모델 선택이 가능하도록 하였다.

2.4 통합 네트워크 기반 다중 객체 추적 기법

TensorRT로 최적화된 객체 검출 및 특징 임베딩 통합 네트워크를 기반으로 다중 객체 추적(Multiple Object Tracking) 기법을 개발하였다. 실시간 추적 기법인 SORT[9] 기법의 IOU 유사도와 딥 특징 임베딩 유사도[10]을 사용하였다. IOU 유사도는 현재 프레임과 이전 프레임의 객체 검출 박스의 겹침 정도를 계산하고, 이를 기반으로 동일한 객체인지 판단하기 때문에 계산량이 적어 실시간 시스템에 적합하다. 하지만 추적 중 가려짐 등에 의해 객체가 검출되지 않고, 일정 영역을 벗어나서 다시 검출됐을 경우 재식별이 불가능하다. 딥 특징 임베딩 유사도는 검출 박스에서 64채널 또는 128채널과 같이 특징을 추출하고, 이를 기반으로 동일한 객체인지 판단한다. 하지만 유사 객체가 많을 경우 신뢰도가 떨어지는 단점이 있다.

본 논문에서는 다중 객체 추적을 위해 두 단계의 연속적인 데이터 연관 전략을 적용한다. 먼저, 프레임 간 검출된 객체들 간의 위치 기반 유사도를 판단하기 위해 IOU 유사도를 계산한다. 현재 프레임의 각 검출 박스와 이전 프레임에서 추적된 객체들의 경계 상자 간 IOU를 산출한 후, 사전에 설정한 IOU 임계값(threshold) 이상인 경우에만 객체 ID를 유지한다. 객체의 이동 경로가 연속적이며 겹침이 충분히 큰 경우에 효과적으로 동작한다. 그러나 객체 간의 IOU가 임계값 미만으로 낮아지거나, 일시적으로 객체가 가려져 위치 기반 연관이 어려운 경우에는, 두 번째 단계로 특징 임베딩을 활용한 재식별 유사도 기반 매칭을 수행한다. 이를 위해 각 객체에 대해 딥러닝 기반 임베딩 벡터를 추출하고, 이전 프레임에서 추적되었으나 현재 IOU 기반으로 매칭되지 않은 객체들과의 코사인 유사도(cosine similarity) 또는 거리(metric distance)를 계산하여 가장 유사한 쌍을 매칭한다. 동일한 객체의 외형적 특징이 시간 경과 후에도 일정 부분 유지된다는 가정 하에, 단기적인 occlusion이나 프레임 손실 상황에서도 안정적인 추적이 가능하다. 이와 같이 IOU 기반의 유사도와 특징 임베딩 기반 유사도를 함께 고려하여 산업 현장의 다양한 환경에서도 견고하게 다중 객체 추적이 가능하다.

Ⅲ. 실 험

3.1 실험환경

엣지 컴퓨팅 환경에서 최적화한 객체 검출 및 특징 임베딩 통합 네트워크의 정확도와 추론 속도 평가를 위하여 COCO 데이터셋을 사용하였다. 객체 검출 COCO 데이터셋은 82,783장의 학습 이미지와 40,504장의 평가 이미지로 구성되어 있고, 클래스 개수는 사람, 동물, 차량, 가전제품, 식품 등 80개로 구성되어 있다. 본 논문에서는 기존 COCO 데이터셋의 다양한 클래스에서 산업 현장에서 모니터링을 위하여 검출하여야 하는 클래스 및 산업 현장 별 레이블링 비용을 고려하여 5개의 클래스로 축소하였다. 산업 현장 내에서 모니터링이 필요한 작업자와 산업 현장 외에서 모니터링이 필요한 이동수단으로 클래스를 축소하였다. 사람, 자동차, 버스, 자전거, 오토바이로 5개의 클래스로 축소하였고, 변경된 클래스 개수에 맞게 학습 및 평가 이미지도 라벨이 존재하는 이미지만 추출하였다. 변경한 커스텀 데이터셋은 48,674장의 학습 이미지와 23,504장의 평가 이미지로 구성하였고, 클래스 개수는 5개로 설정하였다. 객체 검출 딥러닝 네트워크는 NVIDIA RTX 3090 그래픽카드를 탑재한 딥러닝 서버에서 학습하였고, 이를 NVIDIA Jetson Xavier와 NVIDIA Jetson Nano 보드로 컨버팅 및 최적화를 수행하였다.

3.2 객체 검출 추론 속도 비교 분석

엣지 컴퓨팅 환경에서의 실시간 추론 성능을 분석하기 위해, 각 객체 검출 모델에 대해 데이터셋의 평가 이미지 전체에 대한 평균 추론 속도를 측정하였다. 측정에는 딥러닝 네트워크의 추론 시간과 NMS(Non Maximum Suppression)에 소요되는 시간까지 포함하였다. 실험은 NVIDIA RTX 3090 GPU 기반 학습 서버, Jetson Xavier, Jetson Nano 보드에서 각각 수행하였으며, 모델은 복잡도(S, M, L) 및 입력 해상도(320, 480, 640)에 따른 총 9개 조합에 대해 측정하였다. 표 1은 일반 PC 환경(NVIDIA RTX 3090 GPU)에서 각 모델의 추론 속도와 연산량(BFLOPs)을 비교한 결과를 나타낸다. BFLOPs가 가장 낮은 Small 모델(320 입력)

표 1. NVIDIA RTX 3090에서 객체 검출 추론 속도 결과
Table 1. Object Detection Inference Speed on NVIDIA RTX 3090.

Model	Input Size	BFLOPs	Inference time(ms)
S	320	0.67	3.5
	480	1.51	4.5
	640	2.69	4.7
M	320	2.62	4.7
	480	5.90	5.0
	640	10.49	6.1
L	320	10.36	5.5
	480	23.30	5.7
	640	41.43	8.6

의 경우 평균 3.5ms로 약 285 FPS의 성능을 보였고, 가장 복잡한 Large 모델(640 입력)은 8.6ms로 약 116 FPS를 기록하였다. 전반적으로 RTX 3090 환경에서는 모델의 복잡도 및 입력 해상도와 무관하게 전 모델이 100 FPS 이상의 실시간 추론이 가능하다.

표 2는 Jetson Xavier 보드에서 측정한 최적화 전후 추론 속도 비교 결과를 나타낸다. 먼저, 기존 PC 환경에서 학습된 모델을 최적화 없이 실행한 경우, Small (320 입력)는 32.76ms(약 30 FPS), Large (640 입력)는 58.49ms(약 17 FPS)로 측정되어 실시간 처리에 어려움이 있다. 이에 따라 TensorRT 기반 최적화와 INT8 정밀도 보정(calibration)을 수행하여 FP32, FP16, INT8 모델에 대해 추론 속도를 비교하였다. S 모델(320 입력)은 정밀도 별 속도 차이가 크지 않았지만, 모델 복잡도가 증가할수록 최적화의 효과가 뚜렷하게 나타났다. 특히 INT8 연산은 가장 높은 속도를 기록하였으며, 모델에 따라 최소 약 5.6배에서 최대 약 8.8배까지 성능 향상되었다. Jetson Xavier 보드에서는 최적화를 통해 대부분 모델이 100 FPS 이상 실시간 추론이 가능하였다.

표 2. NVIDIA Jetson Xavier 보드에서 객체 검출 추론 속도 결과
Table 2. Object Detection Inference Speed on NVIDIA Jetson Xavier Board.

Model	Input Size	BFLOPs	Inference time(ms)			
			FP32	FP32 (RT)	FP16 (RT)	INT8 (RT)
S	320×320	0.672	32.76	3.85	3.97	3.95
	480×480	1.512	33.99	5.35	4.97	4.73
	640×640	2.689	33.06	7.27	6.15	5.94
M	320×320	2.62	33.30	5.86	3.86	3.80
	480×480	5.90	32.36	8.93	5.48	4.60
	640×640	10.49	34.36	15.60	7.92	5.98
L	320×320	10.356	33.38	13.91	5.60	4.18
	480×480	23.302	37.05	23.91	9.28	6.46
	640×640	41.425	58.49	40.87	15.36	9.59

표 3은 Jetson Nano 보드에서 최적화 전후의 추론 속도 비교 결과를 나타낸다. Nano 보드에서는 Xavier 보드에 비해 전반적인 추론 속도가 느리게 측정되었으나, FP16 정밀도에서 가장 빠른 추론 속도를 나타내었고, 최적화에 따른 성능 향상 폭은 모델에 따라 최소 1.9배에서 최대 4.7배 수준이었다. 최적화 적용 이후 최대 약 70 FPS의 실시간 추론이 가능하였으며, 이는 Xavier 보드 대비 낮은 연산 자원에도 불구하고 실시간 처리가 가능함을 의미한다.

표 4는 사용된 세 가지 GPU 플랫폼의 아키텍처와 연산 자원 구성을 비교한 결과를 나타낸다. RTX 3090은 Ampere 아키텍처를, Jetson Xavier는 Volta 아키텍처를, Jetson Nano는 Maxwell 아키텍처를 기반으로 하

표 3. NVIDIA Jetson Nano 보드에서 객체 검출 추론 속도 결과
Table 3. Object Detection Inference Speed on NVIDIA Jetson Nano Board.

Model	Input Size	BFLOPs	Inference time(ms)			
			FP32	FP32 (RT)	FP16 (RT)	INT8 (RT)
S	320×320	0.672	60.08	14.43	12.69	14.49
	480×480	1.512	62.10	27.40	21.10	28.37
	640×640	2.689	66.08	44.10	34.12	42.30
M	320×320	2.62	62.88	28.80	19.27	28.07
	480×480	5.90	80.10	51.88	35.39	50.38
	640×640	10.49	121.64	87.19	58.09	82.42
L	320×320	10.356	113.03	75.23	45.79	74.63
	480×480	23.302	194.49	136.64	83.06	132.87
	640×640	41.425	323.36	249.50	144.93	242.25

표 4. 일반 GPU와 임베디드 보드 스펙 비교
Table 4. Specification Comparison Between General GPUs and Embedded Boards.

Specs	NVIDIA RTX 3090	NVIDIA Jetson Xavier	NVIDIA Jetson Nano
Architecture	Ampere	Volta	Maxwell
CUDA core	10,496	512	128
Tensor core	328	64	X
Power consumption	350W	30W	5-10W

며, 각 아키텍처는 FP16 및 INT8 연산 지원 수준에서 본질적인 차이를 가진다. 특히 Nano 보드의 Maxwell 아키텍처는 INT8 연산을 소프트웨어적으로 지원하지만, 하드웨어 차원의 최적화(Tensor Core 등)가 적용되지 않아 INT8 연산 성능 향상이 극히 제한적이다. 이와 같은 하드웨어 및 아키텍처 차이는 동일한 모델이라 하더라도 추론 속도, 최적화 효과, 실시간 처리 가능성에 큰 차이를 발생시키며, 엣지 디바이스 특성을 고려한 모델 경량화 및 최적화 전략이 필요하다.

표 5와 표 6은 Jetson Xavier 및 Jetson Nano 보드에서, 특징 임베딩 통합 네트워크의 추론 성능을 측정한 결과를 나타낸다. 통합 네트워크는 기존 객체 검출 네트워크에 추가적인 특징 임베딩 서브 네트워크를 결합하여, 객체 검출과 함께 고차원 임베딩 특징 추출이 가능한 구조로 설계되었다. Jetson Xavier 보드에서 기존 객체 검출 네트워크와 통합 네트워크 간의 성능 비교 결과, FP32 환경에서는 small 모델 기준 5.35ms에서 10.62ms로 약 2배 가까이 추론 시간이 증가하였지만, INT8 정밀도에서는 거의 동일한 수준(약 4.6ms)으로 추론 속도를 유지하였다. 이는 TensorRT 기반 INT8 최적화가 통합 네트워크 구조에서도 효과적으로 적용되었음을 보여준다. 반면, Nano 보드에서는 FP32 환경에서 통합 네트워크 적용 시 평균 2배 가까운 추론 시간 증가가 발생하였다. 그러나 INT8 정밀도에서는 임베딩 차원이 64인 경우 기존 네트워크와 유사한 수준을 유지했으며, 128차원의 경우에도 모델에 따라 약 1.07배~1.25배의 비교적 안정적인 추론 시간이 측정되었다. 특징 임베딩 통합 네트워크는 차원 수에 따라 연산 복잡도가 증가하지만, Xavier 보드에서는 대부분 실시간 추론 성능을 유지할 수 있었으며, Nano 보드에서는 모델 복잡도, 입력 해상도, 임베딩 차원 수 등에 따라 적절한 구성 선택이 필요하다.

표 5. NVIDIA Jetson Xavier 보드에서 통합 네트워크 추론 속도 비교
Table 5. Inference Speed Comparison of the Integrated Network on NVIDIA Jetson Xavier Board

Model	Input Size	Channels	Inference time(ms)		
			FP32 (RT)	FP16 (RT)	INT8 (RT)
S	480×480	64	9.62	5.90	4.46
		128	10.62	6.20	4.63
M	480×480	64	9.82	5.18	4.34
		128	10.58	5.05	4.32
L	480×480	64	24.24	9.89	6.74
		128	25.14	10.16	6.93

표 6. NVIDIA Jetson Nano 보드에서 통합 네트워크 추론 속도 비교
Table 6. Inference Speed Comparison of the Integrated Network on NVIDIA Jetson Nano Board

Model	Input Size	Channels	Inference time(ms)		
			FP32 (RT)	FP16 (RT)	INT8 (RT)
S	480×480	64	29.40	22.55	28.65
		128	34.40	27.98	35.07
M	480×480	64	54.52	37.41	54.15
		128	61.39	40.33	60.55
L	480×480	64	139.96	87.22	139.27
		128	145.91	87.89	142.02

3.3 객체 검출 정확도 성능 비교 분석

표 7은 NVIDIA Jetson Xavier 보드에서 TensorRT 최적화 수행 후 정밀도별(FP32, FP16, INT8) 객체 검출 정확도(mAP)를 비교한 결과를 나타낸다. 일반적으

표 7. NVIDIA Xavier 보드에서 객체 검출 모델 최적화에 따른 정확도 성능 비교 결과
Table 7. Accuracy Performance Comparison of Optimized Object Detection Models on NVIDIA Xavier Board.

Model	Input size	BFLOPs	mAP(%)			
			FP32	FP32 (RT)	FP16 (RT)	INT8 (RT)
				ACC±	ACC±	ACC±
S	320×320	0.672	32.76	45.66	45.67	43.78
				-	0.01	-1.89
	480×480	1.512	33.99	54.13	54.12	53.53
				-	-0.01	-0.59
	640×640	2.689	33.06	60.87	60.88	59.97
				-	0.01	-0.91
M	320×320	2.62	33.30	65.86	65.89	65.11
				-	0.02	-0.78
	480×480	5.90	32.36	66.44	66.46	66.11
				-	0.02	-0.35
	640×640	10.49	34.36	66.60	66.53	65.72
				-	-0.07	-0.81
L	320×320	10.356	33.38	68.98	68.94	68.52
				-	-0.04	-0.42
	480×480	23.302	37.05	70.84	70.80	69.69
				-	-0.04	-1.11
	640×640	41.425	58.49	72.19	72.13	71.27
				-	-0.07	-0.85

로 엣지 컴퓨팅 환경에서는 실시간 처리를 위해 FP16 및 INT8 양자화 연산을 적용하지만, 이 과정에서 비트 표현 정밀도의 감소에 따른 양자화 오류가 발생할 수 있다. 본 논문에서는 양자화에 따른 정확도 저하의 영향을 정량적으로 분석하였다. S, M, L 모델 및 입력 해상도(320, 480, 640)의 조합에 대해, COCO 데이터셋에서 검출 정확도를 평가하였다. 그 결과, FP32 모델과 FP16 모델 간 정확도 차이는 -0.07%에서 +0.02% 수준으로 거의 유사하였다. INT8 모델의 경우 양자화 오류가 누적되며 평균 0.35%에서 최대 1.89%까지 정확도 감소가 발생하였다. 가장 큰 정확도 손실은 Small 모델(320 입력)에서 나타났으며, 복잡도가 높은 모델일수록 상대적으로 정확도 하락 폭은 작아지는 경향을 보였다. 전반적으로, INT8 연산은 추론 속도에서 최대 6~8배 향상을 보이면서도, 객체 검출 정확도는 대부분 모델에서 1% 이내로 하락하여, 실시간 엣지 컴퓨팅 환경에서도 충분히 수용 가능한 수준의 성능을 유지함을 확인할 수 있었다. 따라서, 객체 검출 모델의 경우 INT8 최적화를 적용하더라도 정확도 대비 추론 속도 향상을 확인할 수 있었다.

3.4 산업 현장 모니터링 활용 테스트

딥러닝 서버에서 학습된 객체 검출 및 특징 임베딩 통합 네트워크를 엣지 컴퓨팅 환경으로 변환하고 최적화하여, 산업 현장 모니터링에 적용하였다. 작업자가 활동하는 위험 구역에 IP 카메라를 설치하고, RTSP (Real-Time Streaming Protocol)를 통해 엣지 디바이스에서 실시간 영상을 수신하여 객체 검출을 수행하였다. 객체 검출을 통해 작업자의 위치를 탐지함으로써, 특정 위험 구역 내 작업자의 존재 유무를 실시간으로 판단할 수 있다. 그러나 객체 검출만으로는 작업자의 이동 경로 추적이나 행동 분석이 제한되므로, 다중 객체 추적 기술

그림 4. IOU 매칭 기반 객체 추적 결과
Fig. 4. Object Tracking Results Using IoU-Based Matching.

그림 5. IOU 매칭과 Re-ID 매칭 기반 객체 추적 결과
Fig. 5. Object Tracking Results Using IoU and Re-Identification Matching.

이 필요하다. IOU 유사도 기반의 다중 객체 추적 방식은 계산량이 적고 속도가 빠르지만, 그림 4와 같이 작업자가 일시적으로 가려지거나 겹치는 상황에서는 ID 스위칭 현상이 발생하는 단점이 있다. 특징 임베딩 기반 유사도는 객체 추적의 경우 재식별이 가능하지만 유사한 객체에 대한 신뢰도가 떨어지기 때문에 IOU 유사도와 특징 임베딩 유사도를 모두 사용하였다. 그림 5와 같이 유사한 작업복을 입은 사람들을 그대로 추적하면서, 작업자가 다른 작업도구에 가려졌다가 다시 나타났을 경우에도 재식별을 통하여 지속적으로 추적이 가능하였다. 산업 현장에서 작업자가 특정 구간에 존재하는지 검출하고, 추적을 통하여 이동경로를 파악할 수 있는 엣지 컴퓨팅 환경에서의 작업자 모니터링 기술을 개발하였다.

3.5 다중 객체 추적 정확도 성능 비교 분석

다중 객체 추적 정확도와 테스트 속도 평가를 위하여 MOT 17 dataset[12]을 사용하였다. COCO dataset에서 학습한 통합 네트워크를 기반으로 MOT 17 dataset의 09 시퀀스에서 사람 클래스에 대한 다중 객체 추적을 테스트 하였고, CLEAR metric[12]로 정확도를 평가하였다. 표 8과 같이 Small, Medium, Large 모델에서 각각 MOTA 정확도가 48.23%, 52.73%, 51.74로 나타냈다. 테스트 속도는 500장 이상의 이미지에서 다중 객체 추적 기법을 테스트하여 평균 시간을 측정하였을 때 평균 1.93ms로 나타났다. 다중 객체 추적 정확도는 객체 검출 박스와 특징 임베딩에 영향을 많이 받기 때문에 산업 현장의 테스트 환경에서 영상을 수집하고 파인튜닝을 수행하면 모니터링 시스템의 정확도를 효과적으로 향상시킬 수 있다. 또한 NVIDIA Xavier 보드에서 제일 복잡도가 높은 모델에서 다중 객체 추적 테스트 시간을 포함하여도 10ms 이하로 100 FPS 이상 실시간 모니터링이 가능하다.

표 8. NVIDIA Jetson Xavier 보드에서 다중 객체 추적 정확도 결과
Table 8. Evaluation results of multi-object tracking accuracy on the NVIDIA Jetson Xavier platform

Model	Input Size	MOTA(%)
S	480×480	48.23
M	480×480	52.73
L	480×480	51.74

Ⅳ. 결 론

본 논문에서는 객체 검출 및 특징 임베딩 통합 네트워크를 기반으로 다중 객체 추적 기술을 적용하고, 엣지 컴퓨팅 환경에서 산업 현장의 다양한 요구사항을 반영할 수 있는 모니터링 시스템을 제안하였다. 산업 현장의 다양한 요구를 반영하기 위해, 네트워크 복잡도(S, M, L)와 입력 해상도(320, 480, 640)에 따라 다양한 스케일의 네트워크를 설계하고, 학습 서버에서 훈련된 모델을 엣지 컴퓨팅 환경에 적용 가능하도록 ONNX 변환 및 TensorRT 기반 최적화를 수행하였다. 특히, Jetson Xavier 및 Jetson Nano 보드와 같은 저전력 임베디드 GPU 환경에서 실시간 처리를 위해 FP32, FP16, INT8 정밀도별 최적화 모델을 생성하고 비교 분석하였다. 실험 결과, Jetson Xavier에서는 INT8 연산 기준 100 FPS 이상의 실시간 객체 검출이 가능하였으며, Nano 보드에서도 최대 약 70 FPS 성능을 확인하였다. 정확도 측면에서는 INT8 양자화로 인해 평균 1% 내외의 성능 저하가 발생했으나, 추론 속도 증가에 비해 성능 손실이 매우 적었다. 또한, 객체 검출뿐 아니라 다중 객체 추적 기법도 엣지 컴퓨팅 환경에서 구현하기 위해, 두 단계의 연속적인 데이터 연관 전략을 적용하였다. 먼저, 프레임 간 검출된 객체들의 IOU 유사도를 계산하고, 매칭 시 추적 ID를 부여하였다. 추적 중인 상태에서 IOU 매칭이 되지 않을 경우 특징 임베딩 유사도를 기반으로 재식별 매칭을 수행하였다. Jetson Xavier 보드의 복잡도가 가장 높은 모델을 기준으로 객체 검출 및 다중 객체 추적 기반으로 100 FPS 이상 실시간 모니터링이 가능한 시스템을 제안하였다. 향후 연구에서는 보다 다양한 엣지 플랫폼에서 성능 최적화 및 확장성을 검증하고, 다양한 산업 현장 데이터셋 구축, 그리고 실제 어플리케이션 적용을 통한 산업 현장 맞춤형 지능형 영상 분석 솔루션 개발을 지속적으로 추진할 계획이다.

References

[1] M. Hu, Z. Luo, A. Pasdar, Y. C. Lee, Y. Zhou, and D. Wu, "Edge-based video analytics: A survey," *IEEE/ACM Trans. Multimed. Comput. Commun. Appl.*, vol. 25, no. 4, pp. 2951-2982, Apr. 2023. (https://doi.org/10.48550/arXiv.2303.14329)

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

detectors," in *Proc. IEEE/CVF Conf. CVPR*, pp. 7464-7475, Jun. 2023. (https://doi.org/10.1109/CVPR52729.2023.00721)

[3] D. J. Shin and J. J. Kim, "A deep learning framework performance evaluation to use YOLO in Nvidia Jetson platform," *Appl. Sci.*, vol. 12, no. 8, Art. 3734, Apr. 2022. (https://doi.org/10.3390/app12083734)

[4] ONNX Runtime developers, "ONNX: Open neural network exchange for deep learning framework interoperability," *GitHub repository*, 2017. Retrieved Jun. 12, 2025, from https://github.com/onnx/onnx

[5] NVIDIA, "TensorRT: High performance deep learning inference on NVIDIA GPUs," *GitHub repository*, Retrieved Jun. 12, 2025, from https://github.com/NVIDIA/TensorRT

[6] A. Bochkovskiy, C.-Y. Wang, and H. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," *IEEE Access*, vol. 8, pp. 153-160, 2020. (https://doi.org/10.48550/arXiv.2004.10934)

[7] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proc. IEEE Conf. CVPR*, pp. 761-769, Jun. 2016. (https://doi.org/10.1109/CVPR.2016.89)

[8] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Proc. ECCV*, pp. 740-755, Zurich, Switzerland, Sep. 2014. (https://doi.org/10.1007/978-3-319-10602-1_48)

[9] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *Proc. IEEE ICIP*, pp. 3464-3468, Sep. 2016. (https://doi.org/10.1109/ICIP.2016.7533003)

[10] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *Proc. IEEE ICIP*, pp. 3645-3649, Beijing, China, Sep. 2017. (https://doi.org/10.1109/ICIP.2017.8296962)

[11] J. P. Park, S. G. Park, J. H. Shin, I. Bang, and T. Kim, "Performance comparison of object detection based on edge computing in drone networks," *J. KICS*, vol. 45, no. 9, pp. 123-135, Sep. 2020.

[12] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler, "MOTChallenge 2015: Towards a benchmark for multi-target tracking," in *Proc. IEEE ICCVW*, pp. 1-8, Dec. 2015. (https://doi.org/10.48550/arXiv.1504.01942)

[13] Y. C. Lim and M. Kang, "One-stage object detection and feature embedding network for multiple object tracking," in *Adv. Comput. Sci. Ubiquitous Comput. - Proc. CUTE/CSA 2023* (LNEE), vol. 1190, pp. 420-425, Singapore, Dec. 2023. (https://doi.org/10.1007/978-981-97-2447-5_66)

강 민 성 (Minsung Kang)

2012년 2월 : 영남대학교 정보통신공학과 학사
2014년 2월 : 영남대학교 정보통신공학과 석사
2014년 3월~현재 : DGIST 미래모빌리티연구부 전임연구원
<관심분야> 딥러닝, 객체 검출 및 추적, 이상 상황 감지
[ORCID:0000-0003-4018-8514]

임 영 철 (Youngchul Lim)

1999년 2월 : 경북대학교 전자전기공학부 학사
2005년 2월 : 경북대학교 전자전기공학부 석사
2013년 3월 : 경북대학교 전자전기공학부 박사
2005년 6월~현재 : DGIST 미래모빌리티연구부 책임연구원
<관심분야> 딥러닝, 객체 검출, 다중 객체 추적
[ORCID:0009-0003-5229-360X]