논문 20-45-04-15 The Journal of Korean Institute of Communications and Information Sciences '20-04 Vol.45 No.04
https://doi.org/10.7840/kics.2020.45.4.722

자율주행 환경에서 딥러닝 기법의 객체검출과 속도 성능 최적화

김 영 준◆, 황 혜 경*, 신 지 태°

Optimization of Object Detection and Inference Time for Autonomous Driving

Youngjun Kim◆, Hyekyoung Hwang*, Jitae Shin°

요 약

자율주행 환경에서의 객체검출 문제를 풀기 위해 본 연구에서는 딥러닝 기반의 객체검출기를 Stem Block, Backbone Network, Detector, Extra Layer의 4개의 영역으로 분리 후, 각 계층별로 여러 딥러닝 최적화 기법을 적용하여 계산 복잡도 대비 풍부한 Receptive Filed를 통해 모델의 정확도와 추론 시간을 효율적으로 trade-off 하였다. 이를 통해 자율주행 환경에서의 정확한 위치 검출과 클래스 분류 성능을 보이는 딥러닝 기반의 객체검출기를 설계하였다. 이를 SSD 계열의 State-of-the-art Model인 M2Det과 자율주행 환경에서 정확도 및 속도를 비교하였을 때, 1.4%의 mAP의 차이로, 1.9배 빠른 실시간 객체검출기를 구현하였다.

Key Words : Autonomous Driving, Object Detection, Deep Learning Model Optimization

ABSTRACT

In order to solve the problem of object detection in autonomous driving environment, the Deep Learning-based Object Detector was separated into four areas: Stem Block, Backbone Network, Detector, and Extra Layer, and several deep learning optimization techniques were applied to each layer. The accuracy of the model and the Inference Time were conducted cost-effectively through the rich Recipient Filed compared to the computational complexity. This allows the autonomous in the environment, classification performance and accurate localization dnn based object detector the design. When comparing accuracy and speed in an autonomous driving environment with M2Det, a state of the art model of SSDs, the real-time object detector was 1.9 times faster, with a 1.4% difference in mAP.

Ⅰ. 서 론

객체검출 문제는 컴퓨터비전 분야의 전통적인 도전과제다. 최근에는 보안, 서비스, 공장자동화 등 많은 분야에서 접목되어 발전하고 있으며, 특히 자율주행 환경에서 또한 중요성이 높아지고 있다. 특히 GPU의

※ 이 논문은 2020년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업(No. 2017R1D1A1B03031752)와 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터지원사업의 연구결과로 수행되었음 (IITP-2020-2018-0-01798).

◆ First Author : Department of Electrical and Computer Engineering, Sungkyunkwan University, yjk931004@skku.edu, 학생(박사), 학생회원

° Corresponding Author : School of Electronic and Electrical Engineering, Sungkyunkwan University, jtshin@skku.edu, 정교수, 정회원

* Department of Electrical and Computer Engineering, Sungkyunkwan University, ristar1234@skku.edu, 학생(석사), 학생회원

논문번호 : 201911-084-C-RN, Received November 21, 2019; Revised January 1, 2020; Accepted January 22, 2020

발달과 함께 인공지능 연구가 빠르게 진행되고 있는 가운데, 자율주행 차량 또한 빠르게 발전하고 있다[1]. 본 연구에서는 카메라센서를 통해 자율주행 환경에서의 객체검출 문제를 풀어보고자 한다.

딥러닝 기반의 객체검출기가 자율주행 환경에서 잘 동작하기 위해서는 몇 가지 요구 사항이 있다. 첫 번째, 자율주행 환경에서의 다양한 상황에 즉각적으로 대처하기 위해 실시간성을 보장해야 한다. 둘째, 자율주행 환경에서의 정확한 위치 검출과 클래스 분류를 보장해야 한다. 그러나 실시간성과 정확성은 trade-off의 관계이기 때문에 둘 다 향상하는 것은 어려운 도전과제다. 따라서 본 연구에서는 위의 두 요구 사항을 만족하기 위해 비용대비 효율적으로 trade-off를 하여 빠르면서도 정확한 모델을 만들고자 하였다. 본 연구에서는 이를 달성하기 위해 계산 복잡도 대비 수용체(Receptive Field)가 풍부한 모델을 만드는 것을 목표로 하였으며, 이러한 목표를 달성하기 위해 3가지 관점에서 접근하였다. 첫째, 모델의 효율성이다. 계산 복잡도 대비 높은 수용체를 가져갈 수 있는 작은 딥러닝 모델을 도입하여 효율성 관점에서 최적화하였다. 둘째, 모델의 추론 속도를 향상하기 위해 하드웨어 및 프레임워크 관점에서 최적화를 진행하였다. 셋째, 객체검출은 클래스 분류문제와 다르다. 따라서 객체검출 관점에서 최적화하였다.

따라서 이러한 연구의 기여도는 아래와 같다. 첫째, 객체검출기의 백본망(Backbone Network)를 최적화 기법을 통해 파라미터 대비 계산 복잡도의 효율성이 약 2..15 배 높은 객체검출기를 설계하였다. 둘째, SOTA 모델인 M2Det과 자율주행 환경에서의 성능 비교를 하였을 때, mAP가 약 1.4% 차이나지만 속도가 1.9배 빠른 실시간 객체검출기 구현하였다. 셋째, 객체검출기의 하이퍼 파라미터 튜닝을 통해 가볍고 빠른 모델부터 정확한 모델까지 일반화하여 제공한다.

Ⅱ. 관련 연구

최근 딥러닝 기반의 객체검출기의 경우 크게 2가지 방향으로 발전하고 있다. 첫째는 RPN(Region Proposal Network)을 기반으로 하는 2-Stage 검출기다[2]. 2-Stage 검출기는 높은 정확도를 갖고 있으나, Region Proposal과 클래스 분류를 순차적으로 수행하기 때문에 속도가 느리다는 한계를 갖는다. 둘째는 이러한 2-Stage 검출기의 문제를 해결하기 위해 RPN을 사용하지 않고, Region Proposal과 클래스 분류를 동시에 수행하도록 만든 1-Stage 검출기다[3,4]. 따라서 실시간성이 필요한 자율주행 환경의 경우 1-Stage가 적합하다고 할 수 있다. 그러나 1-Stage 검출기의 경우 2-Stage 검출기에 비해 정확도가 낮을 수 있다는 한계를 갖는다.

따라서 본 연구에서는 자율주행 환경을 고려하여 실시간성을 보장하는 1-Stage 분류기 중 정확성 대비 FPS(Frame Per Second)가 높은 STDN(Scale Transferable Object Detection Network)을 Baseline Network로 선정하였고, 이를 기반으로 Module Study를 진행할 뿐만 아니라 백본망을 최적화하였다[5].

현재 딥러닝 기반의 검출기의 최적화 기법의 경우 크게 5가지 방향으로 연구되고 있다. 첫째, 딥러닝 모델의 계산량 감소를 통한 속도 최적화 방법이다. 이는 Point-wise Convolution, Spatial-wise Separable Convolution과 같이 컨볼루션(Convolution)의 연산량을 감소시킴으로써 속도를 개선하는 방법이다. 대표적으로 Resnet, SqueezeNet 등에서 사용되고 있다[6,7].

둘째, 딥러닝 모델의 파라미터 수를 유지한 채 계산량을 줄이는 방법이다. 이는 Inception, MobileNet 등에서 사용된 방법으로 커널의 크기가 증가함에 따라 기하급수적으로 계산 복잡도가 증가하기 때문에, Depth-wise Separable Convolution 혹은 2-Way Layer 등의 방식이 등장했다. 대표적으로는 Inception v3, RFBNet 등이 있다[8,9].

세 번째는 하드웨어 및 프레임워크 계산 모델에 최적화하는 방법이다. NVIDIA에서 개발한 TensorRT는 NVIDIA GPU의 성능을 극대화할 수 있도록 가속화 알고리즘을 제공한다. 따라서 이러한 가속화 알고리즘의 지원을 받아 연산이 가속될 수 있도록 딥러닝 모델을 최적화하는 방식이다. 대표적인 예로는 PeleeNet이 있다[10].

넷째, 객체검출기의 최적화 방식이다. 많은 백본망들은 클래스 분류에 특화되어 발전되어 왔다. 그러나 그 둘은 목적이 다르다. 클래스 분류는 무엇인지 잘 분류하는 것이 목적이라면, 검출은 잘 찾는 것이 목적이다. 기존의 백본망의 경우 클래스 분류를 잘하기 위해 표본의 종속적인 정보를 제거해야 하였고, 이를 위해 풀링(Pooling)이나 컨볼루션을 사용하였다. 그러나 이러한 작업은 반대로 표본의 중요한 정보를 제거하기도 한다. 따라서 특징맵의 정보를 최대한 남기는 방향으로 딥러닝 모델을 설계할 수 있다. DetNet의 경우는 이러한 방식을 통해 약 2.3%의 mAP 향상을 보였다.[11].

다섯째는 객체검출기의 성능을 올리기 위해 특징맵을 강화하는 방식이다. 기존의 Baseline Network에

작은 딥러닝 모델을 플러그인하는 연구들이다. 크게 Backbone Network 입력 단계에서 사용되는 Stem Block, Attention Mask 등의 백본 모듈 계열과 Detection Layer의 입력 단계에 위치하는 RFBNet, STDN, FSSD 등의 Detection 모듈 계열이 있다. 특히 최근에는 Detection Layer 앞에 UNet의 구조를 적용한 M2Det은 SSD 계열에서 SOTA(State of The Art)의 성능을 보였다[12]. 따라서 본 연구에서는 이러한 모듈들에 대해 Ablation Study를 수행하였고, 추가적으로 모델을 통합하였다. 또한 모델을 변형하여 딥러닝 모델을 최적화하였다. 따라서 SSD 계열에서 SOTA 모델인M2Det에 비해 mAP가 약 1.4% 낮지만 1.9배 빠른 검출기를 개발하였다.

Ⅲ. 객체검출기 최적화

본 장은 객체검출기의 구성도와 객체검출기를 최적화한 방법에 대해 다룬다. 3.1에서는 객체검출기의 전체 구성을 다루고, 데이터의 흐름과 각 모듈의 역할에 대해 설명한다. 3.2장에서는 이를 TensorRT에 대해 소개하고, 계산 모델에 대해 다룬다. 3.3장에서는 VFL(Vertical Fusion Layer)가 적용된 최적화된 우리의 백본망에 대해 다룬다. 3.4장에서는 객체검출기의 속도를 증가시키기 위한 혼합정밀도 학습 및 저 정밀도 추론에 대해 다룬다.

3.1 객체검출기 구성도

그림 1은 객체검출기의 구성도이다. 객체검출기는 크게 3가지로 구성되는데, 첫 번째는 백본망과 Detection Network, Extra Network로 구성된다.

백본망은 Stem Block을 지나 입력으로부터 비용 효율적으로 특징 추출을 수행하는 역할을 하며, 이를 기반으로 Detection Network가 위치 검출과 클래스 분류를 수행한다. 이때 Detection Layer로 들어가기 전에 Extra Layer를 거치는데, Extra Layer에서는 플러그인 모듈을 통해 특징맵을 강화한다. 이때 Baseline Network는 STDN이며, SP는 Super Resolution Module로 본 연구에서는 STDN에서 사용한 Suffle Pixel 알고리즘을 사용하였다. CBR은 Convolution-BachNorm-ReLU 연산이며 AM은 Attention Mask다. 이때 백본망은 DenseNet 을 기반으로 하며, Detection Layer는 SSD를 기반으로 한다. 따라서 DenseNet 기반이기 때문에 C는 Concatenate로 사용하였으며, 이때 발생하는 연산량이 기하급수적으로 증가하는 문제는 각 스테이지 별 한개의 특징맵만을 선택하여 사용하는 것으로 보완하였다.

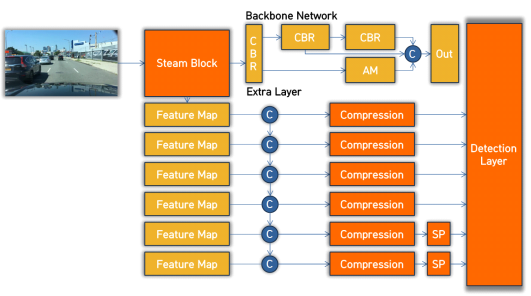

그림 1. 객체검출 구성도
Fig 1. Object Detector Architecture

3.2 Vertical Layer Fusion

TensorRT는 NVIDIA의 딥러닝 프레임워크로 자율주행 플랫폼에 쓰이는 훈련된 신경망을 최적화하고, 추론하는데 주로 사용된다. 이러한 TensorRT는 Caffe, TensorFlow 및 PyTorch와 같은 라이브러리로 작성된 딥러닝 프로그램을 가속화 하는 기능을 한다. 이러한 가속화 기능은 Vertical Layer Fusion이라 하며, 컨볼루션과 Bias ADD, ReLU의 연속된 연산을 하나의 CBR 함수에서 한 번에 처리하여 불필요한 글로벌 메모리 R/W을 줄인다. 그러나 DenseNet의 경우 pre-activation 구조(BN-ReLU-Conv)를 사용하였기 때문에 이러한 Vertical Layer Fusion의 효과를 받을 수 없다. 따라서 3.3에서는 이러한 문제를 해결하기 위해 몇 가지 최적화 기법을 사용하여 딥러닝 모델을 최적화한다.

3.3 딥러닝 모델 최적화 기법

딥러닝 모델은 크게 3가지 관점에서 최적화된다. 첫 번째는 Stem Block을 통한 최적화이다. Stem Block은 Inception에서 제안한 방법으로 입력 크기 줄임으로써 전체 아키텍처를 비용 효율적으로 만드는 역할을 한다. 둘째, PeleeNet에서 영감을 얻은 최적화 기법이다. 이는 컨볼루션의 채널의 크기가 크면 연산량이 기하급수적으로 늘어나는 현상을 막기 위해 채널만 줄인 같은 크기의 커널을 적용한 컨볼루션을 병렬적으로 사용하는 기법이다. 이때 파라미터 수를 유지하기 위해 2개를 중첩해서 사용하는데, 결과적으로 파라미터를 유지하며 빠른 연산 속도를 가져올 수 있다. 세 번째는 Fine Tuning이다. Baseline이 되는 백본망과 SSD의 Fine Tuning 방법에 대하여 설명한다. 자세한 설명은 각 장에서 하도록 한다.

3.3.1 Stem Block

본 연구에서는 PeleeNet에서 사용한 Stem Block나 RFBNet에서 사용된 RFB, RFB-S 모듈을 Stem Block으로 사용했을 때, 성능 저하를 일으키는 것을 발견했다. 따라서 이를 해결하기 위해 STDN의 Stem Block [3x3CBR- 3x3CBR- 3x3CBR- 2x2Pooling-Concatnate]을 기본 구조로 사용하였다. 다만 STDN에서는 Average Pooling을 사용하였지만, 본 연구에서는 Max Pooling을 사용하여도 성능 차이가 나지 않는다는 것을 발견하여, Average Pooling보다 빠른 Max Pooling을 사용하였다. 또한 첫번째 Layer에서 BN-CONV Block을 CBR로 바꿔 사용하였다. 이는 실험적으로 구해진 값이다.

3.3.2 3-Way Dense Layer

그림 2는 3-Way Dense Layer에 관련된 그림이다. 가장 좌측은 Original Dense Layer이며, 이를 PeleeNet에서 사용한 2-Way Dense Layer의 최적화 기법을 적용하였다. 그러나 post-activation 구조가 network가 깊어졌을 때 생기는 성능 저하 문제를 해결하기 위해 각 Layer 별로 Attention Mask를 도입하였다. 이를 통하여 비용 효율적으로 성능을 향상시킬 수 있다. 가장 좌측은 이러한 3-Way Dense Layer가 Vertical Layer Fusion 된 그림이며, CBR은 post-activation 구조를 뜻한다.

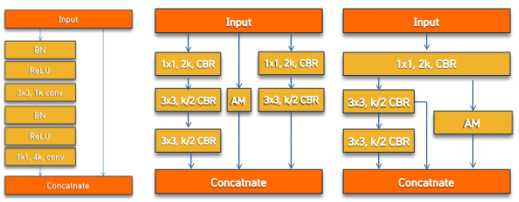

그림 2. 3-Way Dense Layer
Fig. 2. 3-Way Dense Layer

3.3.3 Fine Tuning 방법

Table 1은 백본망의 구조다. 백본망은 총 5개의 스테이지로 구성되며, 구성품들은 크게 3가지로 구성된다. 이때 각 스테이지에서는 하이퍼 파라미터 L_N을 받는데, 이 값은 N 번째 스테이지의 Dense Block의 개수를 의미한다.

Table 2의 G는 Growth Rate를 의미하며, C는 초기 Feature Map의 채널 크기를 의미한다. L_N은 각 N 번째 스테이지의 Dense Block 수를 의미하며, BW는 Bottleneck Width를 뜻한다. 검출기의 경우 SSD의 기본 구조를 가져가며, 각 파라미터는 다음과 같다. 앵커 박스는 전 구간 8개씩 사용하며, 앵커 스케일은 [10,3,2,2,4]를 사용한다. 또한 특징 맵의 크기는 [40,20,10,5,3,1]이며, aspect ratio는 [1.6,2,3]을 기본으로 한다. 이때 min ratio는 15이며, max ratio는 90이다. nms는 soft nms를 사용하였으며, IoU(Intersection Over Union)는 0.75로 필터링 하였다[14]. 이때 score threshold는 0.1로 filtering 하였으며, 각 클래스 별 box는 40개로 고정하였다.

표 1. Backbone Network 구성도
Table 1. Backbone Network Architecture

Stage	Construction	Layer
0	Stem Block	-
1	Dense Block + Transition Layer	L_1
2	Dense Block + Transition Layer	L_2
3	Dense Block + Transition Layer	L_3
4	Dense Block + Pooling	L_4

표 2. 하이퍼 파라미터 셋팅
Table 2. Hyper-Parameter Setting

Stage	G	C	L_N	BW
Our41	32	32	[3,4,8,6]	[1,2,4,4]
Our121	32	64	[6,12,24,16]	[2,3,12,18]
Our161	48	96	[6,12,36,24]	[2,3,18,12]
Our169	32	64	[6,12,32,32]	[3,4,16,16]
Our201	32	64	[6,12,48.32]	[3,4,24,16]

3.3.4 혼합정밀도 학습 및 저 정밀도 추론

기존의 딥러닝을 포함한 많은 분야에서는 HPC(High-Performance Computing)을 요구하였기 때문에 높은 정밀도 연산이 필요하였다. 따라서 Pascal GPU와 CUDA에서는 Single Float(FP32)이나 Double Float(FP64)의 정밀도를 요구하였으나, 최근 연구에서는 Layer Activation Store에 더 낮은 정밀 부동 소수 표현 Half-Float(FP16)을 사용하고 계산에 더 높은 FP32를 사용해도 분류 정확도가 저하되지 않는다는 연구가 보고되었다[7]. 또한 구글 클라우드 TensorRT 개발 가이드 문서에 의하면 ResNet V2기준으로 FP32일 때, 319.1fps를 보였으며, FP16, INT8에서 68%, 79%의 FPS 향상을 보였다. 다만 FP16의 경우 정확도 손실에 대해 큰 영향을 안 미쳤지만, INT8의 경우 명백하게 정확도를 손실을 보였다고 한다. 따라서 본 연구에서는 이러한 사실에 근거하여 정

확도 손실이 크게 나타나지 않지만 높은 속도 향상을 보인 FP16을 기준으로 모델을 구현하였다. 다만 FP16으로 학습할 경우 성능이 저하된다는 보고에 따라 FP32로 학습하고 FP16으로 추론하는 혼합정밀도 학습기법 및 저 정밀도 추론을 사용하였다[15].

Ⅳ. 실 험

본 연구에서는 우리의 딥러닝 모델의 성능 변화를 확인하기 위해 Ablation Study를 진행하였다. 또한 SOTA 모델인 M2Det과 비교 분석을 하였다. 실험은 크게 3가지로 구성된다. 첫째, 딥러닝 모델의 실험이다. 모델 자체의 성능을 확인하기 위해, 효율성, 정확성, FPS 등 다양한 관점에서 성능을 확인하였다. 둘째, 검출기 실험이다. 우리의 객체검출기를 기반으로 자율주행 환경에서 얼마나 잘 동작하는지 정량적으로 실험하였다. 셋째, 실제 검출기의 데모 결과를 확인한다. 본 실험은 Ubuntu 18.04에서 RTX2080TI를 사용하여 Pytorch1.2 TensorRT 5.x, Cudnn 7.6x, Cuda 10.x 환경에서 진행되었다.

4.1 딥러닝 모델 실험

딥러닝 모델의 실험은 최적화의 효과를 보기 위해 학습 방법, 배치 크기, 정규화 기법, 전처리 기법 등을 동일하게 적용하였으며, 초기화는 Xavier 기법을 사용하였다. 학습은 warm up 기법을 사용하였으며, 배치 크기는 32를 사용하였다. 데이터 셋은 클래스 분류용 데이터인 CIFAR 100을 사용하였다.

Table 3은 딥러닝 모델의 효율성을 보기 위한 실험이다. 효율성은 본 연구에서 계산 복잡도 대비 수용체의 표현력으로 평가하고자 한다. 이때 계산 복잡도는 Flops, 수용체는 파라미터의 수로 정하였다. Efficiency는 파라미터 대비 Flops(Floating point Operations Per Second)의 비율로 측정하였다.

결과를 보면 Our41에서 약 0.7이 감소하였으나 더

표 3. 효율성 실험 결과
Table 3. Result of Efficiency Experiment

Model	Flops	Parameters(MB)	Efficiency
Our41	1.72→1.05	1.41→2.17	2.49
Our121	5.96→9.64	7.86→27.1	2.13
Our161	16.1→19.5	28.1→62.1	2.15
Our169	7.04→17.3	12.7→70.5	2.27
Our201	8.97→33.4	18.3→119	1.72

표 4. 정확성 실험 결과
Table 4. Result of Accuracy Experiment

Model	Original	Ours
Our41	64.4	87.4
Our121	83.5	95.6
Our161	86.3	92.3
Our169	86.7	83.6
Our201	89.4	88.6

깊은 모델에서는 오히려 증가한 것을 확인할 수 있다. 이는 깊은 모델일수록 속도 측면에서 비용 효율성이 높다 해석할 수 있다. 또한 동시에 파라미터의 수도 증가하였는데, 이를 표현력 대비 계산량 비율인 효율성 관점에서 보면 적게는 약 1.7배부터 많게는 약 2.5배까지 증가한 것을 확인할 수 있다.

Table 4는 파라미터 증가에 따른 모델의 정확도를 측정한 실험 결과다. 좌측은 기존의 DenseNet의 결과고 우측은 이를 기반으로 최적화한 우리의 딥러닝 모델의 정확도이다. 기존의 pre-activation 구조가 모델이 깊어질수록 효과가 좋다는 연구 결과에 따라 본 연구에서는 속도 저하 대비 정확도 하락을 최소화하기 위하여 Attention Mask를 도입하였다[13]. 실제 Table 4의 데이터를 보면 얕은 모델에서는 성능과 속도가 모두 증가하였지만, 깊은 모델에서는 성능이 감소하는 형태를 보이게 된다. 이러한 정확도 변화는 얕은 모델에서 정확도가 23% 증가하고, 깊은 모델에서는 3%정도 감소하는 것을 확인할 수 있다.

4.2 자율주행 환경에서의 객체검출 정량 평가

자율주행 환경에서의 객체검출 실험은 BDD 데이터 셋을 사용하여 학습하였으며, 각 최적화 기법의 효과를 보기 위하여 Ablation Study를 수행하였다. 또한 주로 사용되는 객체검출기들과 SOTA 모델인 M2Det과의 벤치마킹을 작성하였다. 이러한 벤치마킹을 통해 우리의 검출기의 우수성을 보이고자 한다.

표 5. Ablation Study
Table 5. Ablation Study

Model	mAP:0.75	mAP:IOU	FPS
STDN	15.5	31.5	29
Our169	17.3	36.5	25
Our169+VLF	17.1	36.2	32
Our169+VLF+LP	17.4	36.8	39

Table 5는 각 최적화 기법이 미치는 영향을 확인하기 위해 실험한 Ablation Study 결과표이다. Baseline인 STDN에 비해 IOU 0.75에서 mAP가 약 1.8% 높고 mAP for IOU를 기준으로는 약 5%가 높다. 다만 4FPS 만큼 느려지는데, 이는 VLF(Vertical Layer Fusion)이 적용됨에 따라 빨라지는 것을 확인할 수 있다. 이때 LP(Low Precision)을 사용할 경우 약 39FPS가 나온다. mAP의 경우 VLP 일 때 낮아지고, VFP+LP 일 때 높아지는 경향을 보였는데, 이는 단순히 학습의 오차 범위 내의 값의 변화이기 때문에 성능의 변화로 보기에는 어렵다. 따라서 VLP+LP는 정확성에는 영향을 미치지 않는다 할 수 있다.

표 6은 객체검출 시간의 성능 비교표다. 이때 mAP 0.75는 IOU 0.75에서의 mAP다. mAP for IOU는 IOU를 기준으로 mAP를 측정한 값이다. 우리의 객체검출기의 경우 pre-trained 모델을 사용하지 않았으며, 그 외 나머지 검출기들의 경우 pre-train 모델을 사용하였다. Our41은 가장 기본적인 SSD+VGG 조합에 비해 약 2% 정도 높은 mAP를 가지며, 속도 또한 2FPS 빠르게 측정되었다. 또한 Our169의 경우 SOTA 모델인 M2Det의 mAP가 약 2.8% 높게 측정되었지만 Our169가 약 1.9배 빠른 것으로 측정되었다. 따라서 본 연구에서는 최적화 테크닉을 STDN에 적용하여, SOTA 모델보다 약 1.9배 빠른 모델을 만들었다.

Table 7은 Our169와 M2Det의 클래스별, IOU별로 측정한 표이다. BDD는 크게 10개의 Class를 제공하지만 train의 경우 데이터의 개수가 179개로 학습에 충분한 양을 제공하지 못한다. 따라서 train의 경우 학습이 되지 않아 0%로 나오고 있다.

IOU 별로 mAP를 비교해보면 0.5:0.7에서는 M2Det이 3.7% 높은 성능을 보인다. 다만 IOU 0.75에서는 1.4%, 0.85;0.95에서는 0.3%의 차이를 보인다. 이는 IOU의 제약조건이 높은 환경에서는 성능 차이가 적다 이는 mAP 대비 객체 검출 능력이 강하기 때문이라 해석할 수 있다.

표 6. 객체 검출기 벤치마크
Table 6. Benchmark of Object Detector

Model	Input Size	mAP:0.75	mAP:IOU	FPS
SSD+VGG	300	9.3	13.5	62
SSD+VGG +RFB	300	11.8	15.8	34
FSSD +Mobilenetv2	300	9.3	11.6	32
M2Det	320	20.1	38.4	21
Our41	321	11.3	15.2	64
Our169	321	17.3	36.5	39

표 7. Detection 세부결과
Table 7. Detail Detection Results

Model	AP 0.5:0.7	AP 0.75	AP 0.8	AP 0.85:0.95
bike	23.6/27.7	9.8/10.5	5/14.2	0.7/3.7
bus	50.9/53.8	45.5/47.2	39.8/42.3	17.6/17.4
car	50.1/51.2	36/36.5	30.6/30.8	13.1/13.5
motor	22.3/26.9	9.9/12	4.5/6.7	0.9/1
person	22.6/28	7.6/10.1	3.5/5.2	0.4/0.6
rider	18.8/24.8	6/9.2	2.3/4	0.3/0.3
traffic light	13/16.7	2.3/2.8	0.9/1	0
traffic sign	29.3/35.6	13.7/15.6	8.7/9.5	2.1/2.2
train	0	0	0	0
truck	49.5/52.5	42/43.4	35.6/36.9	13.3/13.8
mAP	28/31.7	17.3/18.7	13.1/15.1	4.8/5.2

4.3 객체검출기 데모 및 정성적 평가

그림 3은 Our169의 객체검출 결과다. 흐린 영상, 맑은 날 영상, 밤 영상을 기준으로 결과를 선정하였다. 사진을 보면 매우 작은 객체에 대해서는 잘 찾지 못하는 현상을 보인다. 그러나 비 오는 환경상 흐린 영상이나, 어두운 밤의 이미지에서는 잘 동작하는 것을 볼 수 있다. 그리고 이러한 환경에서 위치 검출 또한 잘 되는 것을 확인할 수 있다.

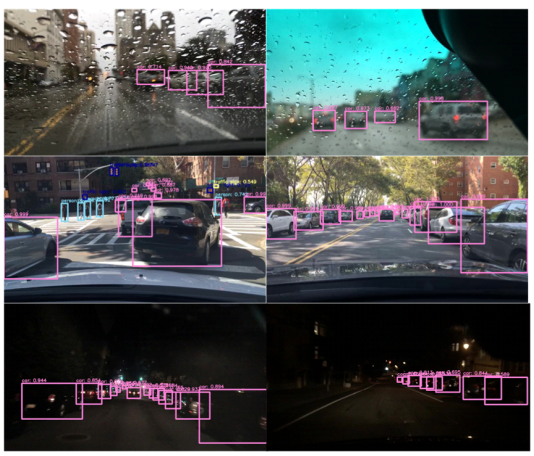

그림 3. Demo of Object Detector
Fig. 3. 객체검출기 데모

V. 결 론

본 논문에서는 딥러닝 기반의 객체검출기가 자율주행 환경에서 실시간으로 동작할 수 있도록 딥러닝 모델의 최적화 연구를 진행하였다. 그러나 실시간 검출기를 만들기 위하여 적용한 post-activation 구조 및 3-way layer 등의 최적화 기법은 성능 저하를 불러왔다. 이러한 문제를 극복하기 위해 Stem Block 계열의 Small Network, Attention Mask 및 Super Resolution 모듈 등을 이용하여 계산 복잡도 대비 풍부한 수용체(Receptive Field)를 가져올 수 있었다. 이를 통해 정확도와 추론 시간을 비용 효율적으로 trade-off 함으로써 자율주행 환경에서의 정확한 위치 검출과 클래스 분류의 성능을 보이는 딥러닝 기반의 객체검출기를 설계하였다. 이를 SOTA 모델인 M2Det 대비 1.4%의 mAP의 차이로, 1.9배 빠른 실시간 객체검출기를 구현하였다. 다만 이는 NVIDIA의 하드웨어 및 프레임워크에 최적화하여 속도 향상을 하는 기법을 사용하였기 때문에 종속성이 높다는 한계를 갖는다. 따라서 향후 연구에서는 종속성이 낮은 딥러닝 모델 자체의 성능을 증가시킬 수 있는 기법을 통하여 보다 정확하고 빠른 객체검출기를 연구 개발해야 할 것이다.

References

[1] M. Markus, J. C. Gerdes, B. Lenz, and H. Winner, et al., *Autonomous driving*, Berlin, Germany: Springer Berlin Heidelberg, 10:978-3, 2016.

[2] S. Ren, et al., "Faster R-CNN: Towards real-time object detection with region proposal networks," *Advances in Neural Inf. Process. Syst.*, pp. 91-99, 2015.

[3] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[4] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. CBerg, "SSD: Single shot multibox detector," in *Eur. Conf. Computer Vision*, pp. 21-37, 2016.

[5] P. Zhou, B. Ni, C. Geng, J. Hu, and Y. Xu, "Scale-transferrable object detection," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 528-537, 2018.

[6] K. He, et al., "Deep residual learning for image recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 770-778, 2016.

[7] F. N. Iandola, et al., "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size," *arXiv preprint arXiv:1602.07360*, 2016.

[8] C. Szegedy, et al., "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 2818-2826, 2016.

[9] S. Liu, D. Huang, and Y. Wang, "Receptive field block net for accurate and fast object detection," *arXiv preprint arXiv:1711.07767*, 2017.

[10] R. J. Wang, X. Li, and C. X. Ling, "Pelee: A real-time object detection system on mobile devices," in *Advances in Neural Inf. Process. Syst.*, pp. 1963-1972, 2018.

[11] Z. Li, et al., "Detnet: A backbone network for object detection," *arXiv preprint arXiv:1804.06215*, 2018.

[12] Q. Zhao, et al., "M2det: A single-shot object detector based on multi-level feature pyramid network," in *Proc. AAAI Conf. Artificial Intell.*, pp. 9259-9266, 2019.

[13] K. He, et al., "Identity mappings in deep residual networks," in *Eur. Conf. Computer Vision*, pp. 630-645, 2016.

[14] N. Bodla, et al., "Soft-NMS--Improving object detection with one line of code," in *Proc. IEEE Int. Conf. Computer Vision*. pp. 5561-5569, 2017.

[15] P. Micikevicius, et al., "Mixed precision training," *arXiv preprint arXiv:1710.03740*, 2017.

김 영 준 (Youngjun Kim)

2018년 2월 : 한국산업기술대학교 컴퓨터 공학과 졸업

2018년 3월~현재 : 성균관대학교 전자전기컴퓨터공학과 석박사통합과정

<관심분야> 컴퓨터비전, 딥러닝, 객체검출

[ORCID:0000-0002-2599-3331]

황 혜 경 (Hyekyoung Hwang)

2018년 8월 : 성균관대학교 수학과 졸업

2018년 9월~현재 : 성균관대학교 전자전기컴퓨터공학과 석박사통합과정

<관심분야> 컴퓨터비전, 딥러닝, 객체검출

[ORCID:0000-0002-8291-6957]

신 지 태 (Jitae Shin)

1986년 2월 : 서울대학교 전기공학과 졸업

1988년 12월 : University of Southern California 전자공학 석사

2001년 5월 : University of Southern California 전자 공학 박사

<관심분야> 컴퓨터비전, 딥러닝, 객체검출

[ORCID:0000-0002-2599-3331]