Smart Media Journal / Vol.8, No.2 / ISSN:2287-1322
http://dx.doi.org/10.30693/SMJ.2019.8.2.66

사물인식을 위한 딥러닝 모델 선정 플랫폼

(Deep Learning Model Selection Platform for Object Detection)

이한솔*, 김영관*, 홍지만**

(Hansol Lee, Younggwan Kim, Jiman Hong)

요 약

최근 컴퓨터 비전을 활용한 사물인식 기술이 센서 기반 사물인식 기술을 대체할 기술로 주목을 받고 있다. 센서 기반 사물인식 기술은 일반적으로 고가의 센서를 필요로 하기 때문에 기술이 상용화되기 어렵다는 문제가 있었다. 반면 컴퓨터 비전을 활용한 사물인식 기술은 고가의 센서 대신 비교적 저렴한 카메라를 사용할 수 있다. 동시에 CNN이 발전하면서 실시간 사물인식이 가능해진 이후 IoT, 자율주행자동차 등 타 분야에 활발하게 도입되고 있다. 그러나 사물 인식 모델을 상황에 알맞게 선택하고 학습시키기 위해서는 딥러닝에 대한 전문적인 지식을 요구하기 때문에 비전문가가 사물 인식 모델을 사용하기에는 어려움이 따른다. 따라서 본 논문에서는 딥러닝 기반 사물인식 모델들의 구조와 성능을 분석하고, 사용자가 원하는 조건의 최적의 딥러닝 기반 사물 인식 모델을 스스로 선정할 수 있는 플랫폼을 제안한다. 또한 통계에 기반한 사물 인식 모델 선정이 필요한 이유를 실험을 통해 증명한다.

■ 중심어 : 딥러닝 ; 신경망 ; 사물인식 ; 플랫폼

Abstract

Recently, object recognition technology using computer vision has attracted attention as a technology to replace sensor-based object recognition technology. It is often difficult to commercialize sensor-based object recognition technology because such approach requires an expensive sensor. On the other hand, object recognition technology using computer vision may replace sensors with inexpensive cameras. Moreover, Real-time recognition is viable due to the growth of CNN, which is actively introduced into other fields such as IoT and autonomous vehicles. Because object recognition model applications demand expert knowledge on deep learning to select and learn the model, such method, however, is challenging for non-experts to use it. Therefore, in this paper, we analyze the structure of deep - learning - based object recognition models, and propose a platform that can automatically select a deep - running object recognition model based on a user 's desired condition. We also present the reason we need to select statistics-based object recognition model through conducted experiments on different models.

■ keywords : Deep Learning ; Neural Network ; Object Recognition ; Platform

Ⅰ. 서 론

최근 자율주행 자동차의 실시간 사물 인식을 위한 방안으로 이미지처리가 주목받고 있다. 기존의 센서 기반 사물인식 기술은 이미 상용 자율주행 자동차에 도입되어 사용되고 있을 정도의 뛰어난 인식률을 보이고 있지만, 고가의 센서를 사용하고 있어 대량생산이 어렵다는 문제가 있다. 반면 이미지처리를 활용하여 사물인식을 수행할 경우 비교적 저렴한 카메라를 사용할 수 있으므로 단가를 크게 낮출 수 있다[1].

2012년 Imagenet Challenge인 ILSVRC에서 GPU연산을 활용한 AlexNet[2]이 기존의 SVM 기반 모델들을 제치고 우승한 이후 CNN 기반의 모델들이 Image Classification 분야에서 주류를 이루게 되었다. 이후 Image Classification을 비롯하여 상위 분야인 사물 인식 분야에서도 CNN으로 구성된 모델들[3-7]이 주류를 이루었다. 이후 Deep CNN의 문제점을 해결한 연구들[8-9]이 등장하면서 CNN에 기반을 둔 딥러닝 사물인식 모델들의 성능이 더욱 향상되었고 고성능 시스템에서의 실시간 동작이 가능해졌다[4].

이전에는 저성능 임베디드/모바일 환경에서 딥러닝 기반 사

* 비회원, 숭실대학교 컴퓨터학과
** 종신회원, 숭실대학교 컴퓨터학부
이 논문은 2017년도 정부(과학기술정보통신부)의 재원으로 한국연구재단-차세대정보컴퓨팅기술개발사업의 지원을 받아 수행된 연구임(No. NRF-2017M3C4A7069432)..

접수일자 : 2019년 03월 26일
수정일자 : 2019년 04월 08일
게재확정일 : 2019년 04월 08일
교신저자 : 홍지만 e-mail : jiman@ssu.ac.kr

물인식 모델들을 직접 구동시키는 경우 매우 느리거나 정확도가 낮았으나, 현재는 모바일 환경을 위한 Image Classification 모델[10-11]들과 Object Detection Model[7]의 연구가 진행되고 있고, Tensorflow Lite 등의 모바일 전용 프레임워크 또한 개발되고 있다. 이들은 공통적으로 고성능 시스템에서 학습을 수행하고, 임베디드/모바일 환경에서는 학습 없이 추론만을 수행한다[12].

이러한 딥러닝 사물인식 모델의 학습을 위해서는 엄청난 연산량이 필요하기 때문에 고사양의 시스템이 요구된다. 따라서 고사양 원격 시스템 대여 서비스인 Amazon AWS, Microsoft Azure 등의 클라우드 컴퓨팅 플랫폼들이 상용 시스템에서 딥러닝 모델 학습을 수행하기 위해 자주 사용되며, AWS와 Azure 자체적으로 딥러닝을 위한 여러 기능을 지원하기도 한다.

하지만 직접 딥러닝 기반 사물인식 모델을 개발하지 않더라도, 모델을 사용하기 위해서는 적절한 모델을 선택하고 하이퍼파라미터를 결정하는 등 딥러닝에 대한 전문적인 지식이 요구되는 작업이 필요하다. 이는 딥러닝 모델을 사용하고자 하는 타 분야의 전문가들에게 큰 문제가 될 수 있다. 따라서 별도의 관련 지식 없이 기존에 개발된 딥러닝 기반 사물인식 모델들을 사용할 수 있는 플랫폼이 필요하다.

따라서 본 논문에서는 딥러닝 기반 사물인식 플랫폼 설계를 위해 최근 좋은 성능을 보이는 YOLO 모델들의 구조를 분석하고 자동 모델 선택을 지원하는 플랫폼을 제안한다.

본 논문에서 제안하는 플랫폼은 딥러닝 기반 사물인식 모델들을 미리 포함하고 있으며, 사용자가 데이터셋 학습을 요청할 경우 각 모델들의 하이퍼 파라미터별, 데이터셋별 실험결과를 기반으로 최적의 모델을 스스로 판단 및 선택하여 학습을 진행한다.

Ⅱ. 관련연구

Junwei Han 외[13]는 사물 인식 분야를 사물의 위치를 바운딩박스로 추정하는 Object Detection(OD), 사물의 정확한 위치를 추정하는 Salient Object Detection(SOD), 그리고 추정한 사물의 종류를 추정하는 Category-Specific Object Detection (COD)의 세가지 분야로 세분화 하여 각 분야에 대한 연구 동향을 표 1과 같이 정리하였다.

표 1. 사물인식 연구동향

분야	접근방법	설명
OD	Region-merging Approaches	유사한 픽셀끼리 결합을 반복하여 사물단위로 픽셀을 분리
	Window-selecting Approaches	미리 생성해둔 윈도우에서 Objectness 점수를 계산하여 사물의 위치를 찾아냄
	Box-regressing Approaches	Regression 함수로 학습을 진행하여 바운딩박스를 생성
SOD	Bottom-up SOD	배경으로부터 전경의 사물들을 분리하여 위치를 추정
	Top-down SOD	이미지에서 오브젝트를 직접 찾아냄
COD	Object Proposal-based Approaches	Region Proposal을 이용하여 사물 인식 및 분류
	Regression-based Approaches	기존 COD를 Regression 문제로 변환하여 기존보다 단순하게 연산

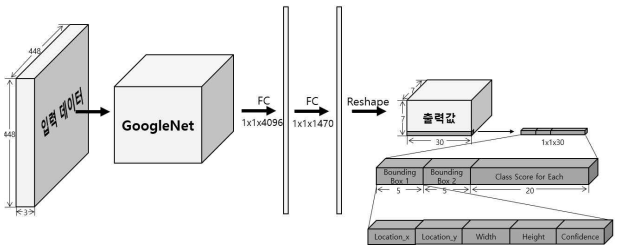


그림 1. YOLOv1 모델 구조

본 절에서는 Regression-based Approaches에 해당하는 YOLO 모델[7]을 분석한다. YOLO는 Object Proposal-based Approaches 모델인 Faster R-CNN[7]의 동작속도 개선을 위해 Region Proposal Network와 Classifier

를 하나의 네트워크로 구성한 모델로, 현재 실시간 사물인식 모델 중 가장 대표적인 모델이다.

그림 1은 초기 YOLO 모델인 YOLOv1의 구조를 도식화한 것이다. YOLOv1은 Fully Connected Layer가 사용되어 후속 모델에 비해 연산량이 많으며, 입력 데이터와 출력값의 크기가 FC에 종속된다는 문제가 있다.

이 문제는 YOLOv2[6]에서 개선되어 동작속도가 대폭 빨라졌다. 또한 입력 데이터의 크기를 조정할 수 있게 되어 정확도와 동작 속도 간 Trade-off가 가능하게 되었다. 입력 데이터가 큰 경우 필요한 연산량이 많아져 동작속도는 느려지지만 정확도는 올라가고, 반대로 작은 경우 필요한 연산량이 적어져 동작속도가 빨라지지만 이미지는 손실되어 정확도가 떨어진다. 그림 2는 416x416의 입력 데이터를 갖는 YOLOv2의 모델 구조를 도식화한 예시이다.

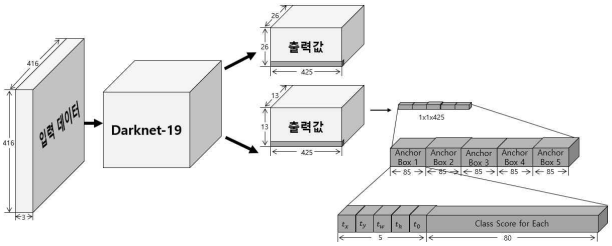


그림 2. YOLOv2 모델 구조

YOLOv2는 정확도 향상을 위해 FC를 제거한 것 이외에도 두 개의 출력값에 대해 사물의 위치 추론 및 분류를 진행하거나 Batch Norm을 적용하는 등 다양한 개선을 하였으며, 독자적으로 개발한 19개의 레이어로 구성된 DarkNet-19을 백본 네트워크로 활용한다. 그 결과 YOLOv1보다 정확도와 동작속도 모두 더 좋은 결과를 보이게 된다. 가장 최근의 YOLO 버전인 YOLOv3[5]의 경우, 53개의 레이어를 사용하고 3개의 출력값에 대한 추론 및 분류로 YOLOv2와 비교하여 동작속도는 느리지만 더욱 정확한 결과를 보인다. 그림3은 YOLOv3 모델을 도식화한 예시이다.

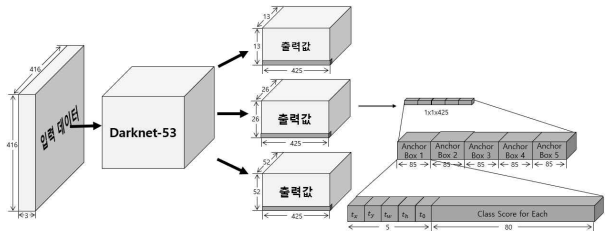


그림 3. YOLOv3 모델 구조

Tiny YOLO는 저성능 기기용 사물인식 모델이다. YOLO 연구팀이 실험한 결과에 따르면[5] YOLOv3의 동작속도는 가장 높은 입력 해상도 608x608에서 20FPS대 수준을 보여주기 때문에 실시간 사물인식 모델로 볼 수 있다. 그러나 이 수준의 동작속도를 내기 위해서는 실시간 사물인식 기능이 도입되어야 할 자율주행 자동차, 드론 또는 CCTV등의 기기에 탑재하기 힘든 수준의 하드웨어가 요구된다. Tiny YOLO는 기존 YOLO 모델의 구조적인 특징은 유지하면서 네트워크의 전체적인 사이즈를 줄인 모델로 YOLO 모델이 점차 발전하면서 Fast YOLO, Tiny YOLOv2, Tiny YOLOv3 모델이 개발되었다.

YOLO 모델은 YOLOv2부터 해상도에 따른 동작속도와 정확도간 trade-off를 지원하기 때문에 별도의 모델 변경 없이 저성능 기기에서는 동작속도를 높이는 선택을 할 수 있고, 고성능 기기에서는 정확도를 올리는 선택을 할 수 있다. 하지만 Tiny YOLO의 존재는 trade-off로는 모델간의 간격을 메꾸기 힘들다는 것을 반증한다.

III. 플랫폼 설계

1. 플랫폼 구조

그림 4. 플랫폼 전체 구조

본 논문에서 제안하는 모델 자율 선정 플랫폼은 마스터노드와 슬레이브노드로 구성된다. 마스터노드는 데이터관리기, 노드관리기, 통신관리기로 이루어지며 모델을 선정하고 슬레이브노드들에게 동작을 지시할 수 있다. 슬레이브노드는 작업관리기, 통신관리기로 이루어지며 실제로 사물인식 모델을 실행시키는 노드이다.

데이터 관리기는 학습 모델과 모델 학습을 위한 데이터, 그리고 학습 종료 후의 가중치를 관리하기 위한 모듈이다. 본 플랫

폼에서는 같은 모델이지만 하이퍼 파라미터가 다른 경우 서로 다른 모델로 취급한다. 주요 기능은 표 2와 같다.

표 2. 사물인식 모델 실험 환경

함수명	기능	파라미터	리턴
Write_Label	바운딩박스의 위치정보와 클래스넘버에 서로 맞게 라벨링 데이터를 생성	Class Number, Coordinate Information	
Upload_Dataset	슬레이브노드에 데이터셋 업로드	Path	Dataset Identifier
Download_Dataset	슬레이브노드로부터 데이터셋 다운로드	Path, Dataset Identifier	
Send_Dataset	데이터셋을 특정 노드에 전송	Node Identifier, Dataset Identifier	result key
Set_Model	DB에서 모델 정보를 가져와 최적의 모델을 찾는 기능	Minimum FPS, Minimum mAP	Model
Get_Model	학습이 완료된 모델을 다운로드	Model Identifier, Path	

노드관리기는 플랫폼 전체의 노드 및 작업들을 관리하는 모듈이다. 플랫폼은 슬레이브노드들을 관리하며 하나의 슬레이브 노드에 최대 하나의 작업을 할당한다. 할당된 작업이 없는 슬레이브노드들은 작업 대기 큐에서 대기하며, 작업 할당시 작업 대기 큐에서 조건에 맞는 노드를 선택하여 작업을 할당한다. 표 3은 노드 관리기의 주요 함수 목록이다.

표 3. 사물인식 모델 실험 환경

함수명	기능	파라미터	리턴
Forward_Task	작업을 노드에 전달. result key를 통해 작업 결과를 알 수 있음	Job	result key
Node_Schedule	작업을 할당할 노드를 선택	Model	Node Identifier

통신관리기는 플랫폼 내 마스터노드와 슬레이브노드간 통신을 담당하는 모듈이며 양측 모두에 존재한다. 처음 연결을 시도할 때 양측 노드 통신관리기의 리스트가 서로 일치하지 않으면 연결이 되지 않는다. 표 4는 통신관리기 모듈이 사용하는 Tcp 패킷 클래스의 주요 함수 목록이다.

표 4. 사물인식 모델 실험 환경

함수명	기능	파라미터	리턴
Tcp_Write	Tcp 패킷 작성	Packet Identifier, String ...	
Tcp_Read	Tcp 패킷 해독	String	
DB_Query	DB에 쿼리 전달 및 결과 확인	String	String

작업관리기는 슬레이브노드에서 실제로 모델 학습이 수행되는 모듈로, 마스터노드로부터 실행 명령어를 전달받아 실행한 후 마스터노드가 요청하는 경우 표준출력으로 실행 결과를 전달한다. 전달된 값은 진행상황에 대한 정보로써 마스터노드가 확인할 수 있다.

Ⅳ. 구현 및 평가

1. 모델 선정

가. 실험 환경

본 절에서는 실험을 통해 사물인식 모델들의 성능을 비교하고 모델 자동 선정의 필요성을 확인한다. 상세 실험 환경은 표 5와 같고, 데이터셋으로는 COCO2014 데이터셋과 VOC2007++2012 데이터셋을 활용한다.

표 5. 사물인식 모델 실험 환경

구분	항목	이름
하드웨어	프로세서	Intel Xeon E5-2620v4 * 2 (8 core, 16 Thread, 2.1GHz each)
	메모리	64GB
	그래픽카드	Geforce GTX1080 8GB
소프트웨어	운영체제	CentOS 7 64bit
	드라이버	Nvidia Driver 390.87
	OpenCV	3.3.0
	Cuda	9.1
	Cudnn	7.0.5
	Python	3.6

사물인식 성능을 테스트하기 위해서는 정확한 측정 여부를 판단할 수 있는 기준이 필요하다. 일반적으로는 물체가 실제로 존재하는 바운딩박스(Ground Truth)와 모델에서 예측한 바운딩박스간 IoU(0~1 사이의 값)을 측정하여 일정 값 이상인 경우 정확하게 측정한 것으로 판단한다. 본 실험에서는 IoU값 0.5를 기준으로 YOLO 모델들의 성능을 비교한다. 또한 YOLO 모델에서 지원하는 trade-off의 성능을 확인하기 위해 해상도는 32의 홀수 배수로 설정하였다.

나. COCO2014 실험

COCO2014 데이터 셋에는 총 80개의 클래스와 이미지당 평균 5~7개의 물체가 존재한다[14]. 그림 5는 COCO2014 데이터 셋에 대한 실험 결과를 그래프로 나타낸 것이다.

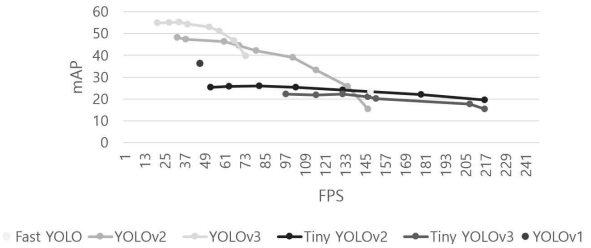


그림 5. 모델간 성능 비교 (COCO2014, IoU=0.5)

YOLO모델은 해상도와 동작속도간 trade-off를 지원하지만 뚜렷한 한계점을 가지고 있다. 본 실험 환경에서 동작속도를 30FPS 이상 기대할 때에는 YOLOv2(608)모델이 48.10%, YOLOv3(416)모델이 54.32%의 정확도를 보이므로 YOLOv3의 정확도가 더 높지만, 동작속도가 70FPS 이상일 때에는 YOLOv2(416)모델이 44.45%, YOLOv3(160)모델이 39.57%로 YOLOv2가 더 높은 정확도를 보인다. 이런 현상은 YOLOv2와 Tiny YOLO모델에도 그대로 나타나며, 145FPS 이상의 동작속도를 얻기 위해서는 trade-off를 적용하여 YOLOv2모델을 동작시키는 것보다 Tiny YOLOv2를 동작시키는 것이 더 좋은 결과를 보여준다.

또한, YOLO 모델을 소형화한 Tiny YOLO의 경우 Tiny YOLOv3보다 Tiny YOLOv2가 모든 결과에서 높은 정확도를 보이는 것을 확인할 수 있다. 따라서 이전의 사물 인식 모델의 개선점이 이후의 다른 사물인식 모델에 적용되지 않을 수 있다는 것을 알 수 있다.

다. VOC2007++2012 실험

VOC 데이터셋은 COCO 데이터셋보다 분류할 클래스의 종류와 사물이 더 적기 때문에 같은 모델을 사용하였을 경우 동작속도가 더 빠르다. 또한 COCO 데이터셋보다 높은 정확도를 가지며 이미지 내 사물들의 크기가 COCO 데이터셋보다 균일하다[14]. 각 사물인식 모델의 특성은 COCO와 동일하게 나타난다. 그림 6는 trade-off를 적용한 YOLO 모델들을 비교한 그래프이다.

VOC 데이터셋 또한 trade-off 적용 이후 일부 결과에서 구

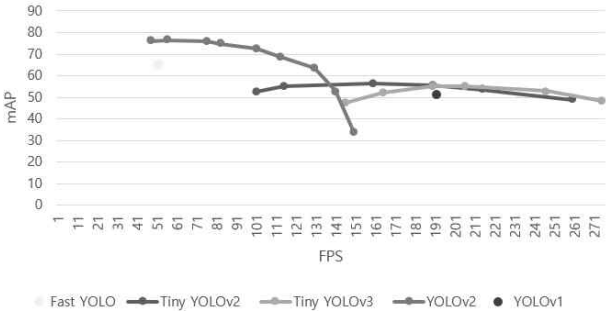


그림 6. 모델간 성능 비교 (VOC2007++2012, IoU=0.5)

형 모델이 신형 모델보다 정확도가 더 높은 모습을 보이고 있다. 따라서 원하는 동작속도나 정확도에서 사용자가 적절한 모델을 정확하게 선택하기 어려우며, 동작속도와 정확도를 고려한 적절한 모델 선정 전략이 필요하다.

2. 구현

제안하는 플랫폼의 구현은 YOLO 모델에 대한 학습을 지원할 수 있도록 진행되었다. 본 장에서는 본 플랫폼에서 데이터를 학습하는 과정을 통해 구현 결과를 확인한다. 표 6은 구현 환경 명세이다.

표 6. 플랫폼 구현 환경

구분	항목	이름
마스터노드	운영체제	Windows 10 Home
	개발언어	Java
	프레임워크	Jedis 2.9.0, Netty 4.1.25
DB서버	운영체제	Windows 10 Home
	DB	Redis 3.0.6
슬레이브노드	운영체제	CentOS 7 64bit
	개발언어	Java
	프레임워크	Jedis 2.9.0, Netty 4.1.25

본 장에서는 학습 과정 및 결과가 타당함을 보이고자 학습결과가 공개되어 있는 YOLO모델과 COCO2014 데이터셋을 사용하였다. 본 구현에서는 추후 두 개 이상의 마스터노드를 동시에 실행하기 위하여 각 마스터노드를 세션으로 구분한다. 마스터노드를 실행하면 DB에서 세션 넘버를 할당받은 후 DB에 세션의 IP정보를 등록한다. 슬레이브노드의 경우 실행 시 DB에서 각 세션이 현재 담당하고 있는 슬레이브노드의 개수를 받고, 담당하는 슬레이브노드의 개수가 가장 적은 세션의 IP를 받아 해당 세션에 연결을 시도한다. 그림 7은 마스터노드와 슬레이브노드간의 연결 과정이다.

그림 7. 노드간 연결 과정

YOLO 모델의 실험결과는 DB에 미리 저장된다. 실험결과는 Redis DB에서 지원하는 Sorted Set 데이터 타입을 사용하여 관리한다. Sorted Set은 하나의 키 값에 값-점수로 구성된 쌍을 정렬된 상태로 여러 개 저장할 수 있는 데이터 타입이다. 본 플랫폼에서는 정확도와 속도에 대한 데이터 타입을 각각 따로 정의하여 실험결과를 저장한다. 그림 8은 Redis 클라이언트 프로그램을 사용하여 DB에 저장된 mAP와 fps의 리스트를 각각 확인한 것이다.

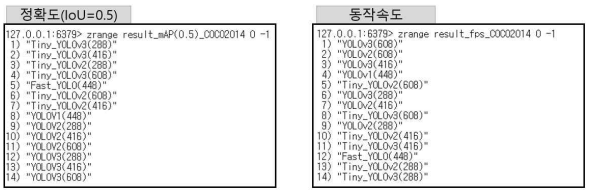


그림 8. DB 내 리스트

모델을 학습시키기 위해서는 데이터셋의 준비가 필요하다. 실제로 학습을 진행하기 위해서는 방대한 양의 데이터와 데이터 라벨링 작업이 필요하므로 본 절에서는 이러한 데이터가 미리 준비된 VOC 데이터셋을 사용한다. 모델 학습을 위해 준비해야 할 파일의 목록은 [표 7]과 같다.

표 7. YOLO 모델 학습에 필요한 파일 목록

구분	이름
dataset.names	데이터셋에서 구분할 클래스의 이름들을 각 줄에 하나씩 기록한 파일
dataset.cfg	데이터셋에 대한 정보를 저장하는 파일로, 구분할 클래스의 개수, dataset.names 파일의 경로, trainlist.txt 파일의 경로 등을 기록
trainlist.txt	학습에 사용할 모든 파일들의 경로를 각 줄에 기록한 파일
*.jpg	학습에 사용할 이미지
*.txt	학습에 사용할 각 이미지에 대한 라벨링 파일

학습할 모델이 정해지면, 슬레이브노드는 전송받은 데이터셋을 사용하여 학습을 진행한다. 모델의 학습 결과는 출력 리다이렉션을 통해 마스터노드로 전달된다. 그림 9는 작업 실행 및 결과 전달 과정을 나타낸 것이다.

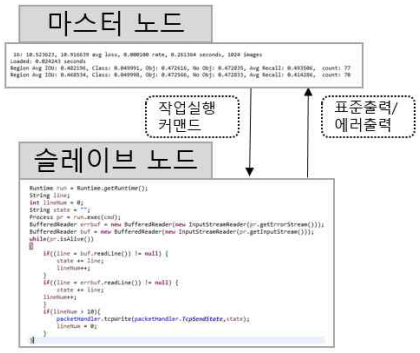


그림 9. 작업 실행 및 결과 전달

V. 결론

본 논문은 사물인식 모델들에 대한 분석 및 실험을 통해 사용자가 원하는 성능의 기준에 따라 적절한 모델을 사용해야 함을 밝혔고, 이를 바탕으로 딥러닝 기반 사물인식 모델을 훈련 및 수행하는 과정 일부를 자동화한 플랫폼을 제안하였다. 사물인식 모델들을 실행하기 위해서는 직접 최적의 모델을 선택하고 하이퍼 파라미터를 설정해야 하지만, 본 논문에서 제안하는 플랫폼은 통계를 바탕으로 각 상황에 적합한 최적의 모델과 하이퍼 파라미터를 플랫폼에서 자동으로 선정한다.

본 논문에서 제안하는 플랫폼은 크게 마스터노드, 슬레이브노드 그리고 DB로 구성된다. DB는 플랫폼의 공유저장소로 활용된다. 마스터노드는 데이터 관리기, 노드 관리기, 그리고 통신 관리기의 4가지 모듈로 구성된다. 데이터 관리기는 학습에 필요한 데이터셋과 사물인식 모델을 관리하고, 노드 관리기는 슬레이브노드들을 관리하며 통신 관리기는 노드 간 통신을 담당한다. 슬레이브노드는 작업 관리기와 통신 관리기로 구분되며, 통신 관리기는 마스터노드의 통신 관리기와 연결되어 패킷의 송수신을 담당하고, 작업 관리기는 실제로 모델을 실행하고 실행한 결과를 저장하는 역할을 한다.

그러나 모델을 학습시키기 위해서는 딥러닝에 대한 전문지식이 요구되기 때문에, 아직 본 플랫폼이 완벽하게 자동화된 학습을 지원한다고 보기 어렵다. 더욱 자동화된 플랫폼을 개발하기 위해서는 모델 학습에 필요한 파라미터를 설정하기 위한 연구를 진행해야 할 것이다.

REFERENCES

[1] 김재상, 문해민, 반성범, "오픈소스 하드웨어 기반 차선검출 기술에 대한 연구," *스마트미디어저널*, 제6권, 제3호, 15-20쪽, 2017년 9월

[2] Krizhevsky Alex., Sutskever Ilya., and Hinton E. Geoffrey.. "ImageNet Classification with Deep Convolutional Neural Networks." A*dvances in Neural Information Processing Systems (NIPS). IEEE*. 2012.

[3] Girshick Ross. "Fast R-CNN." *The IEEE International Conference on Computer Vision (ICCV). IEEE*. 2015.

[4] Ren Shaoqing, He Kaiming,, Girshick Ross, and sun Jian. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." *Advances in Neural Information Processing Systems 28 (NIPS 2015). IEEE*. 2015.

[5] Redmon Joseph., and Farhadi Ali.. "YOLOv3: An Incremental Improvement." arXiv Preprint. arXiv:1804.02767. 2017.

[6] Redmon Joseph., and Farhadi Ali. "YOLO9000: Better, Faster, Stronger." *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE*. 2017.

[7] Redmon. Joseph, Divvala Santosh., Girschick Ross., Farhadi Ali. "You Only Look Once: Unified, Real-Time Object Detection," *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE*. 2016.

[8] Ioffe Sergey. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.," *Proceedings of the 32nd International Conference on Machine Learning*, 2015

[9] He Kaiming., Zhang Xiangyu., Ren Shaoqing., and Sun Jian. "Deep Residual Learning for Image Recognition,." *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE*. 2016.

[10] Sandler Mark, Howard Andrew, Zhu Menglong, Zhmoginov Andrey, and Chen Liang-Chieh, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE*. 2018.

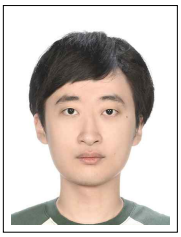
[11] Ma Ningning, Xiangyu Zhang, Zheng Hai-Tao, and Sun Jian . "ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design," *The European Conference on Computer Vision (ECCV)*. Springer. 2018.

[12] 김진수, 김민구, 반성범, "임베디드 모듈 기반 지능형 영상감시 시스템의 최적화에 관한 연구," *스마트미디어저널*, 제7권, 제2호, 40-46쪽, 2018년 6월

[13] Han, Junwei, Zhang Dingwen, Cheng Gong, Liu Nian, and Xu Dong, "Advanced Deep-Learning Techniques for Salient and Category-Specific Object Detection," *The IEEE Signal Processing Magazine. IEEE*. 2018.

[14] Gauen Kent, Dailey Ryan, Laiman John, Zi Yuxiang, Asokan Nirmal, Lu Yung-Hsiang, Thiruvathukal K. George, Shyu Mei-Ling, and Chen Shu-Ching, "Comparison of Visual Datasets for Machine Learning," *The IEEE Conference on Information Reuse and Integration (IRI). IEEE*. 2017.

저 자 소 개



이한솔

2017년 숭실대학교 컴퓨터학부 학사 졸업.
2019년 숭실대학교 컴퓨터학과 석사 졸업

<주관심분야 : 시스템 소프트웨어, 운영체제>

김영관

2019년 숭실대학교 컴퓨터학부 학사 졸업.
2019년~현재 숭실대학교 컴퓨터학과 석사 재학

<주관심분야 : 시스템 소프트웨어, 운영체제>

홍지만(종신회원)

2003년 서울대학교 컴퓨터공학과 박사 졸업
2004년~2007년 광운대학교 컴퓨터공학과 교수.
2007년~현재 숭실대학교 컴퓨터학부 교수

<주관심분야 : 시스템 소프트웨어, 운영체제>