캡스톤디자인 최종결과물 제출 리스트

구분	제출 자료명	제출 여부	제출형태	비고
1	최종보고서 첨부 자료 1) 프로젝트 제안서 2) 요구사항분석서 3) 설계서 4) 주간보고서		팀 화일	팀장 명의로 제출
2	창의설계경진대회 발표 (동영상 등) 자료		팀 화일	최종보고서에 포함
3	SW 프로그램 등록증		팀 pdf 화일	프로그램등록절차와 등록증 샘플 첨부 참조 (수수료 영수증 및 통장 사본)
4	Git hub 등록실적		팀 화일	프로그램등록절차 첨부
5	Git hub 등록 source code		팀 화일	source code를 하나의 압축 파일로 생성
6	논문실적		팀, 혹은 개별	최종 보고서에 첨부
7	특허실적		팀, 혹은 개별	최종 보고서에 첨부
8	외부경진대회		팀, 혹은 개별	최종 보고서에 첨부
9	기타 결과물		팀, 혹은 개별	최종 보고서에 첨부
10	동료평가		개별 화일	블랙보드에 개별 제출
11	TOPCIT 응시		개별 화일	재출치 않아도 됨

1. 최종보고서는 조별로 팀장 명의로 블랙보드에 1부만 제출할 것.
   (팀장이 불가시 팀원명의도 가능하나, 사전에 담당교수에게 사유를 설명하고 제출할 것)
2. "최종결과물 제출 리스트"는 문서로 출력하여 제출하고, 팀장과 팀원의 서명을 할 것.
3. 결과물 1번부터 9번까지 내용과 실적을 최종보고서에 반드시 포함할 것.
4. 제출 여부는 O. X로 구분할 것.
5. 제반 실적은 증명할 수 있는 원본 문서나, 사본이 가능하며 pdf형태로 첨부하고, 접수는 하였으나 아직 등록이 되지 않는 것은 신청서를 첨부할 것.

6. 동료평가는 개별로 작성하고 각 개인의 명의로 블랙보드에 제출할 것.(별도 서류 작성)
7. TOPCIT 응시표도 pdf로 스캔해서 개인별로 제출, 기타 개인 실적은 개별 제출

2020. 06. 26.

캡스톤 2반 2조 팀장 : 이승규 (서명)

팀원 :

2020 학년도 1 학기

캡스톤 디자인 결과보고서

소 속 (전공)	컴퓨터공학과 2반	
팀명(주제)	미래창조(Accident Detection System based on CCTV Images)	
지 도 교 수	문현준교수 (인) 권기학교수 (인)	
팀 장	학번: 15011031	이름: 이승규
팀 원	학번: 14010990	이름: 김정혁
	학번: 13011051	이름: 김준한
	학번: 15011044	이름: 송창석
	학번: 16011033	이름: 위진
제 출 일 자	2020. 06. 26	

세종대학교 컴퓨터공학과

최종보고서

1. 개발 목표

CCTV영상을 딥러닝 기술로 분석하여 사고상황을 자동으로 식별한다. 사고차량 수, 위치, 움직임, 화재발생 여부 등 사고 상황에 대한 구체적인 데이터를 제공하여 관련기관에서 신속한 사후처리를 가능하게 한다.

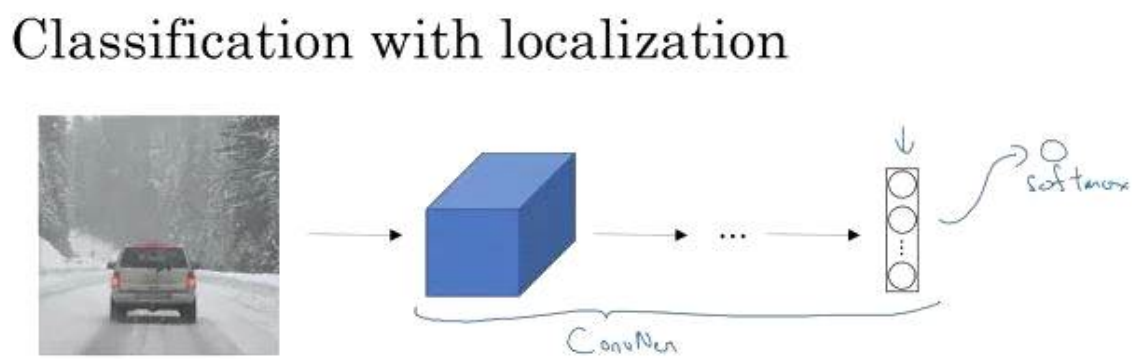
2. 설계 사양서

2.1 제안프로젝트 계획서

2.1.1 이론적 배경

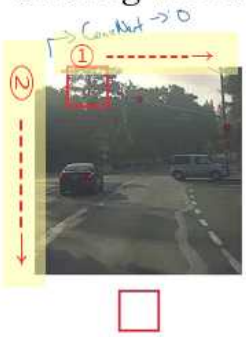
▶Object detection, Pose estimation

Object detection은 Image Classification과 Image localization을 동시에 처리하는 기술이다. Image Classification을 통해 어떤 종류의 이미지인지 분류하고 Image localization을 통해 이미지의 위치를 잡아낸다.

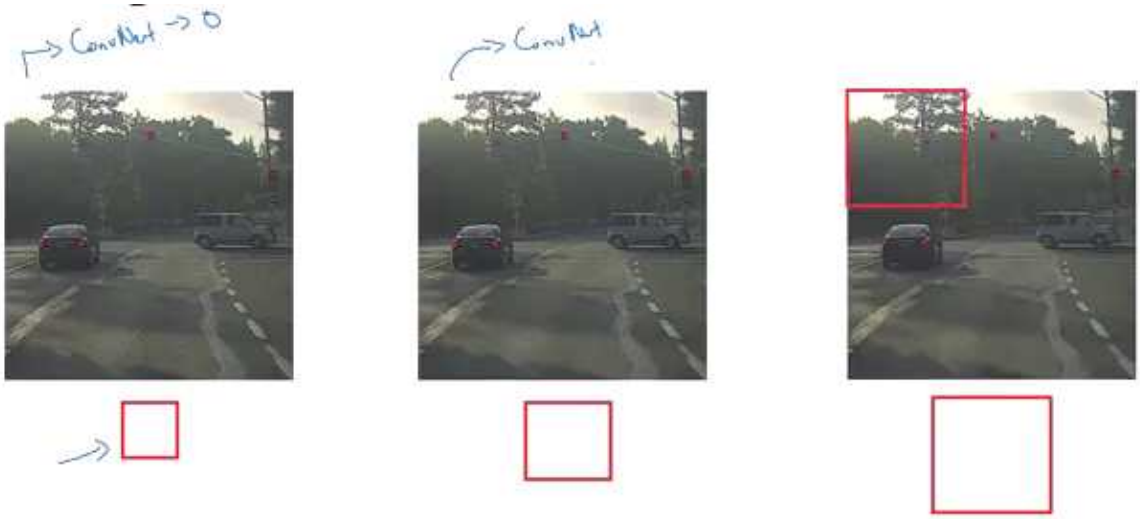


두 기술을 동시에 사용하기 위해 output layer를 조정한다. 일반적인 classification할 대상의 수와 물체를 감지할 bounding box의 top-left좌표, height, width등의 4가지 노드를 추가한다. 예를 들어 모델이 pedestrian, car, motorcycle, background를 감지하고 bounding box를 그린다면 output layer의 노드는 8개가 된다.

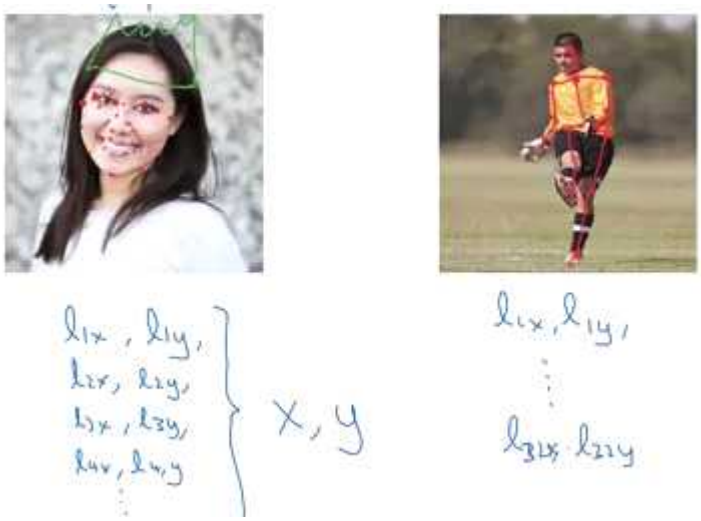
Sliding windows detection



위의 모델로는 하나의 물체밖에 알아낼 수 없다는 단점이 있다. 여러 개의 물체를 감지하기 위해서 이미지를 여러 개로 나누어 모델에 주입한다. Sliding windows detection의 경우 window size와 stride를 결정하고 window size만큼의 이미지를 왼쪽 위부터 stride만큼 움직이며 모델에 주입하여 대상 물체를 감지한다. 정한 window size로 모든 이미지를 검사하였다면 window size를 키우면서 다시 반복하여 물체를 검사한다. 이 알고리즘을 통해 여러 물체의 detection이 가능하다.

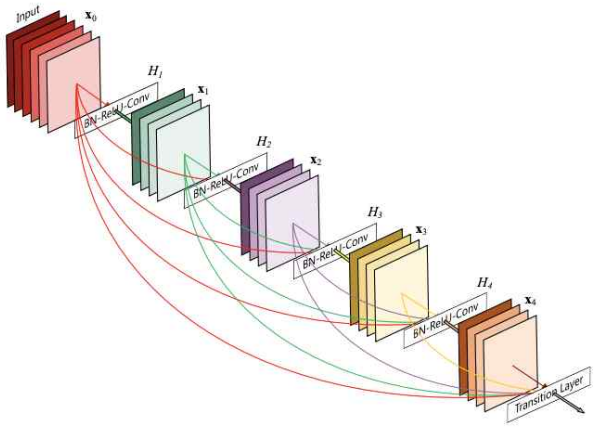


Pose estimation의 경우 landmark detection 기술을 이용하여 일반적인 object detection의 output layer 노드에 pose에 대한 정보를 삽입하는 형태로 진행된다. 예를 들어 posenet의 경우 사람의 양쪽 눈, 코, 양쪽 귀, 양쪽 어깨, 팔꿈치, 허리, 무릎 등을 감지하는데 감지하고 싶은 정보의 수만큼 object detection의 output layer의 노드를 추가한다. 여러 물체나 사람의 pose를 감지하는 방식 또한 object detction의 방식과 유사하며 output layer의 수가 상대적으로 많은 만큼 연산량 또한 증가한다.

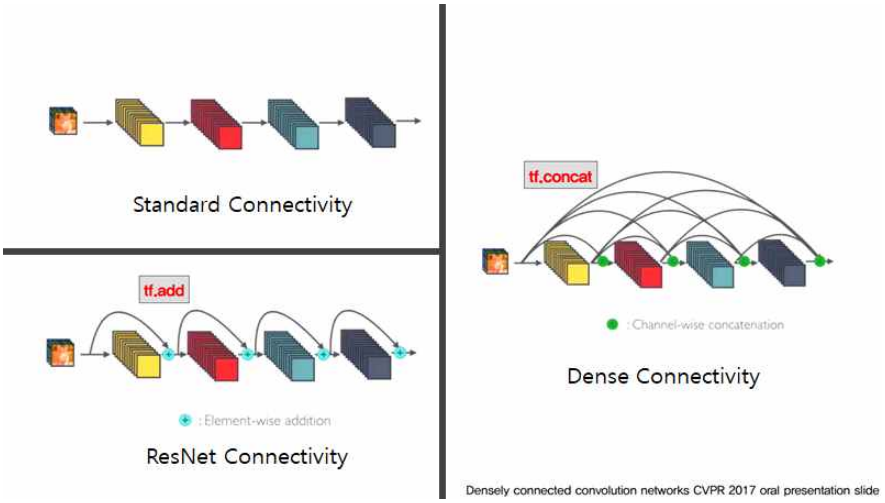


▶ **학습에 활용된 각 네트워크 설명**

- DenseNet201



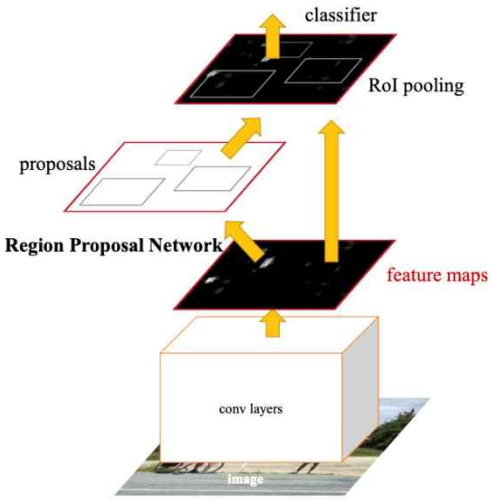
차량 화재 및 파손 탐지를 위한 모델에서는 **DenseNet201** 을 사용하였다. 이 네트워크는 (3 x H x W)의 이미지를 입력으로 받게 되는데 정확히 말하면 가로, 세로 픽셀 수가 각각 224이고, 3채널인 이미지를 입력으로 받는다. DenseNet(Dense Convolutional Network)은 피드 포워드 방식으로 각 계층을 **다른 모든 계층에 연결**한다. 이를 통해 각 계층에 대해 모든 이전 계층의 피쳐 맵(feature map)이 입력으로 사용되며, 이 피쳐 맵이 다시 모든 후속 계층에 대한 입력으로 사용되는 구조이다. 이러한 구조로 인해 DenseNet은 Vanishing Gradient 문제를 완화하고 각 레이어의 재사용률이 높으며, 하이퍼 파라미터의 수를 크게 줄일 수 있어서 결과적으로 우리가 가진 상용 GPU에서도 연산량의 부하를 조금이나마 줄일 수 있었다.



DenseNet의 또 다른 장점으로는 단순히 ResNet처럼 정보를 add하는 것이 아니라 **concatenate** 한다는 것이다. 정보를 더하지 않고 쌓는 것이므로 과거의 정보가 쉽게 wash out 되지 않고 전체적인 정보의 흐름을 유지하는 데에 유리하다. 또한, dense connection 방식은 정규화(Normalization)에도 유리하므로 우리가 가진 비교적 적은 데이터 셋에서도 과적합을 줄여주는 효과가 있었다.

그리고 우리는 이러한 베이스 모델 뒤에, 추가적으로 Fully Connected를 위해 1920 크기의 flatten 레이어와 1개의 output layer를 배치하였다. 또한 0과 1의 이진 분류이므로 당연히 binary crossentropy의 loss function을 적용하였다.

- Faster RCNN



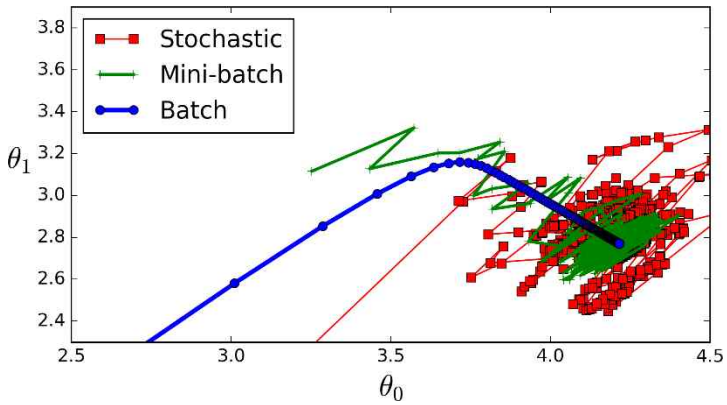
일반 객체 탐지를 위한 모델에서는 Faster RCNN 을 활용하였다. 해당 네트워크의핵심 아이디어는 **Region Proposal Network**(RPN) 이다. 기존 selective search 방식을 제거하고 RPN

을 통해서 Region of Interest(RoI)를 계산할 수 있게 되었다. 이를 통해 GPU를 통한 RoI 계산이 가능해졌으므로 정확도가 확연히 올라가게 되었다. 해당 네트워크는 객체 탐지 분야에서 매우 보편화된 것이라 큰 고민없이 사용하였다. 다만 Faster RCNN의 단점은 SSD에 비해서 학습 속도는 빠르지만 predicton이 느리다는 것이다. 현재 우리의 프로젝트에서 이 네트워크에 종속되는 모델이 꽤 많은데다가, 정확성 보다는 속도 면에서 성능상 이슈가 있으므로 차후에는 SSD 또는 SSD MobileNet 사용을 고려해볼 수 있을 것이다.

▶전체적인 하이퍼 패러미터 값 설정

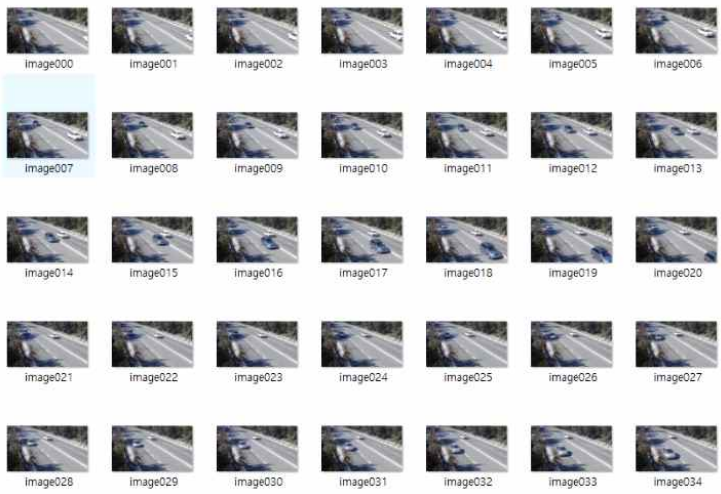
아래의 하이퍼 패러미터들의 초기 값은 일반적으로 권장되는 값들로 시작하였으나, 학습을 지속하면서 반복적으로 튜닝한 결과, 현재까지는 해당 세팅이 가장 유리하다는 결과를 얻게 되었다.

- **base model trainable : true**
- **learning rate : 0.0001**
- **optimizer : Adam**
- **activation function : ReLU**
- **batch size : 32**
- **epochs : 7**



batch size를 128로 설정하였을 때, 메모리의 크기가 부족하여 ran out of memory 이슈가 발생하였는데, 가용 자원의 한계를 고려하여 32로 조정하였다. 더 좋은 성능의 컴퓨터에서 작업한다면, batch size를 늘려서 그라디언트 추정의 정확도를 더 높일 수 있다.

▶데이터 셋의 구성



우선 train : validation의 비율은 **7 : 3**으로 조정하였는데, 이는 전통적인 방법론에서 지향하는 방식이다. 데이터 셋의 크기가 클수록 validation에 적은 비율만 할당하더라도 꽤나 많은 양이 제공되므로 상관이 없지만, 우리는 데이터 셋의 크기가 작으므로 3 정도는 투자할 필요가 있다고 판단하였다.

positive와 negative 데이터 셋의 비율에 대해서는 여러 이견이 있으나, 우리는 **1 : 1** 의 비율로 수집하였다. 참고하였던 대부분의 논문에서도 동등한 비율을 권장하였다.

데이터 셋의 개수는 약 **3천 장**의 이미지 셋으로 구성되어 있다. 차량 파손이나 화재 관련 데이터는 당연히 우리가 직접 만들 수는 없고, kaggle 같은 곳에서 범용적으로 제공되는 양질의 데이터 셋을 활용하였다. 여기에 추가적으로 몇몇 영상에서 클리핑하거나, 별도로 구한 이미지를 약간 추가하였다.

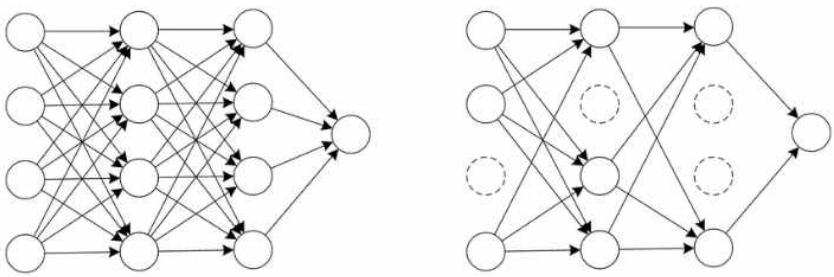
다행히도 탐지에 관해서는 0과 1의 이진 분류이므로 별도의 영역을 그리는 라벨링은 필요가 없었다. 단지 디렉토리를 구분하여 이미지를 넣어주는 작업을 수행해 주었다.

▶과적합 이슈 해결

- Horizontal Flip

학습 시에 train accuracy는 적은 epoch으로도 지나치게 높게 나오고, 반면에 validation accuracy는 낮게 나오는 현상이 있었는데 이를 통해 과적합이 발생하였음을 유추할 수 있었다. 우리는 과적합 이슈를 해결하기 위하여 일단 전체 데이터 셋에 horizontal flip을 적용하였다. 이는 data augmentation 방식의 일종인데, 장점은 크게 두가지가 있다. 하나는 데이터 셋을 두 배 늘려서 부족한 데이터 셋을 보충할 수 있다는 것이다. 또 하나는 데이터 양이 늘어남으로써 적은 데이터의 특정 패턴이나 노이즈에 국한하지 않고, 일반적 패턴을 학습하게 되므로 과적합을 방지하는 데에 유리하다.

- Dropout



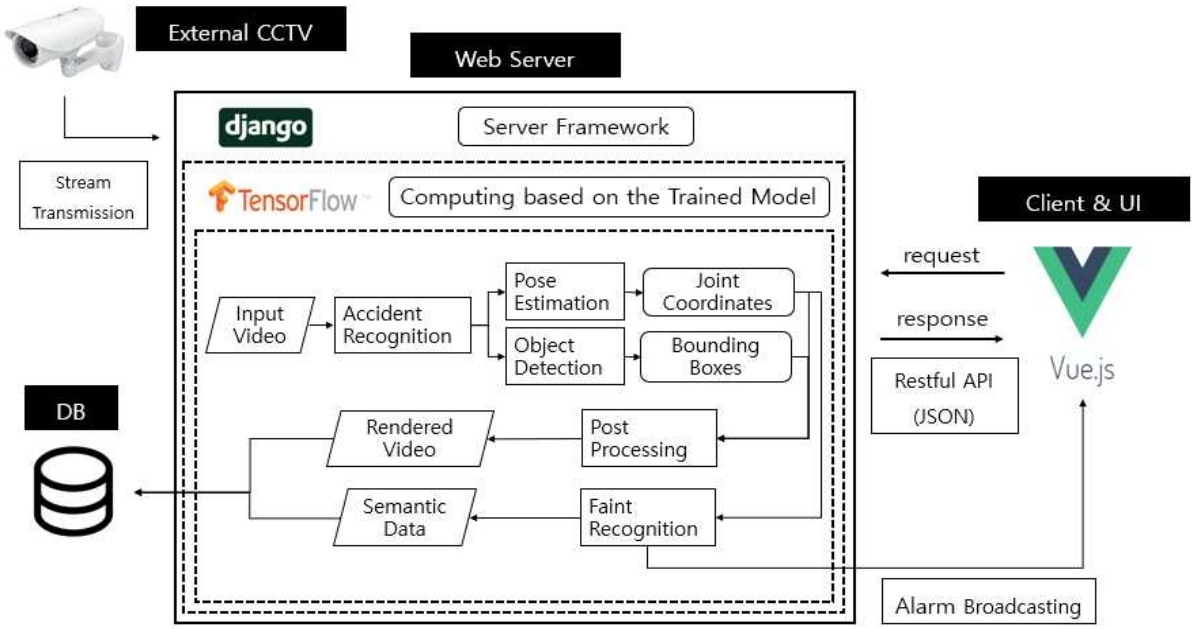
(a) Standard Neural Network (b) Network after Dropout

drop-out은 학습 과정에서 신경망의 일부를 사용하지 않는 방식이다. 우리는 dropout의 비율을 0.2로 조정하였는데 이 말은 학습 과정마다 랜덤으로 20% 비율의 뉴런을 사용하지 않고 절반의 뉴런만을 사용한다는 것이다. 이 방식을 통해 학습 시에 특정 뉴런이나 조합에 의존적이게 되는 것을 방지할 수 있으며, 결과적으로 과적합을 방지하는 데에 도움을 준다.

- Epoch 수 조정

가장 일차원적인 방식은 epoch 수를 조정하는 방식이다. 이는 수동으로 조정하는 방법도 있겠지만 early stopping을 적용하여 더 간단하게 처리하였다. 즉, 이전 epoch에 비해 몇번의 step 이상 오차가 증가하였다면 학습을 중단하는 것이다. 다만 우리는 적은 epoch으로도 빠른 목표치에 도달하였기 때문에 patience는 낮게 적용하였다.

▶ 전체적인 시스템 아키텍쳐



▶ 개발 환경

개발 환경

-Language

Python
Javascript

-Code version control

Github

-Library

Tensorflow (인공지능)
Opencv (이미지 처리)
Django (웹서버)

-Tool

Visual Studio Code

2.2 요구사항 분석서

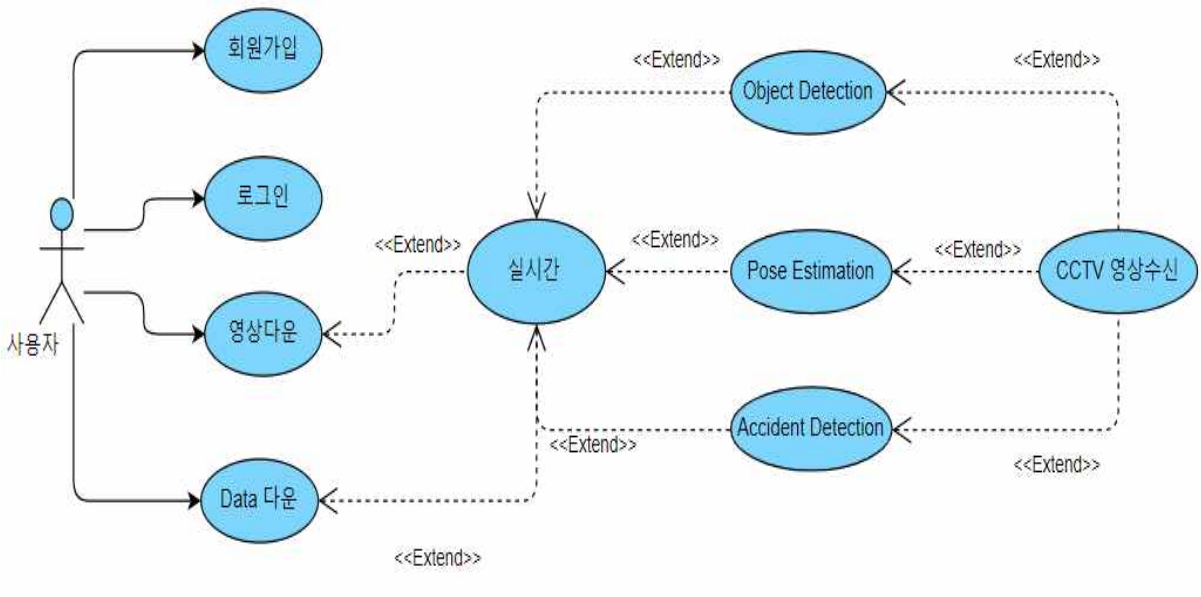
- 기능적 요구사항1

요구사항	ID
회원가입	UR001
중복회원방지	UR002
로그인	UR003
실시간 영상	UR004
사고 감지	MR001
알림 기능	MR002
Object Detection	MR003
Pose Estimation	MR004
영상/데이터 다운로드	UR005

- 기능적 요구사항2

구분	요구사항	상 세 내 용	중요도	난이도
UR001	회원가입	회원가입	하	하
UR002	중복회원방지	중복된 ID의 회원가입을 방지한다.	하	허
UR003	로그인	로그인/아웃 기능을 구현한다.	하	하
UR004	실시간 영상	CCTV 영상을 웹을 통해 실시간으로 볼 수 있게 한다.	중	중
MR001	사고 감지	CCTV로부터 받은 영상이 사고상황인지 아닌지 판단한다.	상	상
MR002	알림 기능	사고가 발생한다면 로그인이 된 이용자에게 알림 전송한다.	중	중
MR003	Object Detection	영상 내에서 사람과 차량에 대한 Object Detection을 수행한다.	상	상
MR004	Pose Estimation	사람의 관절부를 찾아 좌표 값을 구하고 영상에 추정된 골격을 표시한다.	상	상
UR005	영상/데이터 다운로드	사고가 발생했을 때의 영상이나object Detection, pose estimation 등으로 얻은 정보들을 다운 받을 수 있게 한다.	하	하

2.3 유스케이스 다이어그램

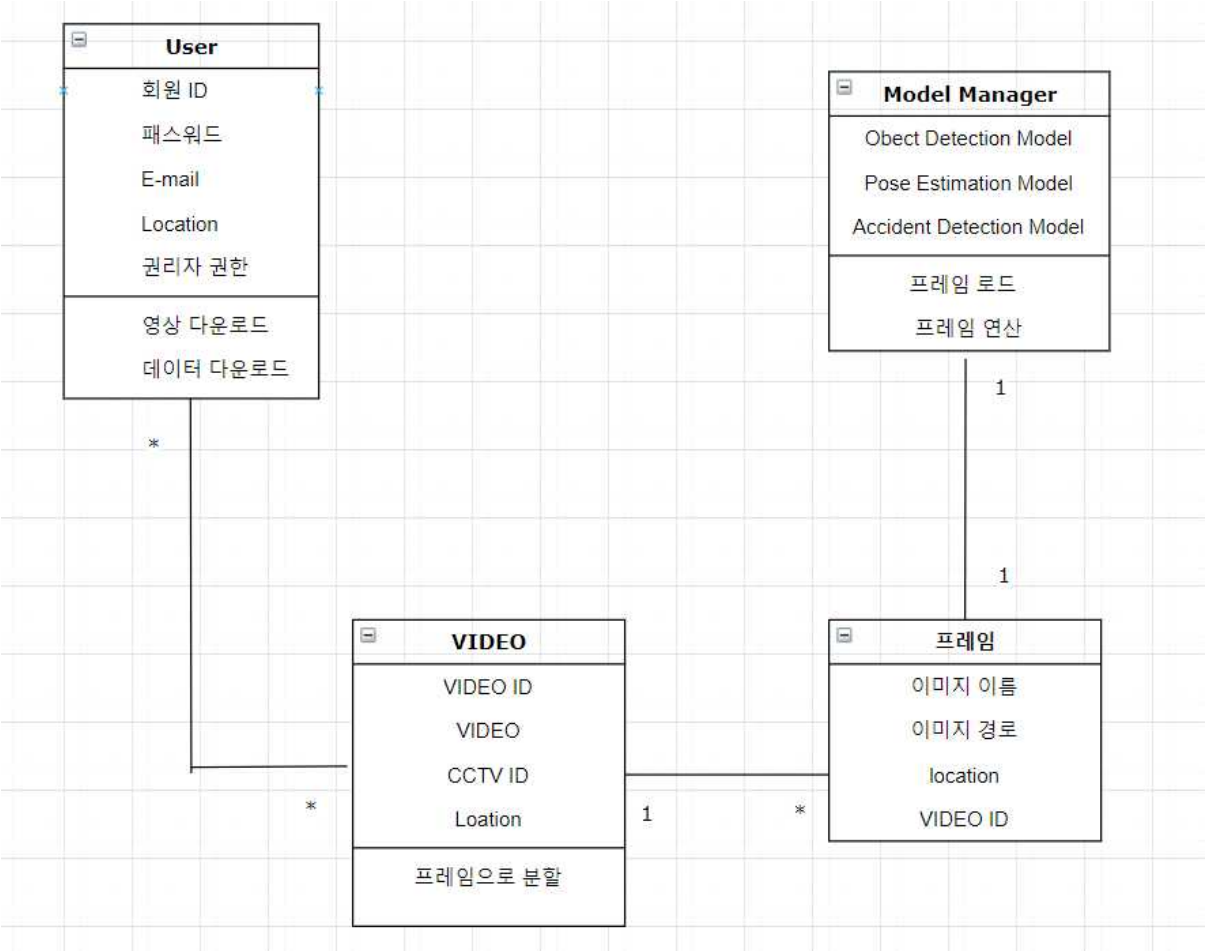


요약 : 사용자는 회원가입, 로그인, 딥 러닝 모델이 처리한 영상과 Data를 다운 받을 수 있다.

회원가입과 로그인은 파일 등의 방식으로 사용자의 정보를 저장하여 관리한다. 다운로드는 딥 러닝 모델로부터 받은 출력 값을 이용하게 되는데, 여기서 딥 러닝 모델은 CCTV로부터 영상을 받으면서 실시간으로 동작하며, Object Detection으로 자동차와 사람을 감지하고, Pose Estimation을 통해 사람의 자세를 추정하며, Accident Detection을 통해서 사고의 유무를 판정한다.

이 과정으로부터 자동차의 위치나 사람의 위치, 자세, 사고의 유무 등의 데이터가 나오고 이 데이터를 masking 한 영상이 나오는데 사용자는 이 영상과 Data를 다운 받게 된다.

2.4 클래스 다이어그램



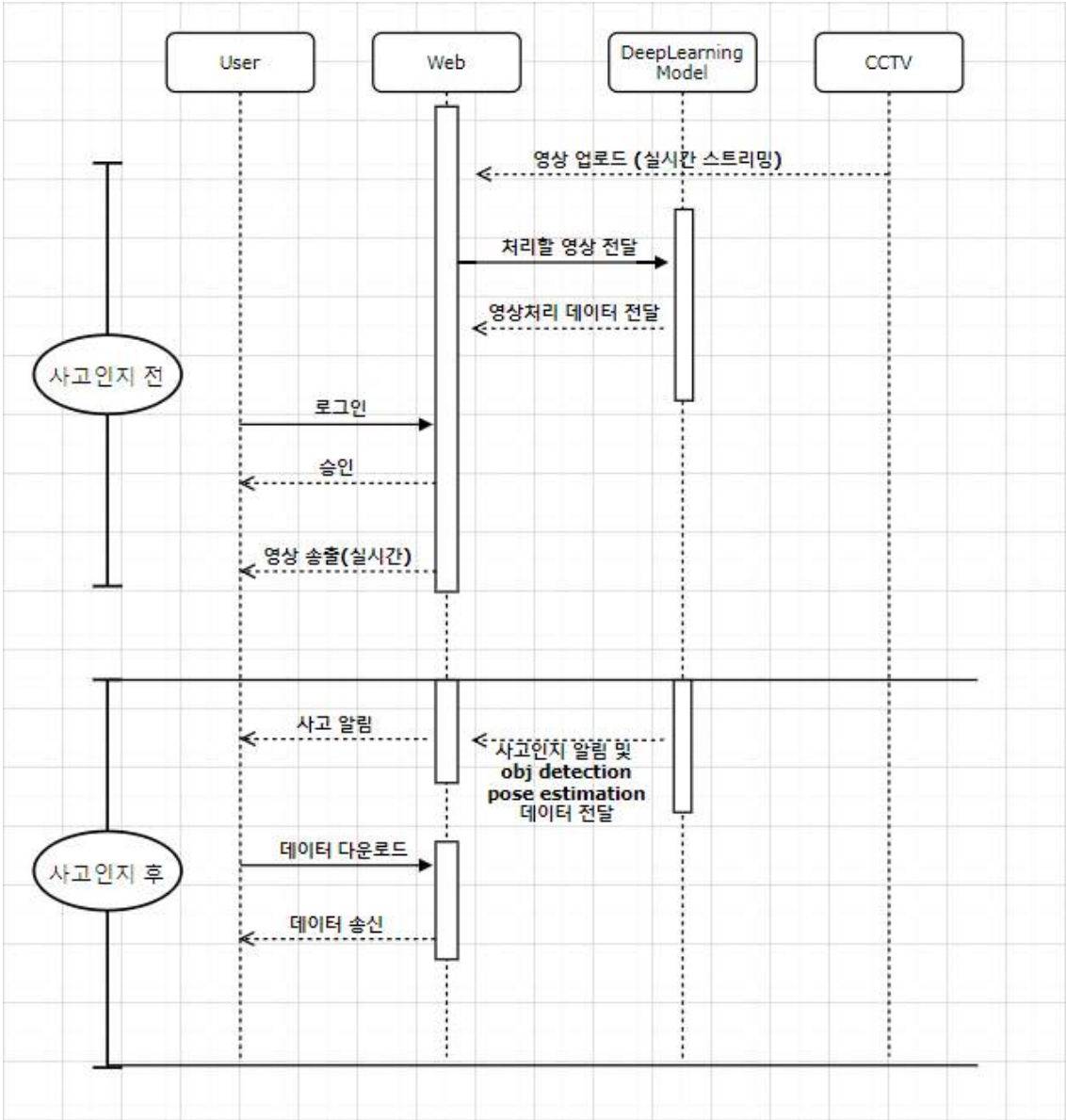
User : 회원의 정보와 회원이 사용 가능한 함수를 갖는다.

Video : CCTV 영상의 정보, 위치, masking 된 영상의 정보와 프레임으로 분할하는 함수를 갖는다.

프레임 : 프레임 관련 정보를 갖는다.

Model Manager : 모델의 정보와 프레임 로드, 연산 함수를 갖는다.

2.5 시퀀스 다이어그램



사고 인지와 사용자의 로그인 전에도 CCTV는 웹에 실시간으로 촬영한 영상을 업로드 한다. 웹은 CCTV로부터 받은 영상을 딥 러닝 모델에 보내며, 딥 러닝 모델은 영상을 처리하여 masking 된 영상을 보낸다. 사용자가 웹으로 로그인을 시도하고 성공하면, 웹은 사용자에게 딥 러닝 모델로부터 받은 영상을 실시간으로 보여준다. 딥 러닝 모델이 사고를 인지하면, 딥 러닝 모델은 Object Detection, Pose Estimation 관련 데이터와 알림을 웹에게 보내고, 웹은 사용자에게 알림을 보낸다. 사용자가 해당 사고 관련 데이터를 다운받고자 하면, 웹은 masking 된 영상과 데이터를 사용자에게 제공한다.

3. 테스트 명세서

▶단위 시험 계획서

단위시험 ID	단위시험 명	작업 권한	시험항목	시험 결과
UT_01	사람 및 차량 인식	전체	해당 동영상에 대하여 사람 및 차량이 정확히 인식되는지 확인	인식이 정확하게 이루어짐
UT_02	자세 추정	전체	해당 동영상에 대하여 사람의 자세가 정확히 추정되는지 확인	추정이 정확하게 이루어짐
UT_03	차량 파손 탐지	전체	해당 동영상에 대하여 차량의 파손 상태를 정확히 식별하는지 확인	식별이 정확하게 이루어짐
UT_04	차량 화재 탐지	전체	해당 동영상에 대하여 차량의 화재를 정확히 식별하는지 확인	식별이 정확하게 이루어짐
UT_05	영상 전송	전체	영상이 정상적으로 전송되는지 확인	정상적으로 전송됨

▶통합 시험 계획서

통합시험 ID	통합시험 명	입력자료	시험항목	시험 결과
IT_01	영상 재생	입력 동영상	해당 동영상이 출력되는지 확인	영상 재생 동작
IT_02	데이터 렌더링	입력 동영상	데이터를 영상 위에 렌더링	렌더링 정상 작동
IT_03	데이터 다운로드	가상버튼 클릭 여부	csv 및 영상 데이터의 다운로드	성공적으로 다운로드 됨
IT_04	알람 발생	없음	사고 상황 시 알람 발생 여부	알람 정상적으로 발생

▶시스템 시험 계획서

시스템 시험 ID	시스템 시험 명	시스템 시험 유형	시험 절차	만족 기준	시험 결과
ST_01	차량 파손 탐지	성능	영상이 재생될때 차량 파손을 탐지하는 시간을 측정함	초당 10프레임의 영상 처리	초당 3프레임
ST_02	차량 화재 탐지	성능	영상이 재생될때 차량 화재를 탐지하는 시간을 측정함	초당 10프레임의 영상 처리	초당 6프레임

ST_03	차량 파손 탐지	신뢰성	영상이 재생될때 차량 파손 탐지의 정확도를 측정함	정확도 90% 이상	92.8%
ST_04	차량 화재 탐지	신뢰성	영상이 재생될때 차량 화재 탐지의 정확도를 측정함	정확도 90% 이상	92.9%

▶ **성능**

	FPS	Accuracy
Car Crash Detection	3.10	92.8%
Car Fire Detection	6.53	92.9%

테스트 명세에 보다 자세하게 적겠지만, 현재로서는 위와 같은 결과가 도출되었다. 정확성은 두 모델 모두 92% 대로 상당히 높은 결과를 얻었으며, FPS의 경우에는 3~6 정도로 꽤나 낮은 성능이지만, 우리가 연산에 사용한 GPU가 일반 사용자 용의 GTX 1060 라는 점을 감안한다면 훨씬 빠른 연산 환경이 제공되는 실서버 환경에서는 전혀 문제가 없을 것으로 판단된다.

4. 요구사항 대비 시스템 구현 내용

요구사항	품질요소	완성도	설명
사고를 인지하는데 초당 10프레임을 처리해야한다. 화재를 감지하는데 초당 10프레임을 처리해야한다. 다운로드 버튼을 눌렀을 때 다운로드가 3초 내에 완료되어야 한다.	성능	90%	컴퓨터 성능에 따라 차이를 보임
사고 인식률이 80%(accuracy) 이상 나와야 한다. 재현률(recall)이 0.8 이상 나와야 한다.	신뢰성	90%	
웹를 사용하여 사용자 편의성을 증대시켜야 한다.	가용성	100%	

5. 개발 추진 내역

▶ 전체 일정

4/7 시작 6/14 종료(10주)	담당	1주차	2주차	3주차	4주차	5주차	6주차	7주차	8주차	9주차	10주차
데이터 확보	공통	데이터 확보									
데이터 전처리	공통		데이터 전처리								
화재 감지	김준한			화재 감지							
사람/차량 인식	송창석			사람/차량 인식							
사고 인식	이승규			사고 인식							
서버 구현	김정혁			서버구현							
UI 구현	위진			UI 구현							
최종 테스트	공통									최종 테스트	

▶개별 일정

- 김정혁

	1주차	2주차	3주차	4주차	5주차	6주차	7주차	8주차	9주차	10주차
데이터 수집										
Django 프레임워크 환경 세팅 및 UI 디자인 구상										
Bootstrap을 이용하여 웹 페이지 구현										
Deeplearning-model 결과이미지 영상화										
사고 인지 알람 기능										
주제 수정으로 인한 프로트 UI 수정 및 버그 해결										
작업 내역 병합 및 테스팅										

- 김준한

	1주차	2주차	3주차	4주차	5주차	6주차	7주차	8주차	9주차	10주차
사전 학습 및 정보 수집										
데이터 수집 및 가공										
개발 환경 세팅 및 아키텍처 구조 구상										
차량 화재 탐지 모듈 (모델 학습, 로직 추가)										
렌더러 프로세스 세팅 및 이슈 해결										
작업 내역 병합 및 테스팅										

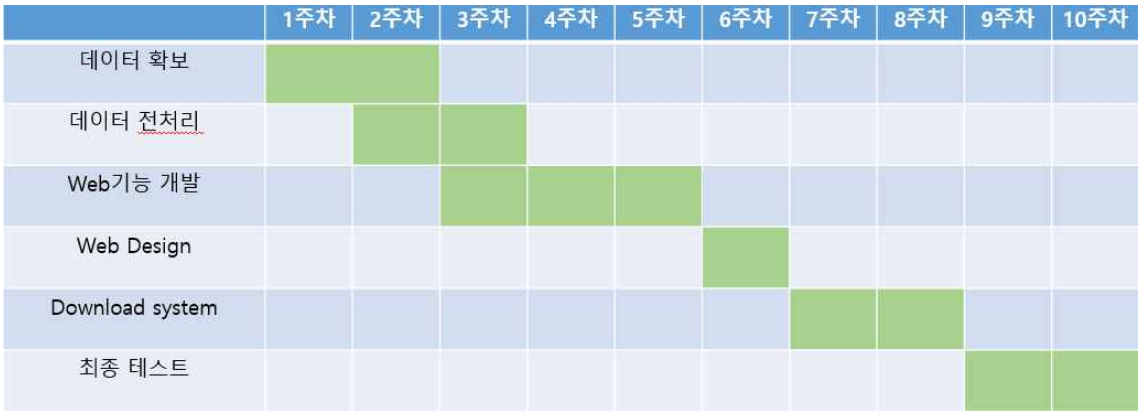
- 이승규

4/7 시작 6/14 종료(10주)	1주차	2주차	3주차	4주차	5주차	6주차	7주차	8주차	9주차	10주차
데이터 확보										
Pose Estimation 구현										
Car crash detection										
시스템 통합 및 테스트										

- 송창석

	1주차	2주차	3주차	4주차	5주차	6주차	7주차	8주차	9주차	10주차
데이터 확보										
데이터 전처리										
Object Detection 모듈 개발										
Pose Estimation 모듈 개발										
Car crash detection 모듈 개발										
최종 테스트										

- 위진



6. 개발 프로그램

▶ 개요

Accident detection service로써 차량 간의 사고와 화재 사고 또한 감지하고 이를 알려주는 서비스입니다. 교통사고는 언제나 대두되고 있는 문제입니다. 특히 대형 사고나 단독사고 등에서 더욱 높은 사망률을 나타내고 있습니다. 이때 사고에 대한 빠른 대처가 늦으면 운전자의 생사는 알 수 없을 것입니다. 이로 인해 사고를 어디에서나 빠르게 인지하고 대처할 수 있는 프로그램의 필요성이 나타나고 있습니다.

이를 위해 Accident detection service를 개발했습니다. 본 서비스는 사고를 감지하는 Deeplearning-model로 cctv영상을 받아 현장의 차량 사고를 빠르게 인지한 후 알림을 보내주고 , 현장에 대한 정보를 다운로드 할 수 있게 합니다.

▶ 주요 기능

Accident detection service의 주요 기능은 차량과 보행자를 자동 인식하고 Object Detection과 Pose Estimation 정보가 마스킹 된 영상을 출력해주고 차량 파손 감지와 차량 화재 감지를 통한 구조가 필요한 상황에 대해서 User에게 알람을 주어 경고합니다.

▶차량 파손 감지(car crash Detection)

차량 파손 감지 (Car Crash Detection)은 먼저 어떻게 하면 차량의 사고를 판단할 수 있을까란 생각을 먼저 하였습니다. 먼저 차량의 사고가 일어났다면 먼저 차량의 외관에 파손이 관찰할 수 있을 것이라는 생각을 먼저 해 cctv를 이용하여 차량들을 object detection한 것들을 CNN을 활용한 파손된 차량 이미지와 정상 차량 이미지 둘을 구분하는 Binary Classification을 해보려고 하였습니다. 하지만 이 방식으로 하면 매 프레임마다 Object

Detection으로 찾은 모든 차량에 대해서 Deeplearning-model을 이용한 연산이 요구되기 때문에 지나치게 많은 연산량으로 구현하기가 힘들다고 판단하였고 차량의 사고를 판단하기 위한 또 다른 기준이 필요하였습니다. 그래서 유튜브 등에서 차량의 사고 영상들을 시청하며 공통점을 찾기 위해 노력하였는데 사고가 발생한 차량은 좀 더 특정 지어서 말하면 저희가 목표로 하는 인명구조가 필요한 수준의 사고가 발생한 차량들은 사고 이후에 더 이상 주행을 하지 못하고 멈춰 있다는 공통점을 발견하였고, 움직임이 없는 차량을 대상으로 파손 감지를 하게 되면 필요한 연산량이 충분히 줄어들 것이라고 생각을 했고 object detection을 이용해 얻어낸 차량들의 위치좌표들을 활용하여 차량의 정지상태를 파악하고자 하였습니다.

정리하면 차량의 충돌 사고 여부를 판단하기 위해서 2가지 기준을 모두 만족하면 사고로 의심할 수 있다고 판단하기로 하였는데 1. 차량의 파손된 상태 2. 차량의 움직임이 없을 것 이 두가지 조건을 만족하면 사고로 의심되는 상황이므로 User에게 경고를 하는 것으로 설계하였습니다.

먼저 동영상에서 초당 10프레임 정도로 이미지를 추출하여 Object Detection 모델을 사용하여 데이터들을 얻어내는데 제자리에 있는 차량 같은 경우에는 계속 같은 이미지 위치에 있으므로 좌표가 같게 나온다고 생각을 하고 위 그림에서와 같이 차량에 사각형으로 차량이 있는 이미지 영역이 Detection되는데 좌상단의 좌표가 일정하게 같은 좌표로 나온다면 정지해 있다고 판단하려 했으나 object detection이 멈춰 있는 차량이라 할지라도 주위의 상황의 변화로 인해 조금씩 좌표가 다르게 나오는 것 때문에 상하좌우로 1%의 오차를 두고 그 영역 내에서 다음 프레임에 좌표가 또 Detection된다면 차량이 정지해 있다고 판단하려 했고 Buffer를 이용하여 연속된 10개의 프레임 중 9개 이상의 프레임에서 같은 위치의 좌표가 나온다면 차량이 정지해 있다고 판단하였습니다.

두번째로 차량이 파손되었는지를 판단하기 위해서 따로 이미지를 binary Classification하는 deeplearning-model을 훈련시켜 사용하였는데 DenseNet201을 기반으로 3000장 정도의 손상된 차량과 정상 차량 이미지를 구분하는 훈련을 진행하여 validation accuracy를 92.8%가 나오도록 훈련시켰습니다. 이전에 설명했던 object detection을 통해 얻어낸 좌표를 통해 정

지 상태에 있는 차량이 존재하는 이미지 영역을 Crop하여 정지해 있는 차량의 이미지만을 deeplearning-model을 이용하여 차량의 손상 여부를 판단하고 차량이 손상되었다고 파악되면 User에게 경고를 주고 사고의 영상과 데이터를 다운로드할 수 있도록 해줍니다.

▶차량 화재 감지(car fire detection)

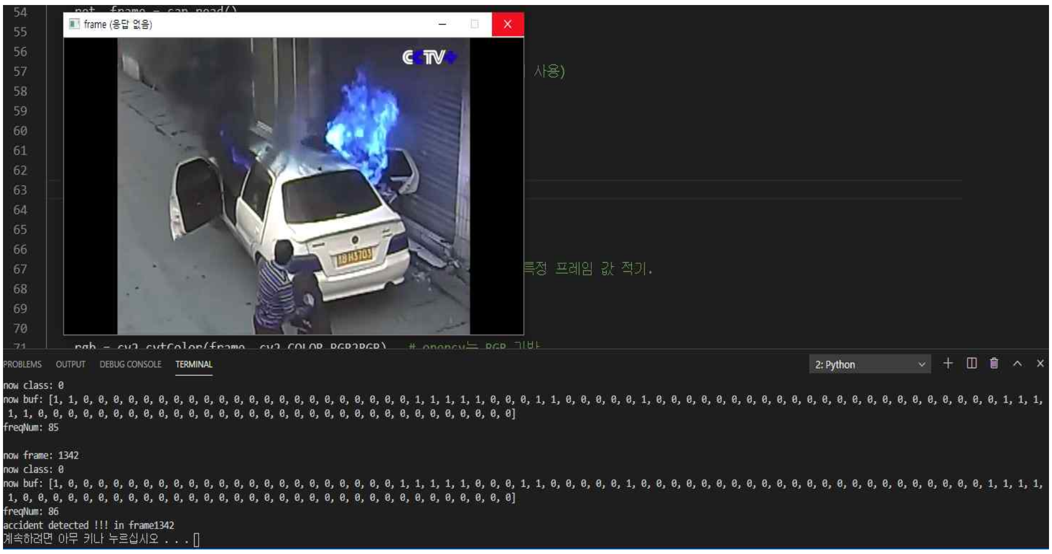
예측할 때는 결국 우리가 학습하여 만들어 낸 frozen graph를 활용하며, opencv에서 영상의 각 프레임을 받아서 지속적으로 입력 이미지에 넣고, predict_classes를 통해 accident 인지 non accident 인지에 대한 산출 결과를 얻어내는 것이므로 크게 설명할 부분은 없다. 다만 하나의 프레임을 통해서 사고를 감지하는 것은 불안정하므로 다음과 같은 약간의 로직을 추가하였다.

" 현재 프레임으로부터 100개의 이전 프레임에 대한 버퍼를 유지함. 전체 버퍼 중 accident(0)이 threshold인 80번 이상이 되면 화재로 판단함. “

```
now frame: 34
now class: 0
now buf: [1, 1, 0, 0, 0, 0, 0, 0, 0, 0]
freqNum: 8

now frame: 35
now class: 0
now buf: [1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
freqNum: 9
accident detected !!! in frame35
```

위의 이미지는 10 크기의 버퍼 중 9개의 frequency로 테스트 했던 결과이다.



차후에는 100개 크기의 버퍼로 구성하여 위와 같이 여러 테스트 영상에서 성공적으로 차량

화재를 감지하였음을 확인하였다. 위에서는 불의 색깔이 파란색으로 나오는데 이는 opencv의 기본 컬러 렌더링 옵션이 BGR이기 때문이다. 학습 및 테스트 시에는 정상적인 RGB 방식이 적용되었으므로 문제가 없다.

▶ 구현 결과(메뉴얼)

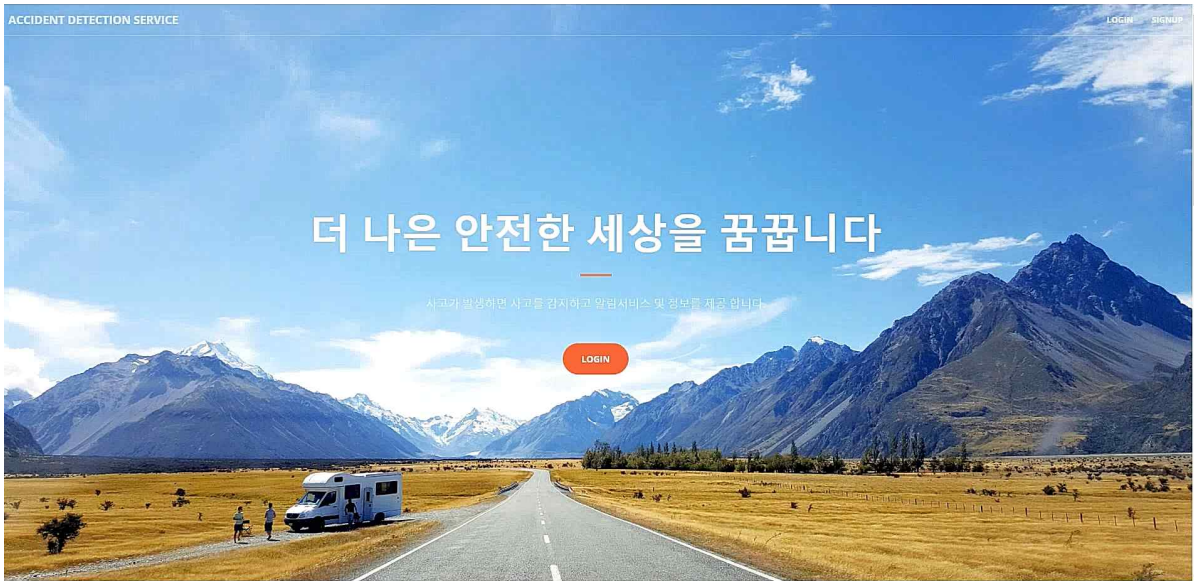


그림 1. 첫 웹페이지 화면

[그림 1]은 accident detection service의 첫 페이지 화면이다. 어떤 서비스를 제공하는지에 대한 내용을 담고 있으며 우측상단의 signup버튼, 중앙과 우측 상단에 login버튼이 있다. 회원가입이후 로그인을 통해 본 서비스 이용할 수 있다.

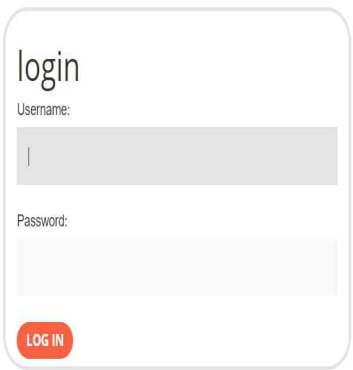


그림 2. 로그인 화면

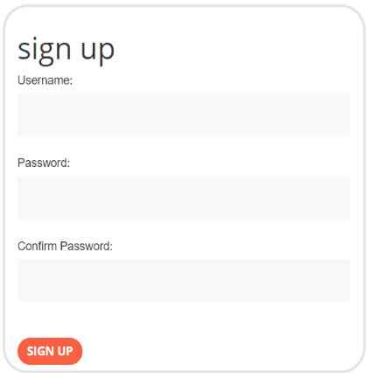


그림 3. 회원가입 화면

앞서 언급했던 login과 signup의 버튼을 누르게 되면 [그림 2] 와 [그림 3] 화면을 볼 수 있다. 아이디와 비밀번호, 비밀번호 재확인을 입력하면 회원가입이 가능하며 이후 일치하는 아이디와 비밀번호를 입력하면 서비스를 이용할 수 있다.

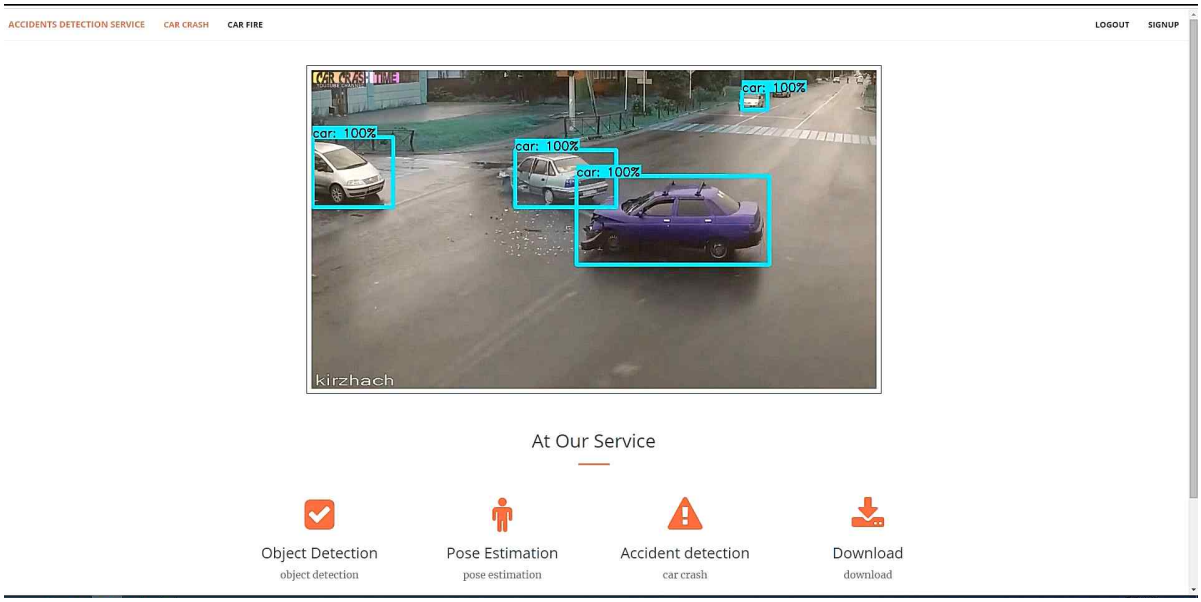


그림 4. 자동감지(Car Crash)

[그림 4]는 로그인 이후에 보이는 "car crash" 메인 화면이다. 좌측 탭의 car crash가 주황색으로 활성화 되어 있는 것을 확인 할 수 있으며 car crash탭 우측에 있는 car fire탭을 누르면 [그림 4]와 동일한 car fire 메인 화면으로 넘어간다.

Car crash는 영상에서 차량들의 Object Detection을 이용하여 얻어낸 차량의 좌표를 이용 사고가 의심되는 차량이 있는 것으로 의심되는 영역의 이미지를 잘라내어 차량의 손상을 검사하여 사고 유무를 감지해낸다.

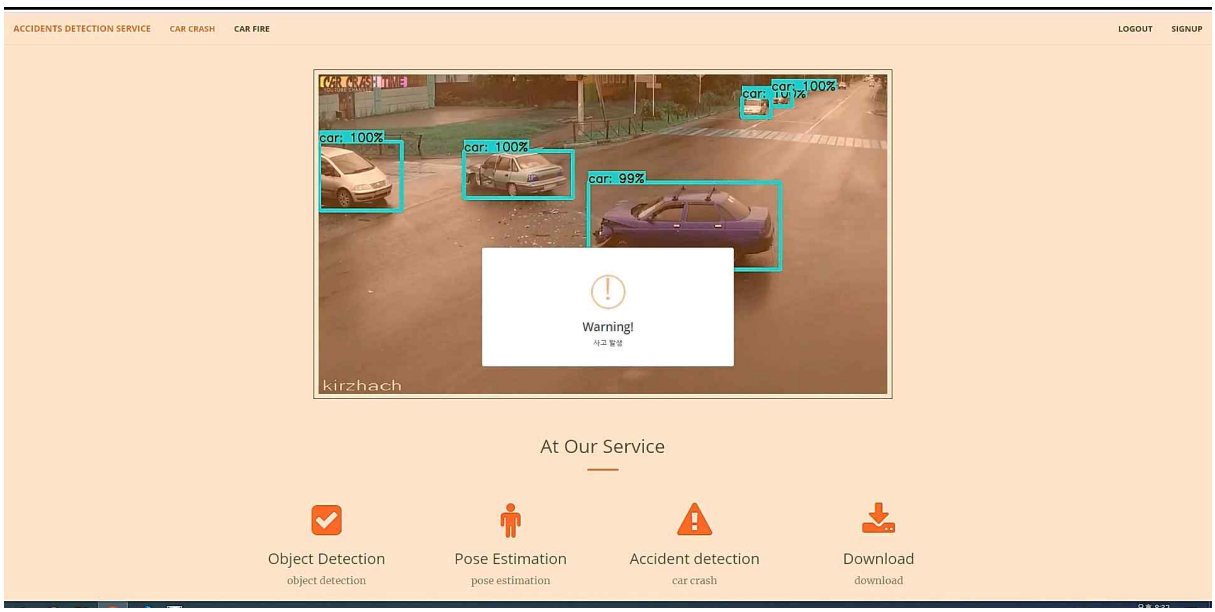


그림 5. 사고 인지 알람(Car Crash)

사고를 인지하면 [그림 5]처럼 warning알람으로 사고가 발생했음을 User에게 알린다. 이후 알람 modal창이 사라지고 download modal창이 보이고

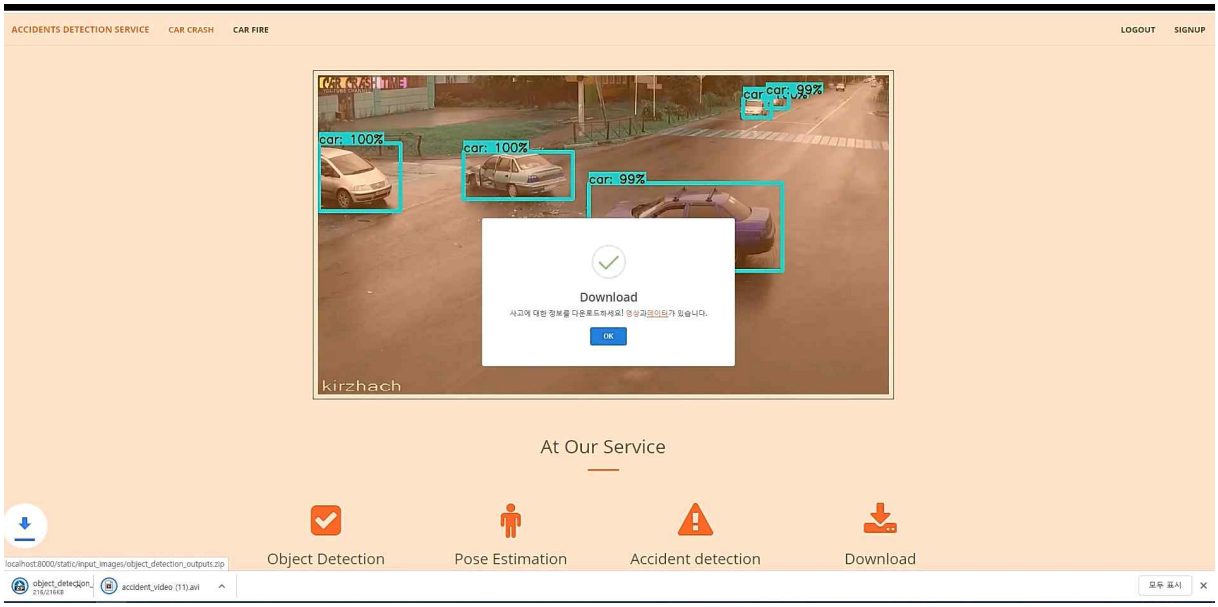


그림 6. 다운로드 기능(Car Crash)

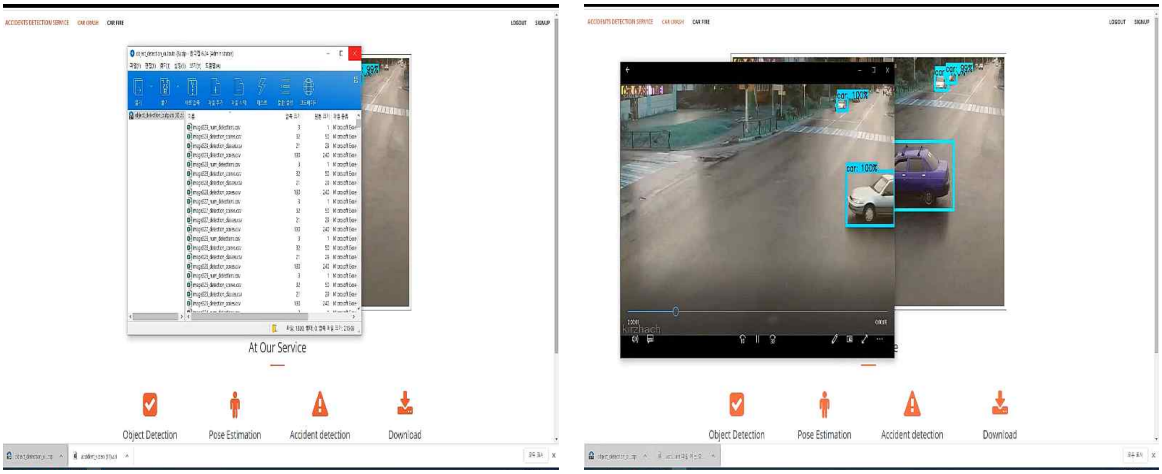


그림 7. Csv 파일 및 영상 다운로드(Car Crash)

이후 창에서 10초 길이의 사고 관련 영상과 관련 정보가 담긴 csv파일을 다운로드를 할 수 있다.
(참고) [그림 6], [그림 7]

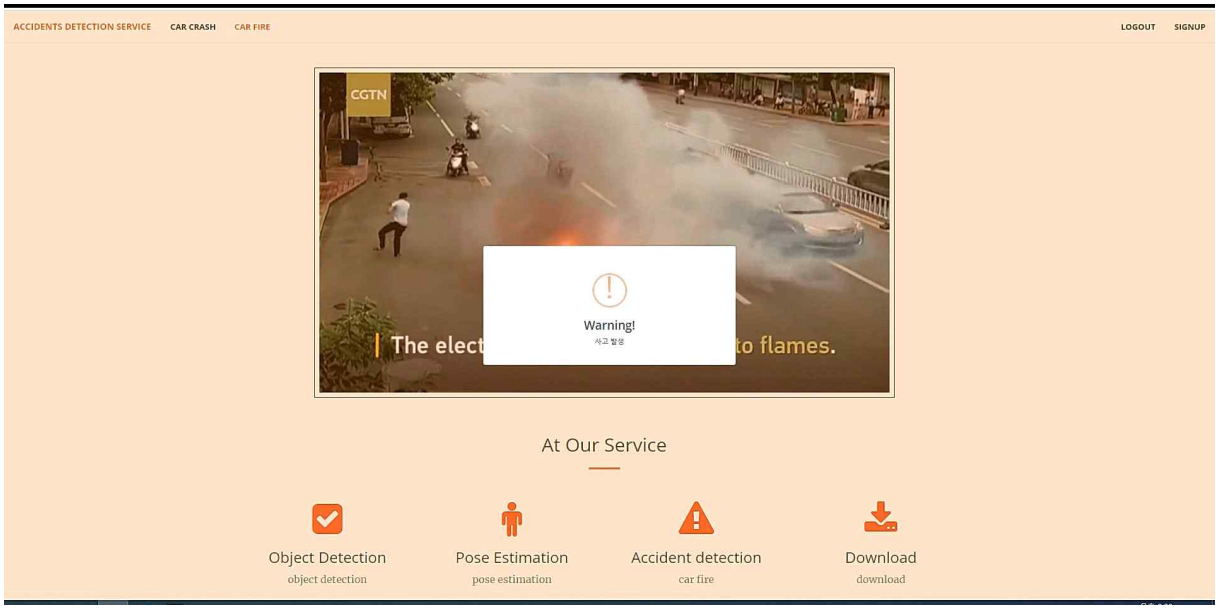


그림 9. 사고 인지(Car Fire)

[그림 4]에서 보았든 좌측 상단에 있는 car fire 탭을 누르게 되면 화재감지 탭으로 넘어가게 되고 위 그림[그림 9]는 영상 내 화재를 감지하고 알림을 주고 있는 모습이다.

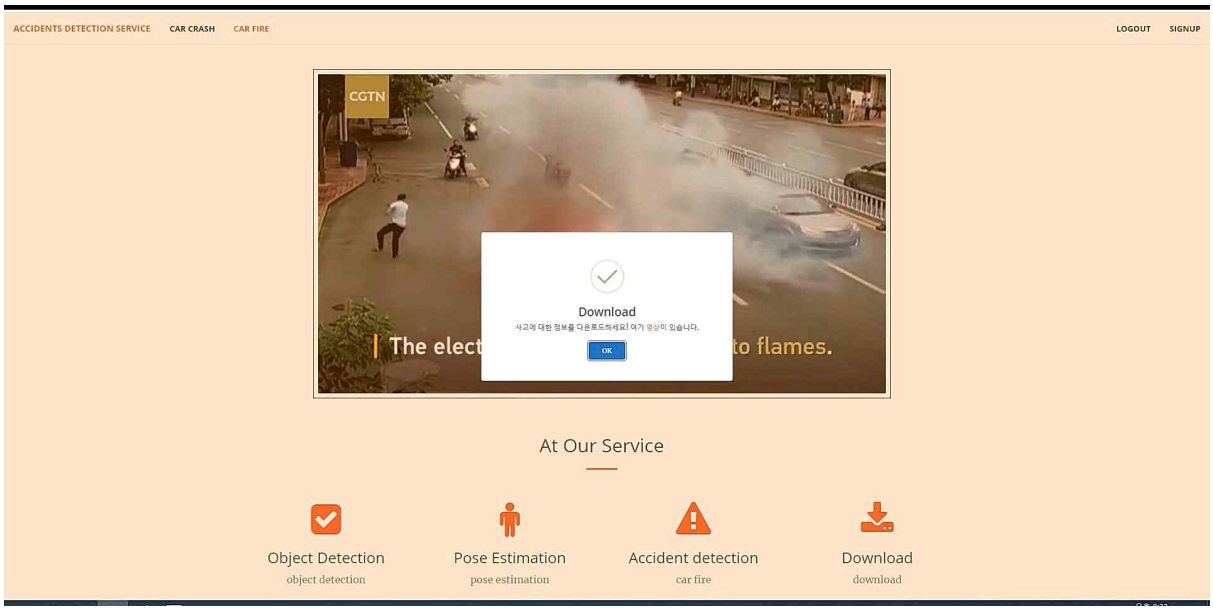


그림 10. 영상 다운로드(Car Fire)

화재 감지 또한 10초 길이의 사고 관련 영상을 다운로드 할 수 있다.

▶제약사항

- 차량 파손 탐지(car crash Detection)

현재로서 차량 파손 탐지에는 몇 가지 제약 사항들이 있는데 첫번째로 차량이 정지돼 있는 상태가 카메라 안에 잡혀야 사고를 감지할 수 있는 구조기 때문에 차량이 충돌 사고 후에 튕겨져 나가 CCTV의 촬영 범위 밖으로 나가버리면 차량의 사고를 인식할 수 없다는 제약 사항이 있고 또 차량의 파손이 너무 심각한 상태여서 Object Detection Model이 손상된 차량을 차량으로 인식하지 않는 경우에도 사고를 감지할 수 없다는 점이 있습니다. 그리고 멀리 있는 차량 같은 경우에는 이미지 자체가 너무 작아 제대로 판단을 할 수 없다는 문제가 있어 현재로서는 Object Detection된 차량의 영역 넓이를 이용하여 판단이 원활히 가능한 크기 이상의 차량들만 차량의 파손을 검사하고 있으므로 CCTV에 명확히 가까이 찍힌 사고들은 제대로 인식을 하지만 멀리 있는 차량들의 사고들은 제대로 탐지하기가 쉽지 않은데 이는 CCTV의 해상도를 높이면 어느정도는 해결될 수 있을 것으로 보입니다. 그리고 혹시나 충돌 사고와는 관련이 없는 손상된 차량이 주차되어 있는 상황 같은 경우에 CCTV에 계속 잡혀도 이를 계속 경고해주므로 문제가 발생할 소지가 있어 특정한 상황에 대한 것은 무시를 시킬 수 있는 기능도 필요할 것으로 보입니다.

- 차량 화재 탐지(car fire detection)

완성된 테스트 코드에는 약간의 제약 사항이 존재한다. 일단 화재의 크기가 너무 작아서 이미지 내에서 10픽셀 이하 정도로 작게 차지하는 경우에는 잡기가 힘들다. 사실 이 경우는 사람의 눈으로도 애매한 상황이므로 입력 이미지의 해상도를 높이지 않는 이상 크게 개선될 여지는 없어 보인다. 그리고 불과 비슷한 색감을 가진 객체를 화재로 오인하는 경우도 간간히 있다. 불의 색상에 대해서는 어느정도 학습이 되었지만, 불이 가지는 패턴에 대해서는 아직 학습이 부족한 것으로 여겨진다. 그래도 대부분의 상황에서는 이러한 오류가 거의 없으므로 추가적인 튜닝을 통해 차후에는 해결할 수 있을 것이다.

▶ 결론

교통사고 문제가 계속해서 대두되고 있고 이 가운데 빠른 사후 처리는 더욱 중요해지고 있다. 더 나은 안전한 세상을 만들고자 하는 본 프로그램은 Deeplearning-model을 이용하여 영상내 사고를 자동으로 인지하고 이는 교통 사고 이후 신속한 대처를 가능케 하고 있다. 본 프로그램을 통해 교통사고 피해자의 골든 타임을 확보하여 안전을 도모할 수 있을 것이다. 또한 사고의 상황에 대한 정보를 통해 연구자료 확보 역할을 수행할 것이다. 마지막으로 기존의 cctv에 추가 설치없이 cctv 영상을 관리하는 주체에서 사용하여 보다 나은 가용성을 확보, 무리없이 자동사고인지 서비스를 사용 할 수 있을 것이다.

7. 설계 구성 요소

설계 구성 요소	**목표 설정**	CCTV로 촬영된 영상을 실시간으로 딥 러닝 모델로 분석하여 차량 관련 사고 발생 시 사용자에게 웹을 통해 알림을 보내며, 관련 데이터와 마스킹 된 영상을 다운로드 가능하게 해 주는 기능을 개발
	합성	웹 관련 기능, 사고 발생 탐지 기능이 필요하다고 판단하였고, Object Detection과 Pose Estimation을 관련 데이터로 선택하였다.
	분석	요구사항 명세서를 작성한 뒤, 유스케이스, 클래스, 시퀀스 다이어그램을 통하여 구성 요소를 분석하였다.
	제작	구성 요소를 모두 분석한 뒤, 오픈 소스를 활용하여 제작하였다.
	시험	테스트용 이미지를 통해 분석하였다.
	평가	최종 성능 : Object Detection(speed :79ms, Pascal mAP@0.5 : 87) Car Crash Detection (FPS : 3.10, Accuracy : 92.8%) Car Fire Detection (FPS : 6.53, Accuracy : 92.9%)
제한 조건	**산업 표준**	해당사항 없음
	경제성	딥 러닝 모델의 실시간 작업을 위해서 GPU 필요
	안정성	웹을 통해 사용자가 접근하므로, 운영체제가 달라도 사용 가능하다.
	미학	Sweet alert, bootstrap을 활용, 심플한 외형을 추구하여 호불호가 갈리지 않는다.
	사회 영향	경찰청, 아파트 경비실 등 CCTV를 관리하면서 사고 발생의 유무를 판단하는 사람들에게 사용될 것이며, 사고가 발생했는 지를 실시간으로 탐지하여 골든 타임 내에 구급차가 도착하는 등의 효과를 기대할 수 있을 것이다.

8. 팀 목표 대비 달성 정도

▶ 사고상황 식별: 90%
30가지 상황 중 27가지의 상황을 정확하게 판단하였다.

▶ 구체적인 데이터를 제공: 100%
cctv에 촬영된 차량, 사람의 좌표, 화재 발생 여부, 사람의 pose좌표 등 다양하고 구체적인 데이터를 제공한다.

9. 기대효과

저희가 개발한 차량 충돌 및 화재 감지 시스템을 사용하면 고속도로나 터널의 CCTV를 이용하여 사고를 감지하고 신속하게 관련 기관에 전달하여 운전자의 생명을 보전하게 할 수 있는데 이를 이용하면 인적이 드문 도로나 밤 늦은 시간에 운전자가 사고가 발생하여 스스로 구조 요청을 할 수 없는 상황일 때 CCTV를 이용하여 사고를 감지, 구조에 도움을 줄 수 있을 것입니다. 또한 사고 상황에서의 차량 위치, 움직임 등의 데이터를 제공해 줌으로써 이를 분석하고 이용할 수 있을 것으로 보입니다. 이를 통하여 현재 구조를 보내지 못하는 상황에서의 구조율을 높일 수 있을 것이고 CCTV를 통해서 계속하여 사고 감지가 가능하므로 사고 이후 사람의 신고보다 더욱 빠른 구조 요청을 함으로써 교통사고 상황에서의 생존률을 더 높일 수 있을 것으로 예상됩니다.

10. 개선 방향

본 프로젝트의 한계점을 해결하고 더 나은 방향에 대해 생각한다.

car crash : 차량과 차량 간의 사고를 인지할 때 사고 난 차량이 cctv 밖으로 나가거나 차량 파손 정도가 심하면 사고를 인지하기 힘든 부분이 있다. 파손 정도에 대한 data 를 확장시켜 training 을 할 필요가 있고 사고인지 알고리즘은 개선할 필요가 있다.

car fire : 현재 화재의 정도가 적은 화재 사고를 잘 인지하지 못한다. 이를 위해 많은 학습데이터를 통해 모델 개선이 필요하다.

pedestrian accident detection : 추후에 차량과 보행자 간의 사고 또한 감지할 수 있는 학습데이터를 마련하고 모델을 구축할 필요가 있다.

웹-서버 : prediction 속도를 높이기 위해 서버의 computing power 를 개선하고 클라이언트와 서버를 명확히 구분하여 본 서비스를 이용하는 user 에게 보다 나은 접근성을 부여할 필요가 있다.

관련기관 신고 관련 오픈 API : 사고를 감지한 이후 신고 관련 오픈 API 를 통해 보다 신속하게 관련기관에 연락을 취할 수 있게 할 필요가 있다.

11. Q&A

문: 이 프로젝트에서 Recall이 중요한가 Precision이 중요한가?
답: 이 프로젝트는 사고가 난 상황에서 인명구조에 도움이 되는 것을 목표로 하고 있다. 사고가 발생하지 않았을 때 사고로 파악하는 오류보다 사고가 발생하였는데 사고를 파악하지 못하는 경우가 더욱 치명적이라고 생각한다. 따라서 Recall을 높이는게 더 중요하다고 본다.

문: 데이터셋 구성 비율은 어떠했는가?
답: 사고인 상황 20개, 사고가 아닌 상황 10개로 구성되었다. 이중 3개 상황에서 정확한 판단을 보이지 못했는데 사고가 아니었는데 사고라고 판단한 상황이 2건, 사고였는데 사고를 파악하지 못한 상황이 1건이었다.

문: 큰 사고가 아닌 작은 접촉사고같은 경우도 잡을 수 있나?
답: 현재 학습된 모델로는 쉽지 않을 것이라고 생각한다. 그것을 잡아내려면 더 많은 데이터셋이 필요하고 학습 또한 더 시켜야 할 것이다. 현재 차량의 손상 여부로 사고를 파악하는데 작은 손상의 경우 cctv마다의 해상도 문제 또한 있을 것이라고 생각한다.
이 프로젝트의 목적 또한 큰 사고로 인한 인명을 구조하는데 도움을 주겠다는 목적이 있는데 작은 접촉 사고는 의미가 없다고 판단하여 초기 계획에 포함시키지 않았었다.

12. sw 프로그램 등록

13. Github 등록 내용 요약 및 등록(Github에 등록한 source는 별도의 압축파일 제출)