Date 2020-7-20

Team13

Software Requirement Specification

for Smart Traffic Light System

201411285 유종혁

201411286 유환규

201611308 최준오

201714166 신예슬

목차

1. Introduction

1.1 Purpose

딥러닝 기술을 이용한 효율적인 신호등 시스템인 Smart Traffic Light System(STLS)을 구현하기 위한 요구사항을 명세한 문서이다. 본 문서는 Smart Traffic Light System을 개발하는 개발팀원들을 주요 독자로 한다. Smart Traffic Light System 개발자들은 명세서에 따라 기능을 설계 및 구현한다. 추후 본 프로젝트의 평가에 참여하는 사람들이 추가적인 독자가 될 수 있다.

1.2 Scope

Smart Traffic Light System 은 실시간 도로 상황을 촬영하고, 촬영한 사진을 분석하여 신호 스케줄링을 관리하는 시스템이다. 기존의 고정된 시간으로 하는 신호 시스템이나 교통량의 통계를 이용해서 시간을 정하는 시스템과는 다르게 실시간으로 신호 스케줄링을 정하면서 현재 교통상황에 알맞게 적용한다.

1.3 Definitions, Acronyms, and Abbreviations

용어	정의
SRS	Software Requirement Specification, 요구사항명세서
R-CNN	Regions with Convolution Neural Network
Faster R-CNN	기존의 R-CNN구조를 계승하면서 selective search를 제거하고 RPN을 통해서 RoI를 계산하는 딥러닝 기술
Bounding Box Regression	selective search를 통해 찾는 박스의 위치는 부정확하기 때문에 성능을 끌어올리기 위해 박스의 위치를 교정해주는 것을 말한다.
RoI	Region of Interest
RPN	Region Proposal Networks, 이미지를 입력 받아 사각형 형태의 Object Proposal과 Object Score를 출력해주는 역할을 한다.
GUI	Graphic User Interface, 사용자가 편리하게 사용할 수 있도록 입출력 등의

	기능을 알기 쉬운 아이콘 따위의 그래픽으로 나타낸 것을 말한다.
Object detection	객체 검출, 객체라고 판단되는 곳에 직사각형(Bounding Box)을 그려주고 그 객체가 무엇인지 분류하는 것을 말한다.
DESC	Description
DEP	Dependency

표 1. Definitions, Acronyms, and Abbreviations

1.4 References

IEEE Std. 830-1998

Rich feature hierarchies for accurate object detection and semantic segmentation

Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

1.5 Overview

본 문서의 구성은 다음과 같다. 2장에서 Smart Traffic Light System의 일반적인 기술사항을 알아본다. 그를 위해 제품의 관점, 제품의 기능, 사용자 특성, 제약사항, 가정 및 의존성 등의 각 절로 나누어 살펴본다. 3장에서는 Smart Traffic Light System의 상세한 요구사항을 알아본다.

2. Overall Description

2.1 Product perspective

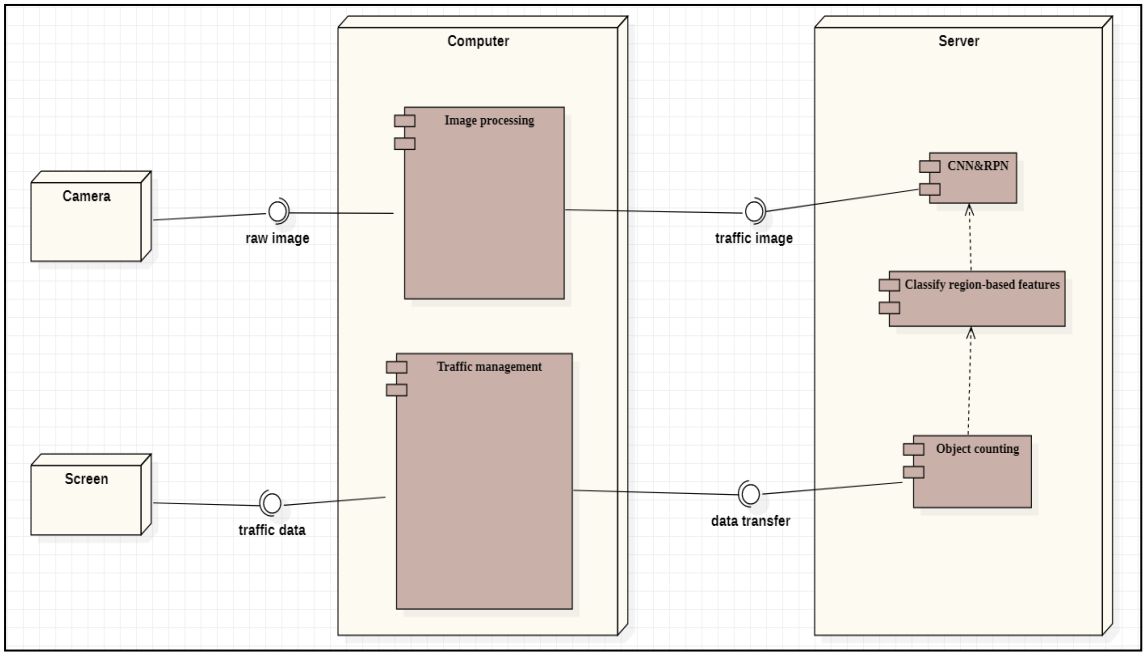

Figure 1. Component Diagram

딥러닝 기술을 도입하여 현재 도로 신호 체계를 보다 효율적으로 개선시킬 수 있는 시스템이다. 시스템은 신호를 스케줄링 하는 컴퓨터(A)와 도로 상황을 분석하는 서버(B)의 두 가지 파트로 구성되어 있다. 카메라를 이용해 도로 상황을 촬영하면 이를 A에서 이미지의 전처리(Image processing)를 진행 후 B에 전달하고 딥러닝 모델을 활용해서 객체 탐지(CNN&RPN / Classify region-based features)를 진행한다. 그 결과를 이용하여 현재 도로의 교통량을 파악(Object counting)하고 A로 보내 신호 스케줄링 시스템(Traffic management)에 반영한다. 각 도로신호와 보행자신호는 모니터(Screen)상의 GUI로 표현된다. 도로 상황 촬영에는 4개의 카메라가 사용되며 객체 탐지를 수행할 딥러닝 모델은 클라우드 기반으로 운영된다. 시스템의 동작을 확인하기 위한 테스트는 사거리의 도로상황을 표현할 세트장을 제작하여 진행한다.

2.2 Product functions

2.2.1 도로 상황 촬영 및 전송

- 각 도로를 카메라로 촬영하여 전송한다.
- 캡처는 0.5초 간격으로 동작한다.

2.2.2 Bounding Box Regression

- 객체의 bounding box 탐지한다.
- 물체가 있을 법한 곳의 중복을 제거하고 위치를 보정한다.
-

2.2.3 이미지 속 객체 classification

- 이미지 속 추출된 객체를 분류기를 사용하여 정의한다.
- 정의할 객체의 종류는 일반차량, 사람, 구급차로 3가지이다.

2.2.4 객체의 개수 counting

- 이미지에서 분류한 각 객체의 개수를 count 한다.
- count한 객체는 (객체 종류, 개수)의 쌍을 갖는 변수로 저장한다.

2.2.5 신호 표시

- 각 도로 신호와 보행자 신호를 표시한다.

2.2.6 신호 스케줄링

- 분석된 객체들의 수를 기반으로 교통신호체계를 조정한다.
- 신호 유지시간을 변경하거나 새로운 신호 상황을 추가한다.
- 스케줄링 알고리즘은 다음과 같다.

2.2.6.1 기본 흐름

(1) 처음 신호등부터 다시 켜질 때 까지를 하나의 사이클로 생각한다.

(2) 각 도로별로 정지된 자동차와 보행자의 수를 이용해서 신호 우선순위의 가중치를 증감시킨다.

(3) 가중치를 기반으로 다음에 켜질 신호의 유지 시간을 결정한다.

2.2.6.2 예외 상황

(1) 신호등에 차가 없을 경우나 보행자가 없을 경우 신호등 순서를 생략할 수 있다.

(2) 구급차가 진입 시, 기존 신호를 10 초간 유지 후 구급차 진입 도로에 10 초간 신호를 부여한다. 그 후 정상적인 스케줄링으로 복귀한다.

2.3 User characteristics

2.3.1 User Class(stakeholders)

2.3.1.1 신호관리자

: 신호 관리자는 사거리의 신호시스템을 모니터링 사람이다. 4개의 차량신호등과 4개의 보행자신호등을 관찰하고 있다. 제공되는 한 개의 노트북으로 시스템을 관리한다.

2.3.1.2 일반자동차 운전자

: 일반자동차 이용자는 교통신호등을 보고 운전을 한다. 운전자는 직진신호와 좌회전신호에 반응하여 그에 해당하는 행동을 한다. 3차선을 이용하며 적색 신호에는 정지해 있고 녹색 신호에는 차선에 맞게 직진 또는 좌회전을 한다. 황색 신호에서는 이미 통과하는 중이면 지나가고 아니면 정지한다.

2.3.1.3 구급차 운전자

: 구급차 운전자는 차량 운전자이다. 대부분 구급차는 신속하게 이동해야 하는 상

황이기 때문에 신호가 빠르게 바뀌기를 바란다.

2.3.1.4 사람

: 여기서 사람이란 횡단보도를 건너는 사람을 말한다. 그들은 횡단보도 신호등이 녹색신호일 때 횡단보도를 건넌다. 보통의 경우 자전거를 타거나 걷는다.

2.4 Constraints

2.4.1 Regulatory policies

도로교통공단 기준의 신호 정책

2.4.1.1 차량신호등

(1) 녹색의 등화: 차는 직진 또는 우회전할 수 있다.

(2) 황색의 등화: 차는 정지선이 있거나 횡단보도가 있을 때에는 그 직전이나 교차로의 직전에 정지하여야 하며, 이미 교차로에 차의 일부라도 진입한 경우에는 신속히 교차로 밖으로 진행하여야 한다.

차는 우회전할 수 있고 우회전하는 경우에는 보행자의 횡단을 방해하지 못한다.

(3) 적색의 등화: 차는 정지선, 횡단보도 및 교차로의 직전에서 정지하여야 한다. 다만, 신호에 따라 진행하는 다른 차의 교통을 방해하지 아니하고 우회전할 수 있다.

(4) 녹색 화살표의 등화: 차는 화살표시 방향으로 진행할 수 있다.

2.4.1.2 보행신호등

(1) 녹색 신호등: 보행자는 횡단보도를 횡단할 수 있다.

(2) 녹색 등화의 점멸: 보행자는 횡단을 시작하여서는 안 되고, 횡단하고 있는 보행자는 신속하게 횡단을 완료하거나 그 횡단을 중지하고 보도로 되돌아와

야 한다.

(3) 적색 신호등: 보행자는 횡단보도를 횡단하여서는 아니 된다.

2.4.2 Hardware limitations

2.4.2.1 어플리케이션 제약사항

(1) RAM: 4GB

(2) HDD: 5GB

(3) GPU: GTX 1060

(4) 네트워크: 딥러닝 모델 이미지 전송이 필요하므로 네트워크 연결이 필요하다 따라서 NIC 카드 필요함.

2.4.2.2 딥러닝 모델 학습 서버 제약사항

(1) GPU: 모델 학습을 위하여 최소 GTX 1080이상 필요로 한다.

(2) RAM: 대량의 이미지를 이용하여 학습해야 하므로 16GB 이상의 메모리가 필요하다.

2.4.2.3 세트장 제약사항

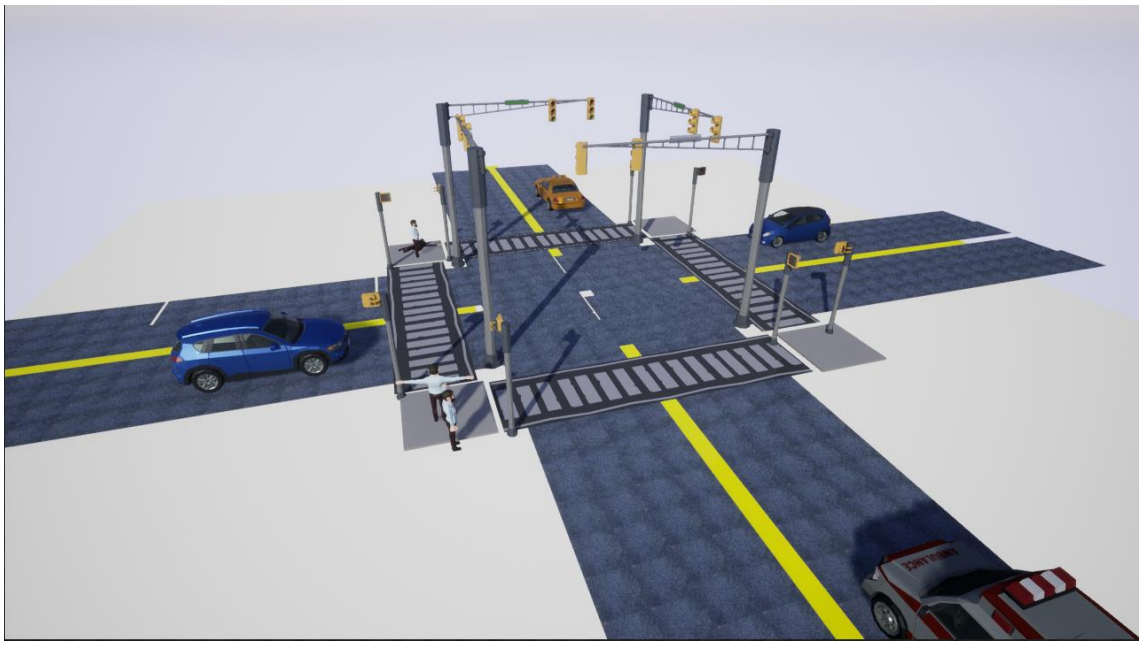

Figure 2. 세트장 모델링

(1) 세트장 공간 제약사항: 세트장 크기는 1.5M * 1.5M로 한다.

(2) 필요한 카메라 수: 네 방향의 차선 및 횡단보도 정보를 얻어야 하기 때문에 4개의 카메라가 필요하다.

2.4.3 Higher-order language requirements

(1) 운영체제: Window 10

(2) 개발언어 및 API: Anaconda, Tensorflow, Keras API

(3) 개발환경: Jupyter Notebook, Google Colab, University Lab

2.4.4 Safety and security considerations

(1) 보행자신호는 20초이상 녹색신호를 유지해야만 한다.

(2) 녹색신호에서 적색신호로 변경될 때 황색신호는 반드시 2초이상의 시간을 두어야한다.

(3) 구급차 도착 시에도 신호를 바꿀 때에는 기존 차량이 지나갈 수 있도록 10초 이상 지나간 후에 신호 변경을 해야 안전하다.

2.5 Assumptions and dependencies

(1) 인공지능 신호등 시스템을 실제 도로에 적용할 수 없기 때문에 세트장을 구성하여 시뮬레이션 하는 것을 가정으로 한다.

(2) 도로는 각 방향당 3차선 도로이다.

(3) 차량신호는 4개의 상태를 가지고 4개의 상태는 순차적으로 바뀐다는 것을 가정하고 진행된다.

(4) 세트장에서는 모형차량을 이용할 예정이고 구급차와 사람 역시 사진이 부착된

모형을 사용한다.

(5) 세트장에서 신호등을 실제로 보여줄 수 없기 때문에 모니터 화면에 4개의 차량 신호와 4개의 보행자 신호를 UI로 보여준다.

(6) 모형 자동차는 실제로 움직일 수 없으므로 특정상황마다 직접 사람이 모형자동차와 모형 보행자를 배치하여 인공지능 시스템이 각 상황에 대하여 결정하는 것을 확인한다. 구급차가 나온 상황의 경우도 세트장에 직접 배치하여 상황을 테스트한다.

(7) 신호등 시스템의 디폴트 값은 보행자 신호를 20초, 차량 신호를 40초로 설정하고, 각 도로의 교통량에 따라서 각 신호등의 시간을 20초 ~ 60초 사이의 값으로 설정한다.

(8) 구급차 발견 예외사항에서 구급차 쪽 신호를 받았을 때 5초 안에 구급차는 신호를 통과한다고 가정한다.

(9) 보행자는 20초 안에 횡단보도를 다 건넌다고 가정한다.

2.6 Apportioning of requirements

(1) 요구사항들은 이후 버전으로 미루지 않고 이번 버전에서 충족할 예정이다.

(2) Object detection의 정확도가 떨어지는 경우 이후 버전에서 정확도를 더 올린다.

3. Specific requirements

3.1 External interface requirements

3.1.1 User interfaces

Figure 3. 유저 GUI 프로토타입

사용자 인터페이스는 Python GUI를 이용하여 제공하고 다음 인터페이스 요구사항은 다음과 같다.

Figure 3의 인공지능 신호시스템 관리 화면으로 들어오게 된다.

(1) 1번 항목은 현재 시스템 시간을 보여준다.

(2) 2번 테이블은 사거리의 차량 수와 보행자 수와 구급차 수를 제공하는 테이블로 0.5초마다 업데이트된 도로상황정보를 제공한다.

(3) 3번 항목은 Static time 버튼, Real time 버튼으로 Static time 버튼은 기본값으로 설정된 신호시간을 제공하고, Real time 버튼은 실시간 도로 상황을 분석해서 시스템에 설계된 알고리즘에 의해 신호시간을 동적으로 조절해준다.

(4) 보행자신호 버튼과 차량 신호 버튼을 제공하는데 보행자 신호 버튼은 Figure3, 차량 신호는 Figure4 화면을 나타낸다.

(5) 5번 테이블은 전체 신호등의 현재 신호등 상태, 부여시간, 흐른시간을 보여준다.

(6) 6번 항목은 현재 전체 도로에서 신호를 기다리고 있는 차량의 총합을 보여준다.

Figure 4. 도로 신호등 GUI

Figure 4는 세트장에서 보여주는 차량신호등4개를 UI로 보여주고 현재 신호상태와 각 신호에 부여된 시간정보와 현재 신호가 몇 초 흘러갔는지에 대한 정보를 제공한다.

Figure 5. 보행자 신호등 GUI

Figure 5는 보행자 신호등 4개를 표시해주고 신호상태와 부여된 시간 정보와 흐른 시간 정보를 제공한다.

3.1.2 Hardware interfaces

이 프로젝트는 시뮬레이션을 위한 프로그램이므로 직접적인 유저와의 하드웨어 인터페이스는 없다. 시뮬레이션을 위한 사진 촬영은 세트장안에 4대의 카메라로 찍은 다음에 개인 컴퓨터에서 사진을 분석해주는 서버에 사진을 전송한다.

3.1.3 Software interfaces

카메라로 촬영한 사진을 어플리케이션에서 resize 한 이미지를 Traffic image operation을 통해서 딥러닝 모델에 전송한다. 딥러닝 모델로부터 나온 객체 정보를 Data transfer operation을 통해서 어플리케이션에 전달한다. 객체 정보는 Dictionary 형식으로 주고받는다. 어플리케이션에서 UI controller에 신호시스템 정보를 전달하여 Traffic data operation을 통해 화면에 신호를 표시한다. Traffic image operation 과 Data transfer operation 에서는 네트워크 통신이 발생하고 Socket 라이브러리를 이용하여 서로 데이터를 주고받는다. 구체적인 장치 간의 인터페이스는 아래 정의하였다.

인터페이스 명	인터페이스 설명
Raw image	카메라로 찍은 도로 상황의 원본 이미지를 보내준다.
Traffic image	image processing에서 반환된 결과를 네트워크를 통해 업로드하고 서버에서는 해당 결과를 다운 받아 학습 및 테스트를 진행한다.
Data transfer	전송받은 이미지 분석을 통해 얻은 사람, 구급차, 자동차 객체 수 정보를 애플리케이션에 전송한다.
Traffic data	traffic management의 결과로, 현재 동작하고 있는 도로 신호등과 횡단보도 신호등의 상태를 보여준다.

표 2. Software interfaces

3.1.4 Communications interfaces

인공지능 신호시스템 어플리케이션과 객체 검출을 하는 인공지능 모델 서버 간의 통신이 발생한다. 어플리케이션에서 객체 검출 서버로 전처리가 완료된 이미지를 전송하고 그 서버는 이미지로부터 교통정보를 추출하여 다시 어플리케이션에 그 정보를 전달한다. TCP/IP 프로토콜을 사용하여 연결 지향적인 방식으로 이미지를 전송하고 교통정보를 받는다.

3.2 Functional requirement

3.2.1 도로 상황 촬영 및 전송

3.2.1.1 Functional requirement 1.1

ID: FR1

TITLE: 도로 상황 촬영

DESC: 도로 상황을 4대의 카메라로 촬영하여 메인 컴퓨터에 전송할 수 있어야 한다. 사진 안에는 도로의 차량들과 사진 기준 도로의 왼쪽 보행자들을 담고 있어야 한다. 사진은 0.5초 간격으로 찍고 전송할 수 있어야 한다.

DEP: None

3.2.1.2 Functional requirement 1.2

ID: FR2

TITLE: 이미지 전처리

DESC: 촬영된 사진을 딥러닝 분석에 적합하도록 사전 처리를 할 수 있어야 한다. 사진에서 필요한 부분만 잘라내고 딥러닝 모델에 입력으로 넣을 수 있도록 크기와 형태를 조절할 수 있어야 한다.

DEP: FR1

3.2.1.3 Functional requirement 1.3

ID: FR3

TITLE: 도로 상황 이미지 전송DESC: 전처리 작업이 끝난 도로 상황 이미지를 서버에 전송할 수 있어야 한다. 이때 4개의 도로 상황 이미지가 항상 일정하게 정렬되어 전송하여야 한다.

DEP: FR2

3.2.2 Faster R-CNN 모델 설명 및 객체 분류 Function 정리

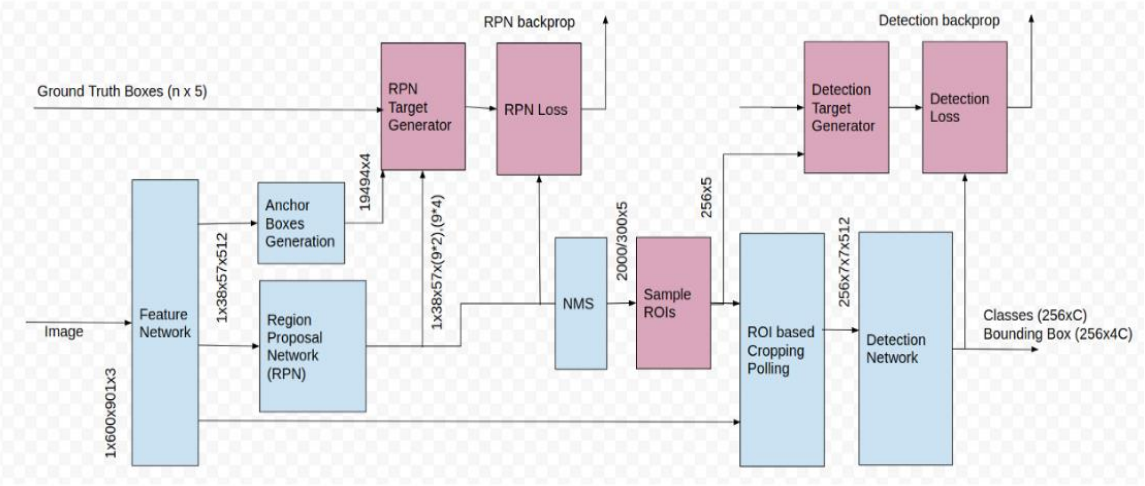

Figure 6. Faster R-CNN 전체 구조

DESC: FASTER-RCNN 전체적인 구조

객체 검출을 위해 구현할 전체적인 구조이다. 위에서 설명한 CNN을 통해 나온 Feature Map은 앞의 RPN Network로 들어가게 된다. RPN Network 에서는 객체가 있을 만한 구역을 제시할 수 있도록 학습되고 RPN Network를 통해 나온 Region Proposal과 Feature Network를 통해 전달된 Feature Map이 Detection Input으로 전달되어 Class와 Bounding Box가 Output으로 나온다. 이 Class와 Bounding Box 정보를 이용하여 사람, 구급차, 자동차 이미지를 식별하고 객체 수를 얻어 낼 수 있다. 각 세부 기능 요구사항은 다음과 같다.

3.2.2.1 Function Requirement 2.1

ID: FR4

Title: CNN

Input: 이미지(224x224x3)

Output: Feature Map

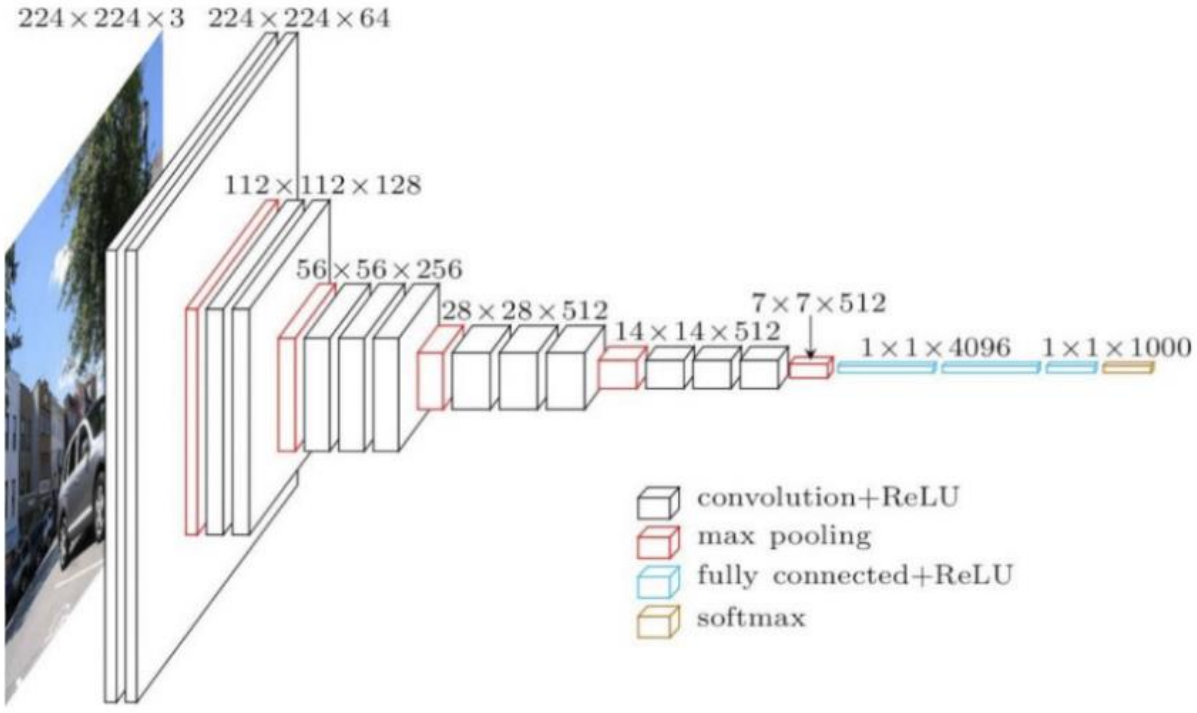

Figure 7. CNN 구조

DESC: VGG-16 Architecture의 구성

(1) CNN(Convolution Neural Network): 이미지를 인식하여 특징을 판단하고 어떤 물체인지 분류해내는 과정이다. Convolution + Subsampling + Convolution + Subsampling + ... + Full Connection 의 과정으로 이루어진다. Convolution 과 Subsampling 의 과정으로 이미지에서 특정 Feature 를 뽑아서 그 Feature 들의 조합으로 물체를 구분해낼 수 있다. (Feature Extraction) Fully Connected Layer 에서는 Subsampling 의 한 방법인 Pooling 의 결과를 벡터로 나타낸 다음 n 차원의 벡터로 바꾸는 과정을 해준다.

Faster R-CNN 은 CNN 을 통해 뽑아낸 Feature Map 을 입력으로 받는다.

이때 Feature Map 의 크기를 가로(H)*세로(W)*채널(C)로 잡는다.

(2) Convolution Layer: 이 Layer 에서는 3X3 Convolution Filter 을 이용하여 이미지 왼쪽 가장자리부터 슬라이딩 윈도우를 사용하여 필터 계산을 한다. 이때 Input 으로 받는 이미지 혹은 Feature Map 의 채널 크기만큼 depth 를 가지고 있어야한다. 이미지는 여러 개의 Feature 을 갖기 때문에 Convolution Filter 는 3X3X(채널 수)X(Feature 수) 가 된다. Convolution 결과로는 Stride 를 1 padding 1 로 하였을 때 Feature map 은 Height X Width X (Feature 수)가 되고 Feature 수는 여기서 채널로 볼 수 있다.

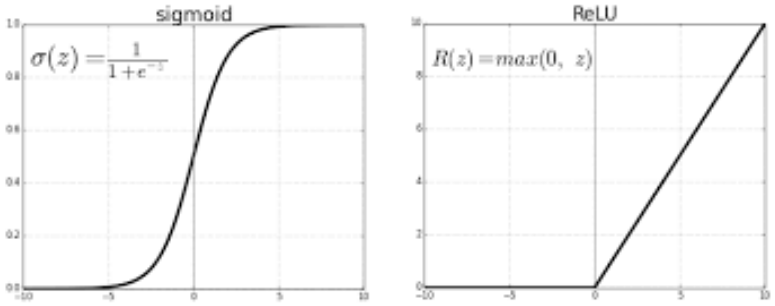

Figure 8. Activation function

(3) RELU 함수: convolution layer 에서 합성 곱을 한 후에 RELU 활성화 함수 계산한다. 이렇게 하여 Neural Network 의 비선형성을 부여한다. Sigmoid 함수를 사용하지 않는 이유는 vanishing gradient problem (기울기 손실)이 발생하여 학습이 잘 안된다.

(4) Stride: Stride 는 Sliding Window 를 적용할 때 몇 칸을 움직일지 결정한다. 이 기능에서는 Stride 를 1 로한다.

(5) Padding: convolution layer 에서 3X3 Filter 계산 후에 나오는 Feature map 은 크기가 줄어들게 된다. 따라서 Input 이미지의 바깥쪽에 0 Padding 값을 넣어 Feature map 의 크기가 이미지의 크기와 동일하게 유지시켜 준다.

(6) Max Pooling: Feature map 의 크기를 결과적으로 줄여주어 Feature 를 추출해야 하는데 이 Pooling 방식을 이용하여 해결한다. 2X2 Max Pooling 을 사용하고 Stride 는 2 로한다. 4 개의 값 중에서 가장 큰 값만 추출한다. 이렇게 하여 특징을 추출하고 Feature Map 크기를 줄인다. Max 값을 추출하면 이미지내 객체가 평행이동 되더라도 객체 식별에 문제가 없다.

(7) Fully Connected Network: 모든 Input 이 다음 hidden state 를 결정하는데 이용된다. 우리 학습 구조의 마지막 부분에서 사용된다.

(8) SoftMax: 객체 분류에서 사용되는 함수로 결과값으로 Input 값이 각 객체일 확률을 얻어낼 수 있다. 이것 역시 우리 모델 마지막 부분에서 분류를 위해 사용된다.

Figure 9. VGG-16 모델

(9) 사용할 CNN 모델(VGG-16)

우리가 사용하는 모델에서 CNN 은 구조의 앞 쪽에 위치하게 되는데 CNN 의 모델의 마지막 레이어까지 통과하지 않고 체크 표시된 중간 Feature map 을 Output 으로 가지고 와서 사용한다. 체크 표시까지의 구조는 이렇다. 3X3 Convolution Filter 개수 64 를 2 번 통과하고 Max Pooling 이 과정을 두 번 반복하여 Feature Map 을 얻어 낸다.

3.2.2.2 Function Requirement 2.2

ID: FR5

Title: RPN Network (Region Proposal Network)

Input: Feature Map

Output: object/not object classify, Bounding Box regression

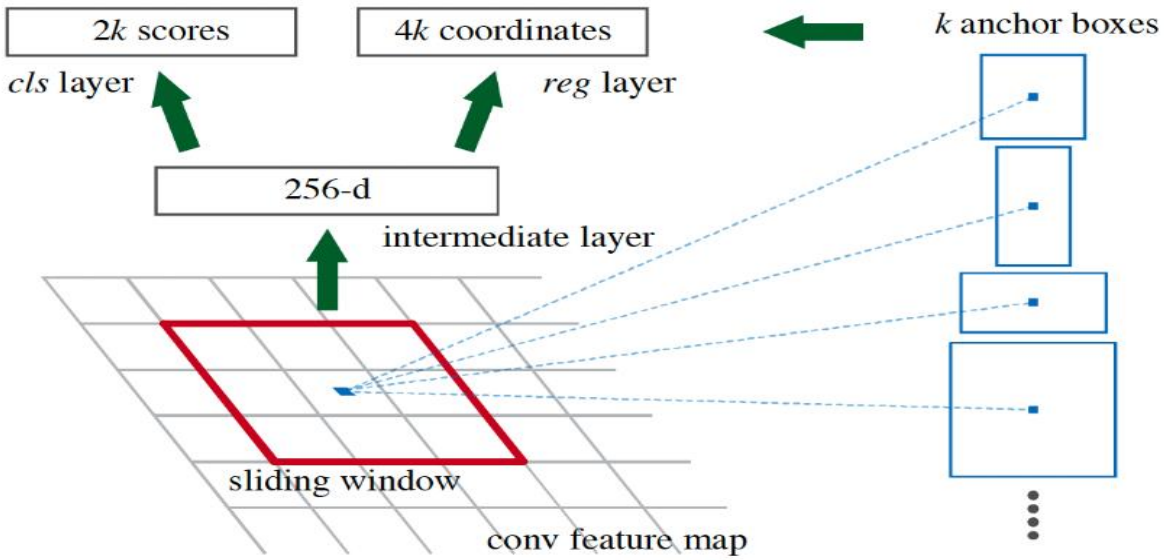

Figure 10. RPN Structure

DESC:

(1) RPN 구조

a) CNN 을 통해 나온 Feature Map 에 3*3 Convolution 연산을 256 개 또는 512 개의 채널 수만큼 수행한다. 그림 2 의 Intermediate Layer 에 해당한다. 이때 Padding 을 1 로 설정해 H*W 가 보존될 수 있도록 해준다. 수행 결과 H*W*256 또는 H*W*512 크기의 두 번째 Feature Map 을 얻는다.

b) 두 번째 Feature Map 을 입력 받아서 Classification(Object 인지 아닌지 분류하는 것)과 Bounding Box Regression 예측 값을 계산한다. 이때 Fully Convolution Network 의 특징을 갖는다.

c) Classification 을 수행하기 위해는 1*1 Convolution 을 2*9 (Object 인지 아닌지 나타내는 지표수*Anchor 개수)개의 채널 수만큼 수행해주며 결과로 H*W*18 크기의 Feature Map 을 얻는다. 이 값들을 Reshape 해준 다음 Softmax 를 적용하여 해당 Anchor 가 Object 일 확률을 얻는다.

d) Bounding box Regression 예측 값을 얻기 위한 1*1 Convolution 을 4*9 (Bounding box 좌표*Anchor 개수)개의 채널 수만큼 수행한다. 결과로 H*W*36 크기의 Feature Map 을 얻는다.

e) 앞에서 얻은 값들로 RoI 를 계산한다.

(2) Anchor Box: 위의 그림에서 Feature Map으로부터 Sliding Window를 할 때 Sliding Window 중심점에서 Anchor Box 영역을 나눈다. 이렇게 하여 객체가 있을 만한 곳의 영역을 먼저 정하여 네트워크를 계산한다. 각 슬라이딩 윈도우당 9개의 Anchor Box를 생성할 예정이다. 3가지 비율과 크기를 가진 Anchor Box를 이용한다. Input Feature Map 13X13크기를 가진다고 한다면 cls layer의 결과로 13X13X2K를 갖는다. 이 벡터에는 각 윈도우마다 K 개 Anchor Box 각각에 대하여 object 인지 아닌지 벡터를 가지고 있다. Reg layer 에서는 13X13X4K를 갖고 각 Anchor Box의 중심좌표와 높이 너비의 벡터를 갖는다.

Figure 11. IOU

(3) IOU (Intersection over Union): IOU의 계산 값은 교집합 영역 넓이/합집합 영역 넓이이다. 이 개념은 RPN 네트워크 학습 시 사용된다. Ground Truth 즉 실제 이미지 안에 있는 객체와 Anchor Box의 IOU를 계산하여 IOU threshold 값이 넘으면 그 Anchor box에는 객체가 있다고 판단하여 Positive(1) 값을 준다. 우리 네트워크에서는 두가지 기준으로 Anchor 라벨에 Positive/Negative를 나눈다. Ground-truth box와 가장 높은 IOU를 가지는 anchors 와 Ground-truth box와 IOU>0.7 이상인 anchors Positive class 값을 주고 IOU<0.3인 anchors에 대해서는 Negative Class를 부여한다.

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*).$$

Figure 12. Loss function

(4) Loss function: 위에 IOU로 얻어낸 Positive 값과 RPN 이 예측한 값의 차이와 Anchor box의 Bounding Box 와 실제 Bounding Box 차이의 합을 줄이는 방향으로 RPN은 학습된다. Pi가 예측 값이고 Pi*가 Ground truth label 값으로 손실함수로는 log function을 사용한다. Ti는 예측 Bounding Box이고 Ti* 가Ground truth Bounding Box이다. 손실함수는 smooth L1을 사용한다.

3.2.2.3 Function Requirement 2.3

ID: FR6

Title: Non-Maximum Suppression

Input: Anchor Boxes

Result: 중복 Anchor 삭제

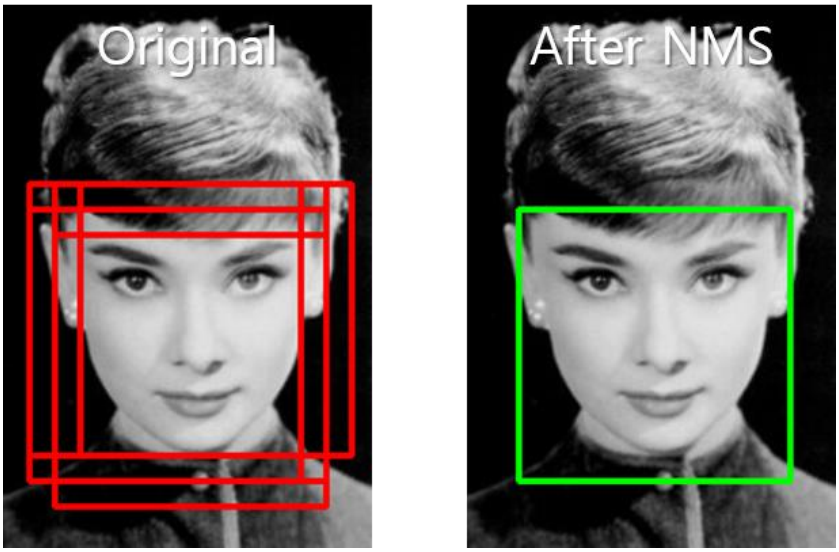

Figure 13. NMS

DESC: RPN으로부터 객체당 여러 개의 proposal을 얻게 되기 때문에 이 개수를 줄이기 위하여 이 기능을 구현해야 한다. 이 알고리즘은 IOU값 높은 순서로 proposal을 정렬시킨 후 가장 높은 proposal 과 다른 proposal의 IOU값이 특정 임계 값 이상이면 그 proposal을 삭제하는 방법이다. 이렇게 동일한 물체에 여러 개의 box가 쳐져 있는 경우 가장 score가 높은 box만 남기는 과정으로 학습속도를 높인다.

3.2.2.4 Function Requirement 2.4

ID: FR7

Title: RoI Pooling

Input: Feature Map, Region of Interest Coordinates

Output: Fixed Feature Map

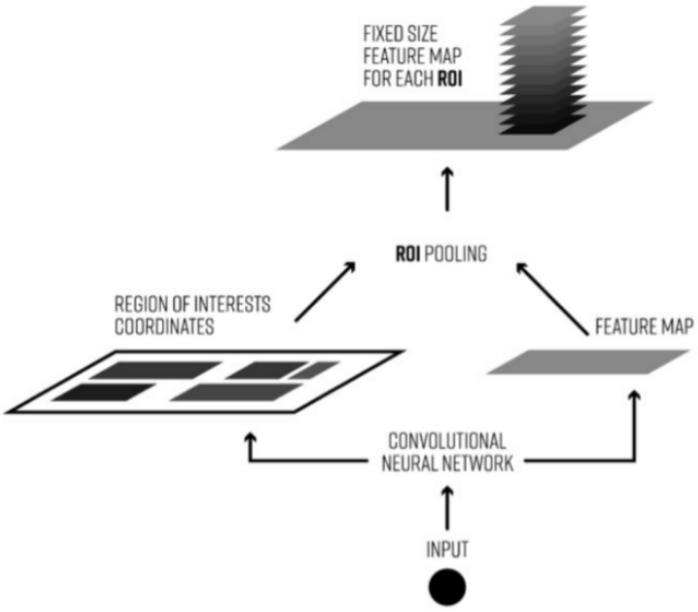

Figure 14. RoI Pooling 과정

DESC: RPN 과정을 지나면 서로 다른 크기의 영역이 나오게 된다. 이후 Fully Connected Layer에 입력으로 넣기 위해서는 같은 크기가 되어야 한다. 따라서 크기를 맞춰 주기 위해 RoI Pooling 과정을 거친다. RoI Pooling은 크기가 다른 Feature Map의 Region마다 Stride를 다르게 Max Pooling을 진행하여 결과값을 맞추는 방법이다. 결과로 고정된 길이의 Feature Vector를 만들어 준다.

3.2.2.5 Function Requirement 2.5

ID: FR8

Title: Detection (Classification & Bounding Box Regression)

Input: Fixed Feature Map

Output: Softmax, Bounding Box

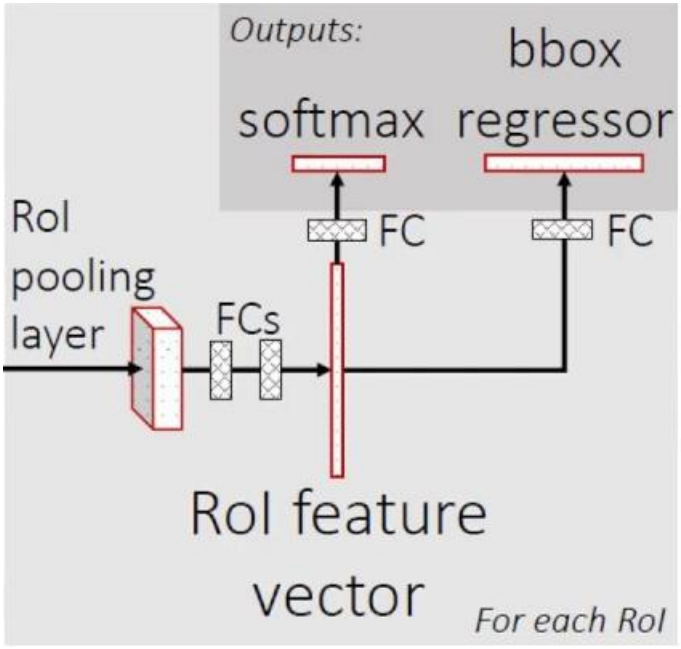

Figure 15. Detection Layer

DESC: ROI Pooling layer로부터 얻은 Fixed Feature Map을 Input으로 받는다. Fully connected layer을 거쳐서 객체 분류 classification과 bounding box regression벡터 값을 얻어낸다. Ground truth 값과Detection Layer 예측 값의 차이가 적어지는 방향으로 학습시킨다. Classification은 Softmax 함수를 이용하여 분류한다. Classification과 bounding box regression은 동시에 학습한다.

3.2.2.6 Functional Requirement 2.6

ID: FR9

Title: 객체 분류 및 탐지

Input: Classification Softmax, Bounding Box Vector

Output: 자동차, 구급차, 사람 객체 개수

DESC: Faster-RCNN 모델로부터 나온 Class벡터와 Bounding Box 벡터를 이용하여 사람, 차, 구급차 Class를 가지는 벡터들의 숫자를 센다. 이 외의 Class들은 배경으로 판단한다. Faster-RCNN으로부터 나온 벡터들의 대하여 위에서 설명한 NMS알고리즘

을 이용할 수 있다. 같은 Class를 가지면서 두개의 Region을 IOU 계산을 했을 때 값이 크다면 두개의 Region은 같은 객체라고 판단할 수 있고 낮은 class 점수를 갖는 객체를 삭제한다. 이렇게 하여 객체 수 계산시에 중복되는 객체를 제거한다.

3.2.3 신호 스케줄링

3.2.3.1 Functional Requirement 3.1

ID: FR10

TITLE: 기본 신호 스케줄링

DESC: 사거리에서 모든 방향에서 차량(또는 보행자)이 존재한다고 가정했을 때 기본적으로 동작하는 신호 스케줄링 매커니즘은 다음과 같다.

(1) 위에서 아래로 보는 것으로 기준을 삼을 때 왼쪽도로, 오른쪽도로, 아래쪽도로, 위쪽도로 순으로 신호를 준다.

(2) 도로신호등의 기본값은 30초, 보행자 신호의 기본값은 20초이다.

(3) 신호 스케줄링은 한 신호에서 다음 신호로 바뀌기 2초전, 황색신호로 들어갔을 때 각 도로의 차량 대수, 보행자 수를 계산하여 도로의 가중치로 환산하여 도로 신호등의 기본값인 30초를 기준으로 다음 신호가 다른 신호에 비해 상대적으로 가중치가 높다면 신호의 시간을 늘려주고, 가중치가 상대적으로 적다면 신호의 시간을 줄여준다. 여기서 신호의 최대 지속시간은 60초, 최소 지속시간은 20초로 한다.

(4) 가중치는 보행자 수와 차량 대수의 합으로 계산하지만, 보행자 수가 많은 경우에는 도로의 시간을 늘려줄 필요가 없으므로 도로 시간은 다음과 같이 구한다.

도로 시간 = 기본시간(30초) + (추가시간 * (차량 수 / (보행자 수+차량 수))

3.2.3.2 Functional Requirement 3.2

ID: FR11

TITLE: 스케줄링 예외 사항

Stimulus: 구급차 출현

DESC: 도로에 구급차가 나타났을 때 신호 스케줄링은 현재 신호를 잠시 멈추게 하고 구급차가 나타난 도로 쪽에 신호를 주어야 한다.

(1) Scenario 1: 현재 켜진 도로에 구급차가 있는 경우, 신호가 남은 시간이 10초 이하인 경우에는 다시 10초를 부여해주고, 10초보다 많이 남아있는 경우 그대로 유지한다.

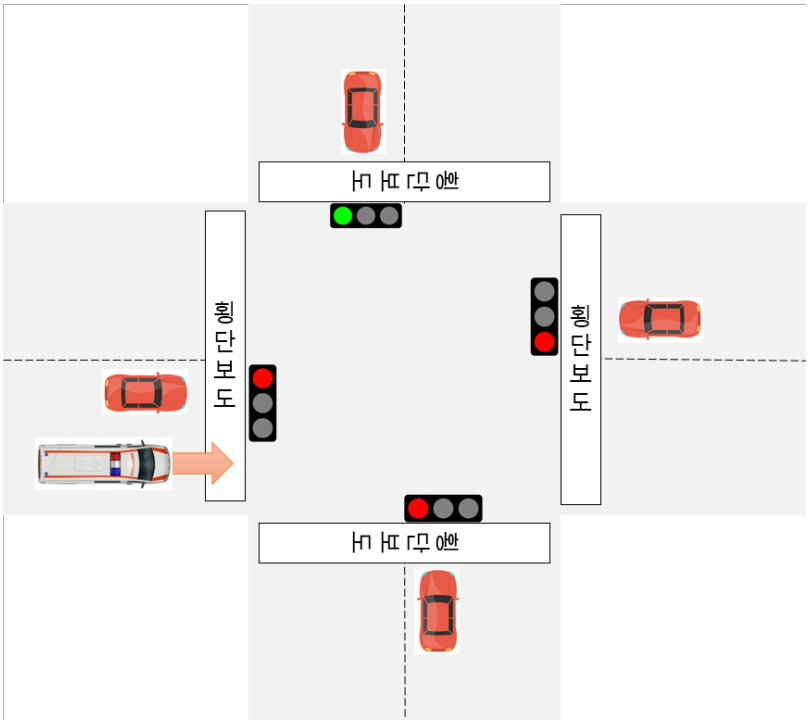

Figure 16. FR11_Scenario 2

(2) Scenario 2-1: 현재 켜진 도로가 아닌 경우, 현재 켜진 도로의 시간이 10초 이하인 경우라면 그대로 시간을 유지하고 다음 신호로 구급차가 있는 도로 쪽으로 신호를 10초 부여한다.

(3) Scenario 2-2: 현재 켜진 도로가 아닌 경우, 현재 켜진 도로의 시간이 10

초 이상이라면 도로의 시간을 10초로 줄이고, 다음 신호를 구급차가 있는 쪽으로 10초간 부여한다. 그리고 다시 원래 켜졌던 도로에 남은 시간 만큼의 시간을 다시 부여한다. 이때 남은 시간이 5초 이하라면 다음 신호로 넘어간다.

(4) Scenario 3: 구급차가 통과해야 하는 도로에 보행자 신호가 켜져 있을 경우에는 10초 후에 현재 켜진 도로신호를 적색으로 변경하고 보행자 신호가 끝날 때 까지 기다린 후 곧바로 구급차 쪽 도로신호를 부여한다..

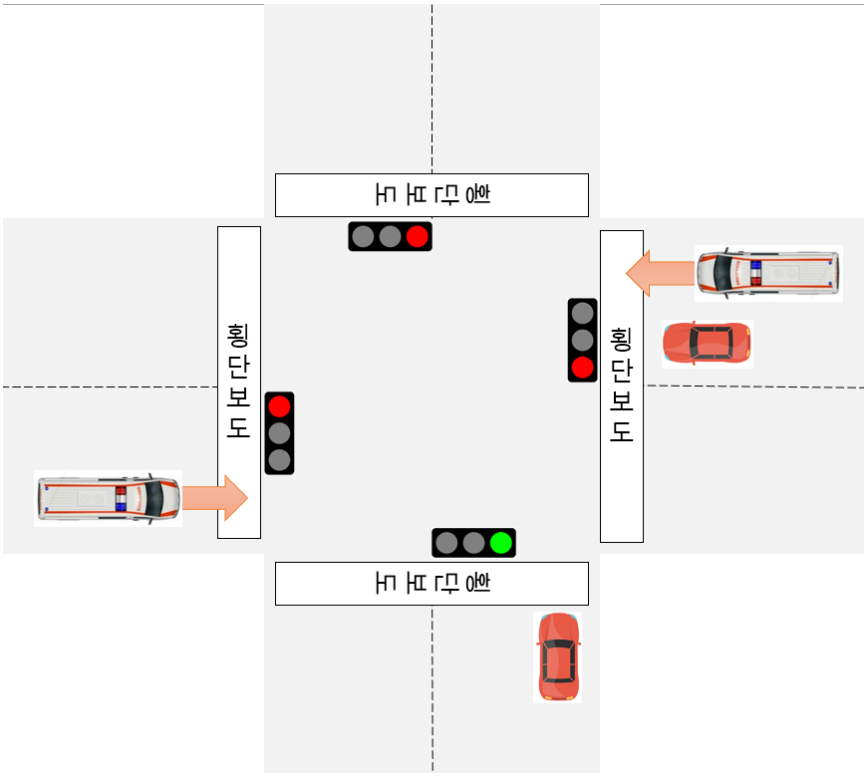

Figure 17. FR11_Scenario 4

(5) Scenario 4: 여러 도로에 구급차가 동시에 식별된 경우에는 큐에 저장해서 발견된 순서대로 신호를 변경해 준다. 큐가 비워지면 원래 신호체계로 복귀한다.

3.2.3.3 Functional Requirement 3.3

ID: FR12

Stimulus: 모든 도로에 차, 보행자가 없을 경우

DESC: 도로에 차나 보행자가 없을 경우 효율성을 높이기 위해서 도로들의 신호를 추가적으로 관리한다.

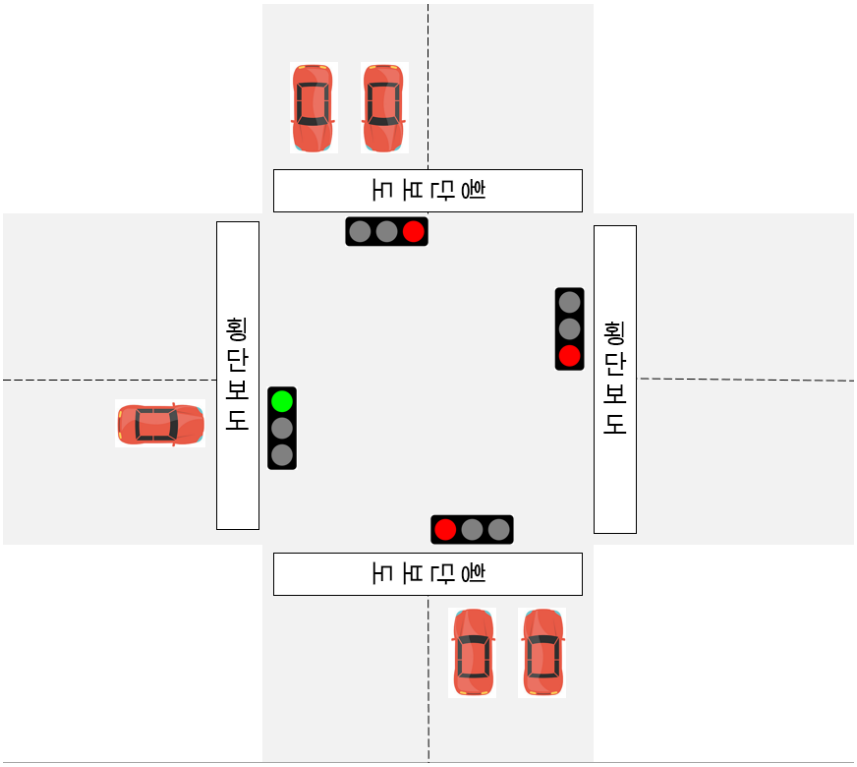

Figure 18. FR12_Scenario 1

Scenario 1: 다음 신호가 켜질 도로의 차량과 보행자가 없을 경우에는 해당 도로의 신호를 부여하지 않고 그 다음 도로의 신호로 넘어간다.

Scenario 2: 모든 도로에 차량과 보행자가 없을 경우에는 신호를 전부 적색신호로 유지하다가 새로운 차량이나 보행자가 들어오면 그 도로부터 한 cycle의 스케줄링을 진행한다.

3.3 performance requirement

(1) 딥러닝을 이용한 Object detect의 결과를 0.5초 이내로 얻을 수 있어야한다.

(2) 분석된 데이터를 이용한 신호 스케줄링이 0.5초 이내로 끝나야 한다.

(3) 한 사진에서 최소한 10대의 차량과 5명의 사람을 인식할 수 있다.

(4) 구급차 통과시 1초 안에 신호변경 요청을 반영한다.

(5) 딥러닝 모델 예측 정확도가 70% 이상이어야 한다.

3.4 design constraints

(1) Report format : iee830-1998 standards

(2) Data naming : Camel naming

3.5 software system attributes

3.5.1 Reliability (Required reliability of the software at time of delivery)

(1) 도로 상황 분석 시스템이 시뮬레이션 안에서 찍은 사진을 분석해서 객체를 인식할 수 있어야 한다.

(2) 신호 관리 시스템은 최소한 최소 유지시간이 지날 때마다 바뀌어야 한다.

(3) 신호등의 순서는 구급차의 예외 상황을 제외하고는 동일한 순서를 유지해야한다.

(4) 보행호는 해당하는 자동차 신호에만 동작할 수 있어야 한다.

3.5.2 Availability

(1) 서버와의 통신이 연결되어 있어야 한다.

(2) 서버로 사용할 Colab의 최대 세션 유지 시간이 12시간이다.

3.6 Other requirement