Date 2020-09-13

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
ResNet	CNN에서 네트워크 Layer를 깊어질 수록 생기는 vanishing Gradient 문제를 해결하기 위해 고안된 모델
RetinaNet	기존의 ResNet + FPN 구조에서 손실함수를 Focal Loss함수로 변경한 모델
FPN	Feature Pyramid Network, object detection을 하기 위해 Feature map을 다양한 형태로 rescale하는 방법
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

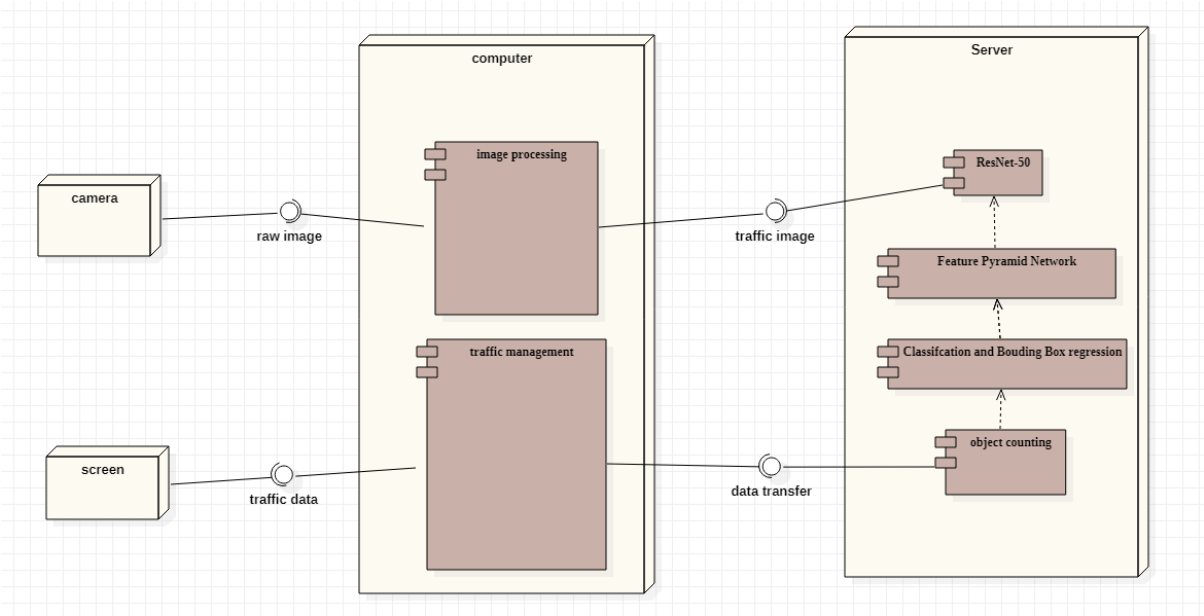
Focal Loss for Dense Object Detection - Tsung-Yi Lin Priya Goyal Ross Girshick Kaiming He Piotr Doll´ar (Facebook AI Research (FAIR))

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



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(ResNet&FPN)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 상황이기 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 되돌아와야 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

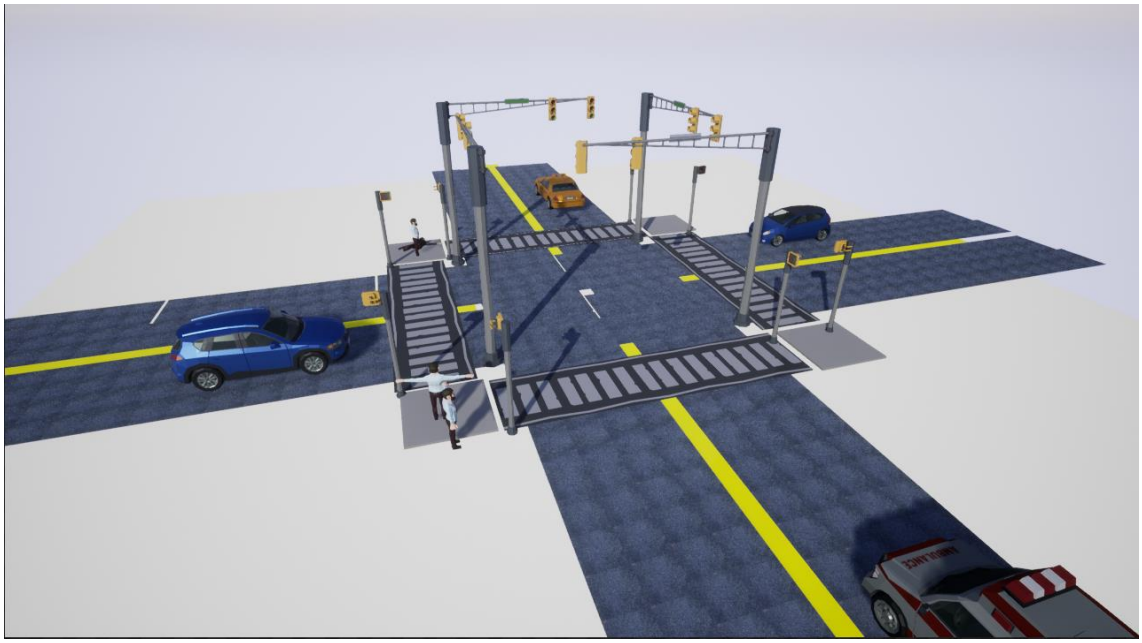


Figure 1. 세트장 모델링

(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 차량신호와 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 2. 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 3. 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 4. 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 RetinaNet 모델 설명

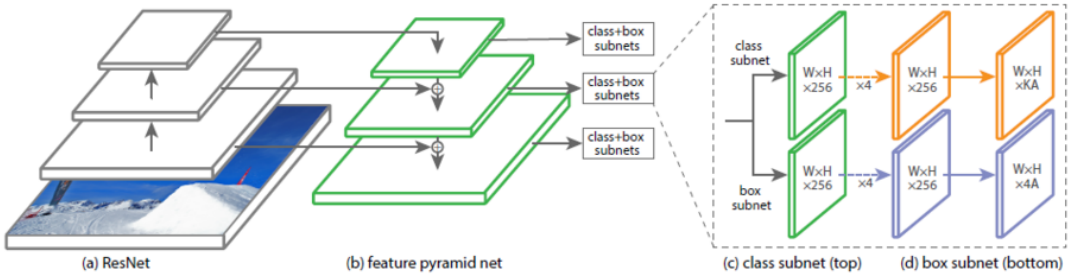


Figure 5. RetinaNet 모델 설명

DESC: RetinaNet 전체적인 구조

RetinaNet 은 하나로 통합 네트워크로 backbone network 와 2 개의 전용 subnetwork 로 구성된다. backbone 은 입력 이미지 전체 영역에 대해서 convolutional feature map 을 계산하는 역할이며, 누구나 사용할 수 있는 공개된 convolutional network 이다. 첫번째 subnet 은 backbone 의 output 으로 부터 convolutional 을 통해서 object classification 을 수행한다. 두 번째 subnet 은 backbone 의 output 으로부터 convolutional 을 통해서 경계 박스의 좌표 (앵커와 정답 간의 offset)를 구한다.

3.2.2.1 Function Requirement 2.1

ID: FR4

Title: Two type of Object Detection

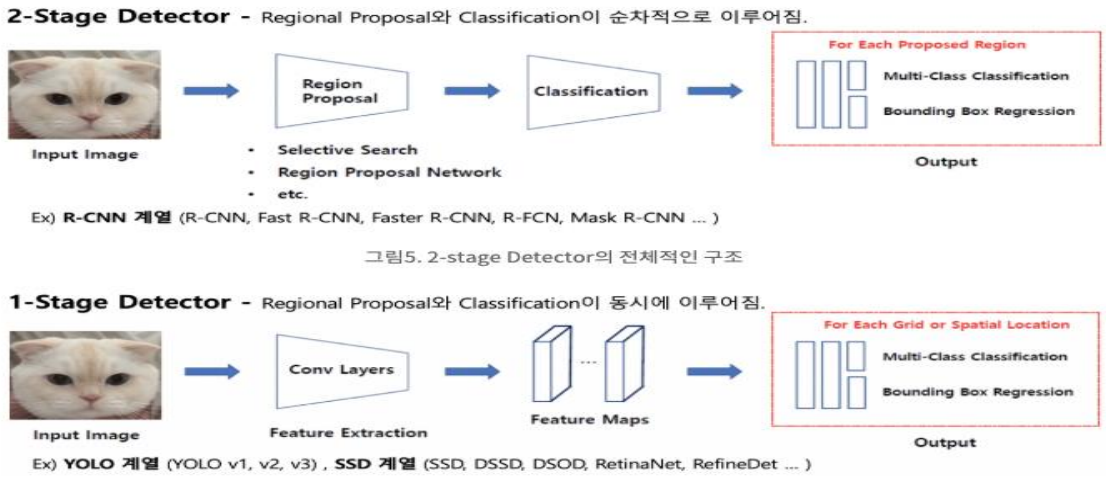


Figure 7. Tow type of Object Detection

DESC:

객체탐지는 물체를 식별하기 위한 Classification, 물체의 위치를 찾는 Localization 두 가지를 수행해야 하는데, 이때 1-Stage Detector 와 2-Stage Detector 로 나눌 수 있다. 1-Stage Detector 는 Region Proposal 과 Classification 을 동시에 해결하는 방식이고 2-Stage Detector 는 앞의 두 가지를 순차적으로 진행하는 방식이다. 2-Stage Detector 는 객체를 검출하는 정확도 측면에서는 굉장히 좋은 성능을 냈지만 속도 측면에서는 느리다는 단점이 있고, 1-Stage Detector 는 처리속도는 빠르지만 정확도가 낮은 편이었다. 1-Stage Detector 의 정확도 문제의 이유는 2-Stage Detector 계열 RPN 과정을 통해 Foreground 와 Background 를 어느 정도 구별해내지만 1-Stage Detector 의 경우에는 Feature Map 의 Grid 를 활용하기에 Background 영역이 상대적으로 많이 포함되어 데이터 클래스 불균형 문제가 생기기 때문이다. 이러한 문제는 기존의 손실함수 Cross-Entropy 를 변형시켜 해결한다.

3.2.2.2 Function Requirement 2.2

ID: FR5

Title: ResNet

DESC:

네트워크의 층(Layer)을 더 쌓으며 아주 깊은 네트워크를 구현하여 성능 향상을 이루고 있었다. 하지만 실제로는 어느 정도 이상 깊어진 네트워크는 오히려 Vanishing/Exploding Gradient 문제 때문에 성능이 더 떨어졌는데 이러한 현상을 Degradation Problem이라고 한다. 문제가 발생하는 이유는 Weight들의 분포가 균등하지 않고, Backpropagation이 제대로 이루어지지 않기 때문이다. ResNet은 이러한 Neural Network의 구조가 깊어질수록 정확도가 감소하는 문제를 해결하기 위해 제안되었다. 문제 해결을 위한 방법이 Residual Learning이다.

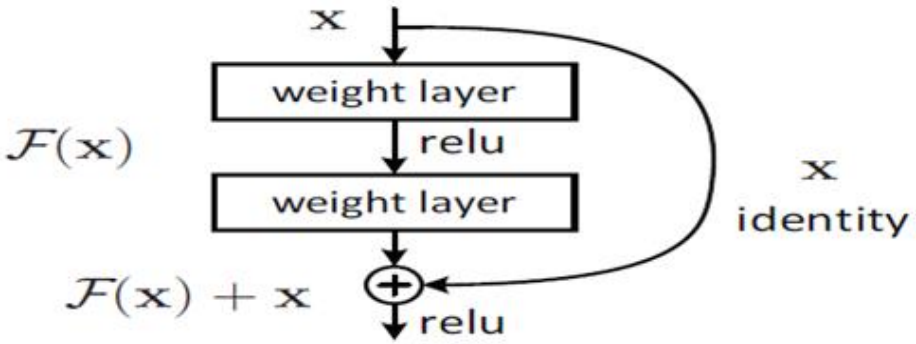


Figure 8. Residual learning: a building block.

기존의 Neural Network의 학습 목적이 입력(x)를 타겟값(y)으로 Mapping하는 함수 H(x)를 찾는 것이었다면 Neural Network는 H(x)-y를 최소화하는 방향으로 학습을 진행했다. ResNet에서는 관점을 바꿔 네트워크가 H(x)-x를 얻는 것으로 목표을 수정하였다. 입력과 출력의 잔차를F(x) = H(x)-x 라고 정의를 하고 네트워크는 이 F(x)를 찾는 것이다. 이렇게 잔차를 학습하는 것을 Residual Learning이라고 한다. 출력은 H(x)=F(x)+x가 되고 이렇게 네트워크의 입력과 출력이 더해진 것을 다음 레이어의 입력으로 사용하는 것을 Shortcut Connection(또는 Skip Connection)이라고 한다. Shortcut Connection을 적용한 네트워크를 학습할 때 역전파과정에서 x로 미분하면 적어도 1이상의 값이 나오기 때문에 네트워크가 깊더라도 이전보다 안정적으로 학습이 가능해진다.

3.2.2.3 Function Requirement 2.3

ID: FR6

Title: Feature Pyramid Network Backbone

DESC:

Feature Pyramid Network(FPN)을 RetinaNet의 backbone으로 채택했다. 간단하게 말하면, FPN은 일반 convolutional network를 top-down 방향으로 augment 한다.

Feature.6 (a)-(b) 와 같이 lateral connection(수평 연결)을 통해서 하나의 resolution 입력 이미지로부터 rich한 multi-scale feature pyramid를 효율적으로 구성한다.

객체 검출에서 객체 크기의 다양함은 아주 중요한 문제이다. 크기와 위치에 관계없이 객체를 하나의 클래스로 탐지해야 하기 때문이다. 이전에는 다양한 크기의 물체를 탐지하기 위해 이미지 자체의 크기를 Resize하면서 물체를 찾았다. 이런 작업은 메모리 및 시간 측면에서 비효율적이다. 이를 개선하기 위해 FPN이라는 방법이 나오게 되었다.

Figure 9. Feature Pyramid Network

먼저 신경망을 통과하면서 단계별로 Feature Map을 생성한다. 그리고 가장 상위 Layer에서부터 거꾸로 내려오면서 해상도와 채널 수를 모두 맞춰준 두 Feature를 Elementwise 덧셈을 수행하여 합쳐준 뒤, Object Detection을 진행한다. 이러한 방식을 통해서 상위 Layer의 추상화된 정보와 하위 Layer의 작은 물체들에 대한 정보를 동시에 살리면서 Object Detection을 할 수 있게 된다.

3.2.2.4 Function Requirement 2.4

ID: FR7

Title: Anchor

DESC:

RPN(Region Proposal Network)에서 사용하는 translation-invariant anchor box(이동 불변 앵커박스)를 사용했다. 앵커박스의 크기는 피라미드 레벨 P3~P7에서 32^2 ...

512^2의 크기를 갖는다. 각 피라미드 레벨에서 3개의 종횡비{1:2, 1:1, 2:1} 를 갖는 앵커박스를 사용했다. 모든 스케일에 대응(denser scale coverage) 하기 위해서 각 피라미드 레벨에서 원래의 3개의 앵커박스에 대해서 각각 3개가 추가되었다. {2^0, 2^1/3, 2^2/3} 이렇게 하는 것이 AP를 올리는데 도움이 된다. 각 레벨에서 총 A = 9 개의 앵커박스를 갖고 있으며 입력 이미지의 32 - 813 픽셀까지 담당할 수 있다. 각 앵커박스에는 K개의 one-hot vector classification target 와 4-vector의 box regression target 이 할당된다. K는 number of ojbect class이다. GT와의 intersection-over-union(IoU)가 0.5 이상인 앵커박스만 ground-truth object box 로 할당된다. IoU가 0.4 이하의 앵커박스는 background 로 할당된다. 앵커박스는 최대(오직) 1개의 object box 에만 할당된다. K label vector는 클래스에 해당하는 entry만 1, 나머지는 0으로 채운다. (one-hot) 만약 앵커박사가 아무 것에도 할당되지 않는다면, 즉 GT와의 IoU가 0.4~0.5 인 경우라면, 해당 앵커박스는 학습과정에서 무시된다. Box regression targets 은 각 앵커박스와 할당된 object box 와의 offset(거리)를 계산한다. 할당되지 않은 앵커박스들은 계산에서 빠진다.

3.2.2.5 Function Requirement 2.5

ID: FR8

Title: Classification Subnet

DESC:

classification subnet은 각 위치에서 A 앵커에의 K개의 object class 확률을 예측한다. 이 subnet은 각 FPN 레벨에 붙은 작은 FCN이다. subnet의 네트워크 구조는 모든 피라미드 레벨에서 전부 동일하다. 디자인도 매우 단순하다.

Feature.6 (c) 와 같이 주어진 피라미드 레벨에서 C 채널의 feature map을 받는다. 3x3 컨볼루션을 256 개의 필터로 컨볼루션하고 ReLU 활성함수를 적용한다. 이것을 4회 반복한다. 마지막으로 KA(K*A) 개의 필터로 컨볼루션한 후 매 공간 위치에서 binary prediction을 위해서 Sigmoid 활성함수를 적용한다. RPN 과는 대조적으로 우

리의 object classification subnet은 레이어가 깊고 3x3 컨볼루션만 사용한다. 모든 피라미드 레벨에서 파라미터를 공유한다.

3.2.2.6 Function Requirement 2.6

ID: FR9

Title: Box Regression Subnet

DESC:

각 피라미드 레벨에서 object classification subnet과는 병렬적으로 또다른 작은 FCN 이 붙어 있다. 이것은 Feature.6 3(d) 와 같이 각 앵커박스에서 근처에서 ground-truth object 가 있는 경우 둘 간의 offset을 예측하는 것이 목표이다. 각각의 위치에서의 A 개의 앵커박스들은 자신과 GT간의 상대적인 좌표 offset 4개를 예측한다. object classification subnet 과 box regression subnet은 모두 공통의 구조를 가지지만, box regression subnet은 파라미터는 공유하지 않는다.

3.2.2.7 Function Requirement 2.7

ID: FR10

Title: Focal loss DESC:

DESC:

One-stage Network(YOLO, SSD 등)의 Dense Object Detection이 일반적으로 Two-stage Network(R-CNN 계열)에 비해 속도는 빠르지만 성능은 낮다. 이는 극단적인 클래스간 불균형에서 기인된다. 따라서 클래스 분류에 일반적으로 사용되는 cross entropy loss 함수를 조금 수정한 Focal loss가 제안된다. 이는 Easy sample에 대해서는 작은 가중치를 부여하는 반면 Hard sample에는 큰 가중치를 부여해서 학습을 어려운 예제에 집중시키는 것이다.

Focal loss는 One-stage object detection에서 object와 background의 클래스간 unbalance가 극도로 심한 상황(ex. 1:1000)을 해결하기 위해 제안되었다.

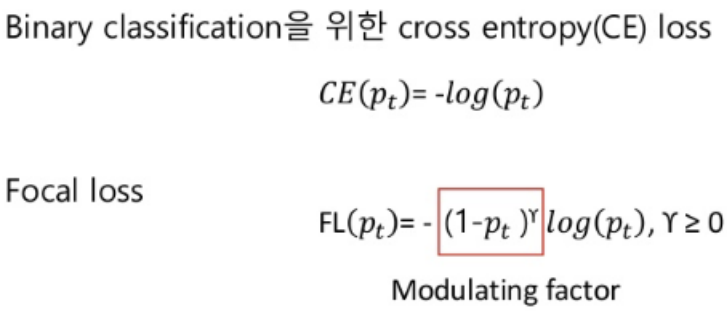


Figure 10. cross entropy loss와 Focal loss

Focusing parameter ϒ는 일반적으로 0~5 사이의 값(2 works best)

(1) 만약 example을 잘못 분류하고, Pt 가 작은 값인 경우

$(1-pt)^{\gamma}$~1 -> loss함수에 영향을 주지 않는다

(2) 만약 example을 잘 분류하고 Pt ~1 인 경우

$(1-pt)^{\gamma}$~0 -> loss for well classified examples is down-weight

Modulating factor는 easy example들의 loss에 대한 영향력을 줄인다.

예를 들어 ϒ=2 경우, Pt = 0.9 로 예측되었던 example의 원래 CE로스에 비해서 FL에서는 CE의 Pt=~0.968 일 때 수준의 작은 loss를 받게 되므로 상대적으로 1000x 작아 진다. 이는 잘못 분류된 examples의 중요도를 상대적으로 높이는 역할을 한다.

3.2.2.8 Function Requirement 2.8

ID: FR11

Title: 각 객체 counting 및 전송

DESC:

RetinaNet으로 진행한 Object detection의 결과로 나온 Class 정보로 각 도로 별 객체들의 개수를 세서 저장하는 Dictionary를 만들어 신호 스케줄링에 전달한다.

3.2.3 신호 스케줄링

3.2.3.1 Functional Requirement 3.1

ID: FR12

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 40초, 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 40초를 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: FR13

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.

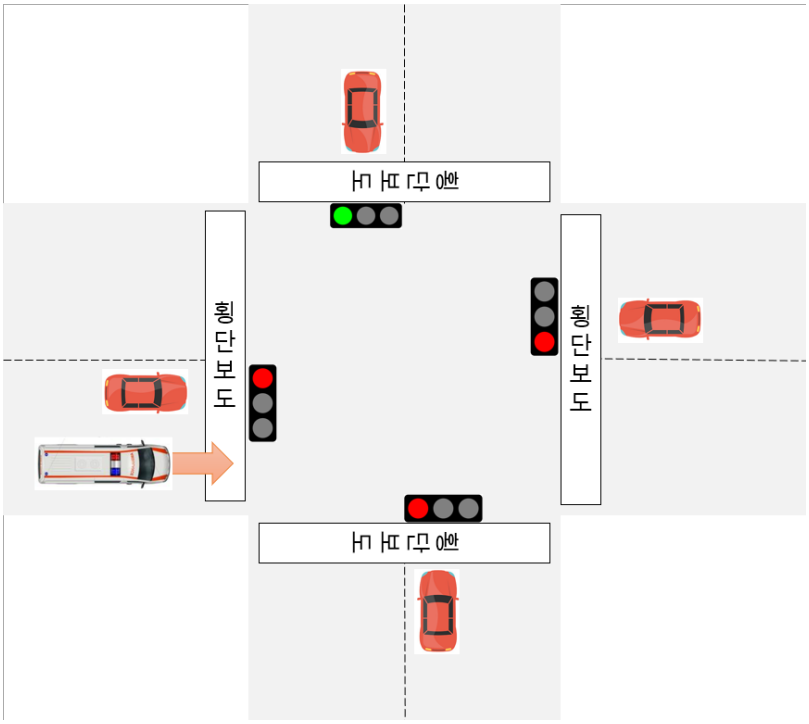


Figure 11. FR11_Scenario 2

(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 10초 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 부여 한다.

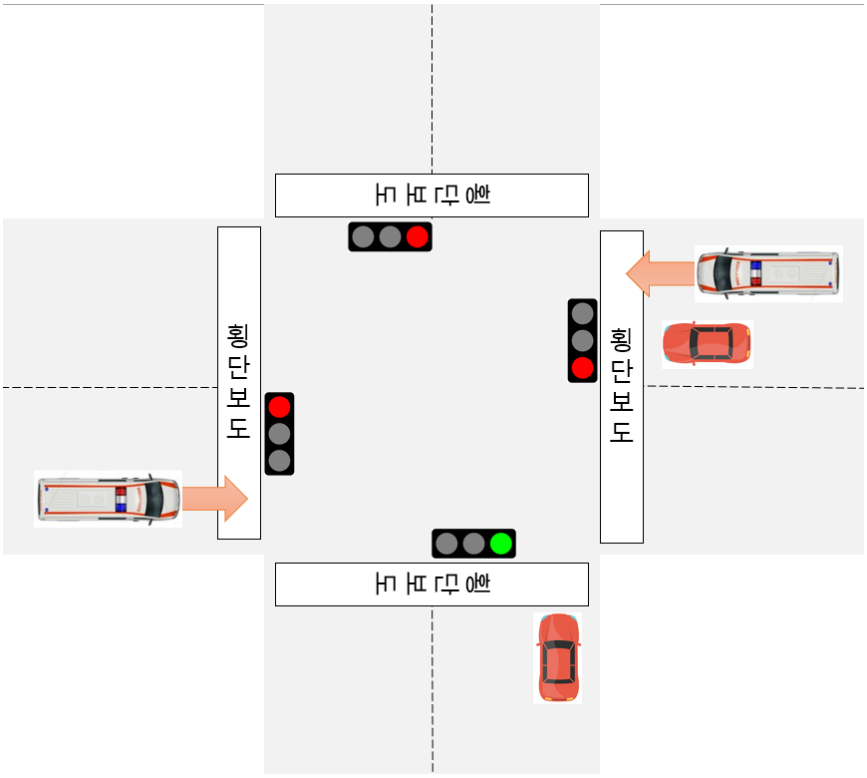


Figure 12. FR11_Scenario 4

(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: FR14

Stimulus: 모든 도로에 차, 보행자가 없을 경우

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

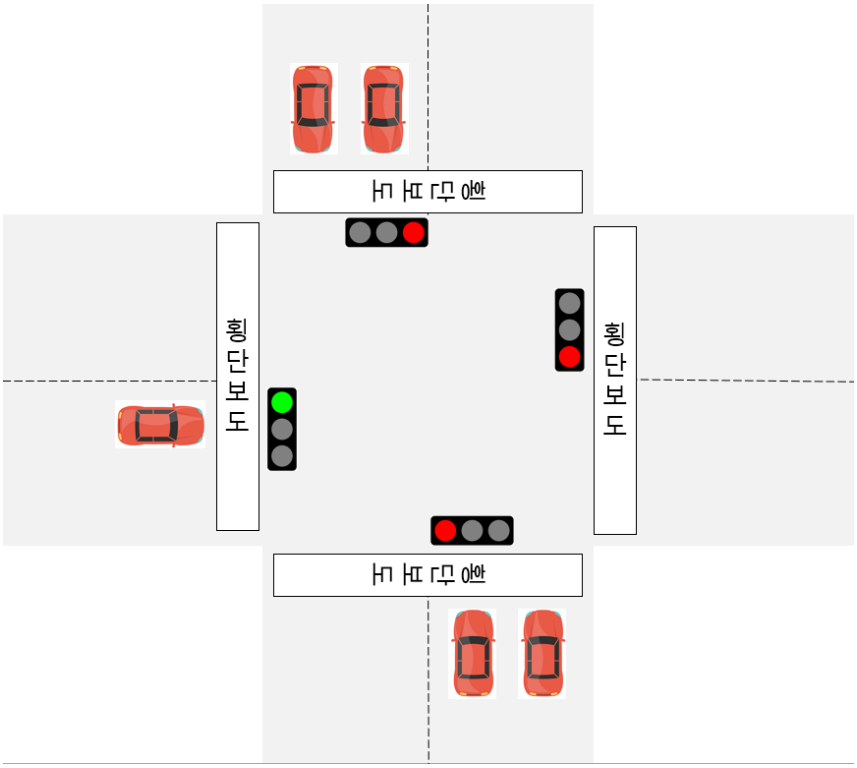


Figure 13. FR12_Scenario 1

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