기술경영학 석사 학위논문

엣지 컴퓨팅 환경에서 객체탐지 기법을 활용한 도로 균열 탐지 시스템 개발

2020년 2월

부경대학교 기술경영전문대학원

기술경영학과

하 종 우

기술경영학 석사 학위논문

엣지 컴퓨팅 환경에서 객체탐지 기법을 활용한 도로 균열 탐지 시스템 개발

지 도 교 수 김 민 수

이 논문을 기술경영학 석사 학위논문으로 제출함.

2020년 2월

부경대학교 기술경영전문대학원

기술경영학과

하 종 우

하종우의 기술경영학 석사 학위논문을 인준함.

2020년 2월 21일

위 원 장 공학박사 이 지 환 (인)

위 원 공학박사 옥 영 석 (인)

위 원 공학박사 천 동 필 (인)

목 차

Ⅰ. 서 론 ---- 1
1. 연구의 배경과 목적 ---- 1
2. 연구의 방법과 구성 ---- 7

Ⅱ. 이론적 배경 및 선행연구 ---- 9
1. 클라우드 컴퓨팅 환경과 엣지 컴퓨팅 환경 ---- 9
2. 딥러닝과 객체탐지 ---- 12
3. 도로 균열 탐지 ---- 32

Ⅲ. 파일럿 시스템의 균열 탐지 설계 ---- 40
1. 제안된 DNN(Deep Neural Network)의 구조 ---- 40
2. DNN의 학습 및 테스트 결과 ---- 41

Ⅵ. 도로 균열 탐지를 위한 모바일 시스템 ---- 54
1. 파일럿 시스템 구조 ---- 54
2. 구축 환경 및 기능 구현 ---- 55
3. 테스트 운용 결과 및 활용 방안 ---- 57

Ⅴ. 결론 ---- 60
1. 연구 요약 ---- 60
2. 연구의 한계점과 향후 연구 방향 ---- 61

참고 문헌 ---- 63
1. 국내 문헌 ---- 63
2. 해외 문헌 ---- 63

감사의 글 ---- 70

표 목 차

<표 Ⅲ-1> 실제 값과 예측 값에 다른 분류 ---------- 49
<표 Ⅲ-2> 학습에 이용된 객체의 수 ---------- 50
<표 Ⅲ-3> 모델의 성능 평가 ---------- 51
<표 Ⅲ-4> 보정한 데이터셋에서의 균열 유형별 객체의 수 ---------- 51
<표 Ⅲ-5> 보정한 데이터셋으로 학습한 모델의 성능 평가 ---------- 52

그 림 목 차

<그림 Ⅰ-1> 도로 균열 탐지와 유지보수 과정 ---------- 2
<그림 Ⅰ-2> 연도별 도로 유지보수 예산 (단위, 억 원) ---------- 3
<그림 Ⅰ-3> 분류 모델과 객체 탐지 모델의 예측 결과 예시 ---------- 5
<그림 Ⅱ-1> 클라우드 컴퓨팅 환경의 기본 구조 ---------- 10
<그림 Ⅱ-2> 엣지 컴퓨팅 환경의 기본 구조 ---------- 12
<그림 Ⅱ-3> 인공지능, 머신러닝, 딥러닝의 포함 관계 ---------- 14
<그림 Ⅱ-4> 퍼셉트론(Perceptron)의 기본 구조(Rosenblatt, F., 1958) --- 15
<그림 Ⅱ-5> 심층 신경망(DNN)의 기본 구조 ---------- 16
<그림 Ⅱ-6> 합성곱 연산의 수행 과정 ---------- 17
<그림 Ⅱ-7> 합성곱 신경망을 현실에 적용한 LeNet ---------- 17
<그림 Ⅱ-8> 객체탐지 분야의 연구 동향 ---------- 18
<그림 Ⅱ-9> R-CNN의 모델 구조 ---------- 19
<그림 Ⅱ-10> Fast R-CNN 구조 ---------- 20
<그림 Ⅱ-11> SPP(Spatial Pyramid Pooling) 계층의 구조 ---------- 22
<그림 Ⅱ-12> RPN의 수행 과정 ---------- 23
<그림 Ⅱ-13> YOLO의 기본 과정 ---------- 24
<그림 Ⅱ-14> YOLO의 객체탐지 과정 ---------- 25
<그림 Ⅱ-15> YOLO의 신경망 모델 구조 ---------- 26
<그림 Ⅱ-16> YOLO와 Fast R-CNN의 성능 차트 ---------- 27
<그림 Ⅱ-17> SSD의 다양한 크기를 가지는 특징맵 ---------- 28
<그림 Ⅱ-18> SSD의 신경망 모델 구조 ---------- 28
<그림 Ⅱ-19> MobileNet 신경망 구조 ---------- 30
<그림 Ⅱ-20> Depthwise Seperable Convolution의 구조 ---------- 30
<그림 Ⅱ-21> MobileNetV2의 신경망 구조 ---------- 32
<그림 Ⅱ-22> 균열 탐지 결과(Sun, L. et al., 2015) ---------- 34
<그림 Ⅱ-23> Crack Saliency Map을 통한 균열 탐지 ---------- 35
<그림 Ⅱ-24> CrackTree의 균열 탐지 과정 ---------- 36
<그림 Ⅱ-25> 제안된 신경망 구조(Zhang, L. et al., 2016) ---------- 37
<그림 Ⅱ-26> 제안한 신경망 구조(Wang, K. C. et al., 2017) ---------- 38

<그림 Ⅱ-27> 균열 탐지 결과(Maeda, H. et al., 2018) ---------- 39
<그림 Ⅲ-1> MobiletNetV2를 활용한 SSD 모델의 구조 ---------- 40
<그림 Ⅲ-2> Tensorflow Object Detection API 사용과정 ---------- 42
<그림 Ⅲ-3> 10000*3739 크기의 학습 이미지 샘플 ---------- 43
<그림 Ⅲ-4> Annotation 파일 샘플 ---------- 44
<그림 Ⅲ-5> 학습 데이터 CSV 파일 ---------- 45
<그림 Ⅲ-6> SSD 모델의 신경망 구조 ---------- 46
<그림 Ⅲ-7> 예측 위치와 실제 정답간의 IoU 값 ---------- 48
<그림 Ⅲ-8> Precision-Recall 곡선의 예시 ---------- 48
<그림 Ⅲ-9> 학습한 모델의 테스트 결과 ---------- 50
<그림 Ⅲ-10> 수정된 데이터로 학습한 모델의 테스트 결과 ---------- 52
<그림 Ⅳ-1> 엣지 컴퓨팅 환경 구조의 파일럿 시스템 ---------- 54
<그림 Ⅳ-2> Jetson Nano 보드의 형태 ---------- 55
<그림 Ⅳ-3> 클라우드 웹 서버 구축 환경 ---------- 56
<그림 Ⅳ-4> 특정 샘플 이미지에 대한 시스템 운용 과정 ---------- 58

A Development of Road Crack Detection System Using Object-Detection Methodology for Edge Computing Environment

Jong Woo Ha

Graduate School of Management of Technology
Pukyong National University

Abstract

In large cities with heavy traffic and many roads to manage, it takes months to even years to detect road cracks and repair them. One of main reasons for this prolonged maintenance job can be attributed to manual and batch processing of road crack detection.

Manual and visual inspections are repeated from the initial crack detection task to the onsite re-diagnosis which cause significant delay in the maintenance process.

Currently, some local governments are attempting to automate and accelerate those manual road crack detection and maintenance systems using artificial intelligence technologies and edge computing devices.

In the field environment where the network connection is unstable, road crack diagnosis should be done without relying on the central server.

During this process, the difference between the road scanner image and onsite acquired image should be properly handled by the edge device.

In addition to this, the computational capacity required for training and operating artificial neural networks for crack detection needs to be properly distributed among the cloud server and edge devices.

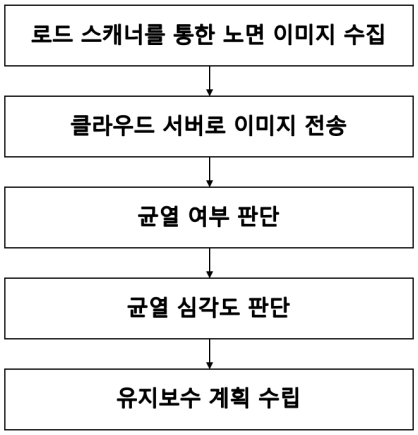
In this paper, to handle those problems of road management system, a method for developing mobile Internet environment over edge device that detects road cracks and supports repair works is studied.

Ⅰ. 서 론

1. 연구의 배경과 목적

국토교통부의 도로포장 유지보수 실무편람에 따르면 자동차도로의 포장관리를 위해 도로의 상태를 측정하고, 그 결과를 체계적으로 분석하여 보수 우선순위 및 최적 보수공법을 결정하도록 도움을 줄 수 있는 도로포장관리체계(PMS: Pavement Management System)를 운영 중이라고 한다. 그러나 현재까지도 도로의 포장상태 파악과 보수작업의 진행은 육안에 의존하고 있는 경우가 많으며, 그 과정에서 효율적인 예산사용과 최적의 포장상태 유지에 상당한 어려움을 겪고 있는 실정이다(국토교통부, 2013).

일반적으로 많은 지자체에서 현재 수행되고 있는 도로의 균열 탐지와 유지보수 작업은 <그림 Ⅰ-1>과 같은 과정으로 진행된다. 먼저 노면 촬영을 위해 설계된 로드 스캐너 카메라가 장치된 차량이 일정한 간격으로 운행되면서 도로를 촬영하게 되고, 이 결과 수많은 도로 이미지들이 수집되어 중앙 서버로 전송된다. 수집된 도로 이미지들에 대해서 많은 작업자들이 육안검사를 통해 균열의 여부와 위치 및 심각도를 판단하게 된다. 해당 판단의 결과를 종합하여 개별 도로의 유지보수 작업에 대한 결정이 내려지게 되는데, 이 과정에 상당한 시간이 소요되어 파손된 도로가 장시간 위험한 상태로 방치되고 있는 상황이다.



<그림 Ⅰ-1> 도로 균열 탐지와 유지보수 과정

특히 도로 균열의 탐지 과정에서 수십만 장에 해당하는 이미지를 작업자들이 육안으로 진단하는 업무는 가장 많은 비용과 처리시간을 유발시키고 있는데, 작업의 일관성에 대한 높은 신뢰도를 기대하기 어렵게 만들고 있으며, 무엇보다도 보수까지의 긴 소요시간 동안 균열의 상태와 양상이 지속적으로 심화되기 때문에, 보수를 결정한 시점에서의 진단 내용과 실제 보수를 위해 현장을 방문했을 때의 상황이 크게 달라지는 문제점이 발생한다.

이러한 현상은 통행량이 많고, 관리해야 할 대상 도로들이 많은 대도시에서 특히 심각하게 발생하고 있는데, 해당 기간 동안 운전자들이 도로상의 위험에 지속해서 노출된다는 점에서 시급한 대책 마련이 요구된다. 이런 문제점을 인식한 일부 지자체에서는 최신의 인공지능 기술과 모바일 인터넷 환경을 적극적으로 활용하여 많은 자원과 시간을 요구하는 균열의 탐지 및 유지보수 작업을 개선하고자 하는 시도를 하고 있다.

위와 같이 도로 포장 상태 불량의 위험성과 포장관리의 중요성이 강조됨에 따라서 국토교통부에서는 2014년부터 매년 도로 유지보수 예산을 확대

해오고 있다(국토교통부, 2019). <그림 Ⅰ-2>는 연도별 도로 유지보수에 책정된 국토교통부 예산을 나타낸다. 늘어가는 도로 유지보수 예산에 있어서 이를 효율적으로 분배하고 세부계획을 수립하기 위해서 도로의 현재 상태에 대한 판단이 우선 효율적인 프로세스를 통해 진행되어야만 한다. 따라서 도로 포장 결함의 주 요소인 균열에 대해 이를 저비용과 낮은 소요시간으로 적시에 탐지하고 유지보수 방안을 제시할 수 있는 시스템의 필요성이 강조된다.

<그림 Ⅰ-2> 연도별 도로 유지보수 예산 (단위, 억 원)

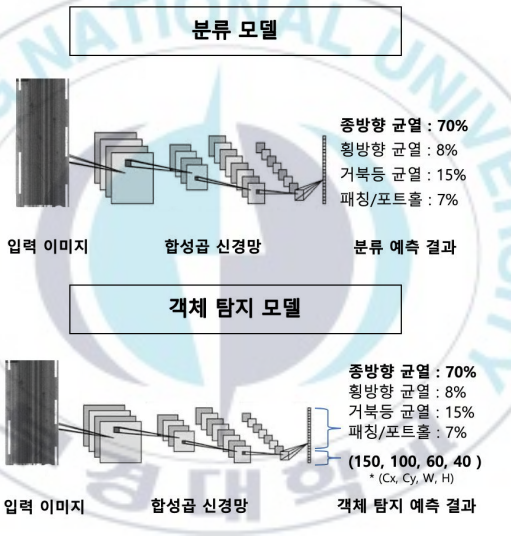
그러나 아직까지 균열의 탐지를 위해 사용되고 있는 기존의 기술과 방법론들이 고성능과 장시간의 학습을 해야 하는 중앙 서버 장비를 중심으로 진행되어 왔으며, 정확한 유지보수 작업의 계획을 위해 매우 중요한 균열 부위의 파악과 심각도의 결정에 대한 연구가 초보적 수준에 그치고 있는 실정이다. 특히, 보수 현장에서의 제한된 계산 용량과 불안정한 네트워크 상황을 고려하여, 이를 극복할 수 있도록 엣지 단말을 활용하는 방법에 대한 연구가 매우 부족한 실정이다. 현실적인 유지보수 계획의 수립을 위해

서는 균열 부위의 한정과 이를 기초로 한 균열 심각도의 결정이 선행되어야 하지만, 기존의 많은 연구들이 도로 이미지로부터 균열의 유형을 분류하는 작업에 치우쳐 있어서, 실질적인 유지보수 계획의 수립에는 이르고 있지 못한 상태이다. 본 연구에서는 제한된 계산능력을 지닌 엣지 장비와 모바일 인터넷 환경을 결합하여, 도로 균열의 이미지 분류에서 더 나아가 균열 부위의 한정과 심각도 결정을 위한 객체탐지(Object Detection) 기술의 심도 있는 활용 방안을 다루고자 한다. 특히, 현장에서 지속적으로 축적되는 신규 데이터를 반영하여 서버상의 신경망 모델이 재학습되고, 재학습된 모델이 현장의 엣지 단말에 전파되어 진단에 활용되기까지의 과정을 체계화하여 구현함으로써 제안된 시스템이 데이터의 축적과 모델의 갱신을 통해 유기적으로 동작할 수 있는 구조를 제안하고자 한다.

도로 균열탐지에 관한 초기 연구들은 대상 이미지의 알고리즘적인 분석과 구조적인 해석을 중심으로 한 이미지 처리를 통해 이루어져 왔으나 최근에는 컴퓨터 비전과 머신러닝 그리고 이미지상의 패턴 인식에 특화된 딥러닝(Deep Learning) 기술의 발달과 더불어, 학습을 통한 분류 및 탐지로 연구의 초점이 이동하고 있다. 그러나 딥러닝 기법을 사용한 기존의 도로 균열탐지에 관한 연구들은 주로 분류(Classification) 모델을 중심으로 하고 있으므로, 분석 대상 이미지에 균열이 존재하는지와 존재한다면 어떤 유형의 균열인지에 대한 판단 문제를 다루어 왔다. 그러나 도로의 유지보수를 위해서는 정확히 도로상의 어느 부위에 어떤 유형의 균열이 발생하였고, 어느 정도의 심각성을 지니는가를 파악해야 하므로, 현장에서 실질적인 활용도를 지니기 위해서는 분류모델만으로는 한계가 있으며, 객체탐지까지 진전된 모델의 연구가 필요하다.

객체탐지 기법을 사용하는 딥러닝 모델의 경우 해당 이미지에 대한 균열 유형의 분류와 함께 균열의 위치(Localization)에 대한 문제를 함께 해결하

는 모델로써, 일단 균열의 위치와 유형이 결정될 수 있다면 후속 처리를 통해 균열의 심각도를 더욱 상세히 분석해 낼 수 있게 된다. 이러한 이유로 객체탐지 기반의 모델을 균열탐지 자동화에 활용할 수 있다면, 도로포장 관리와 유지보수 작업의 효과적인 수행에 큰 도움이 될 것이다. 따라서 본 논문에서는 이러한 객체탐지 모델의 활용에 초점을 맞추어 현장의 유지보수 작업에 활용될 수 있도록 운용환경을 설계하고 구축하는 방법을 다루고자 한다. <그림 Ⅰ-3>는 딥러닝을 활용한 분류 모델과 객체 탐지 모델의 최종 예측 결과 예시를 나타낸다.



<그림 Ⅰ-3> 분류 모델과 객체 탐지 모델의 예측 결과 예시

그러나 객체탐지 모델은 분류 모델보다 훨씬 긴 학습시간과 연산량을 필요로 하므로 현장에서의 활용을 위해서는 중앙 서버와 현장의 엣지 단말과의 역할과 연계 방법에 대한 종합적인 고려가 필요하다. 현재 많은 기기와 서비스들이 중앙에서 데이터를 집중하여 처리하는 클라우드 컴퓨팅 형태로 사용되고 있다. 도로균열 탐지를 위해 곳곳의 도로를 운행 중인 차량에서

촬영된 로드 스캐너의 이미지가 중앙의 서버에 전송되어 축적되고 있으며, 이러한 데이터를 기초로 한 신경망 모델의 학습도 중앙의 서버에서 이루어지고 있다. 여기에 보수작업을 위해 개별 현장에서부터 취득된 이미지의 분석 요구까지 서버로 집중된다면 서버의 처리 부담은 더욱 가중될 것이다. 중앙에 집중되는 처리는 현장에서 취득된 이미지들을 서버에 올리기 위한 통신비용과 서버의 분석 결과를 수신하기까지의 대기 시간을 유발하는 문제가 있다. 반면 현장에 배포되는 엣지 단말과 모바일 장비들이 과거와는 달리 상당한 처리 성능을 지니고 있다는 점에서 클라우드 서버에 집중된 연산을 제한적으로 나누어 가짐으로써 불필요한 데이터 트래픽과 대기 지연을 축소할 수 있는 시스템의 구축이 가능할 것으로 예상된다.

위와 같이, 현재의 클라우드 컴퓨팅 기반 환경에서는 중앙에 집중된 서비스 처리 요구가 사용자 및 연결 단말이 증가함에 따라 처리 지연 및 보안 문제와 같은 한계로 드러나고 있으며, 이를 극복하기 위한 처방으로 더 많은 클라우드 자원을 할당해야 하는 상황을 초래하고 있다. 반면, 급성장한 엣지 단말의 처리 성능을 적극적으로 활용하여 현장에서 일부 데이터의 정제(cleaning)와 분석을 처리하여 중앙 서버의 부하를 줄이고 처리 속도를 향상시킬 수 있는 엣지 컴퓨팅(Edge Computing) 기술이 활용된다면 현재의 제한된 IT 투자 환경 내에서도 충분히 요구되는 성능의 시스템을 구축할 수 있을 것으로 예상된다. 따라서 본 연구에서는 현장에서 취득된 도로 이미지를 엣지 단말 상에서 딥러닝 모델을 통해 즉각적으로 분석하여 도로 균열을 탐지함으로서, 유지보수 계획의 수립을 보조할 수 있도록 엣지 컴퓨팅 환경에서의 도로 균열 탐지 시스템을 구축하고자 한다.

본 연구에서는 다양한 실험을 통해 객체탐지 기술을 기초로 한 도로 균열탐지 모델의 구현과 학습을 수행하고, 엣지 단말의 처리 성능을 고려한 현장 진단 모델을 개발하여 모바일 인터넷 환경에서 중앙 서버의 대규모

학습과 엣지 단말의 진단 탐지가 유기적으로 연계될 수 있는 시스템의 개발에 관한 연구를 진행하였다.

2. 연구의 방법과 구성

본 연구는 크게 도로 균열 탐지를 위한 딥러닝 객체탐지 모델의 구현과 학습, 엣지 컴퓨팅 환경 구축을 위한 엣지 단말에서의 모델 사용, 그리고 단말과 서버와의 통신을 통한 신규 데이터의 축적과 서버 모델의 재학습 및 신규 모델의 엣지 단말로의 재배포를 위한 모바일 인터넷 환경의 구축으로 나뉜다.

먼저 중앙 서버의 객체탐지 모델 학습을 위해, 2015년도 한 해에 걸쳐 수집된 57만 여 장의 서울시 및 경기도 지역의 자동차 도로 표면 이미지가 사용되었다. 객체탐지 모델의 정의와 학습을 위해 Google의 딥러닝 라이브러리인 Tensorflow를 사용하였다. Tensorflow는 1.13 버전을 사용하였으며 Python은 3.5 버전을 사용하여 개발을 진행하였다. 객체탐지를 위한 모델은 MobileNetV2를 기반으로 하는 SSD로 선정하였으며, Tensorflow Object Detection API를 통해 해당 모델을 학습하였다.

모델 학습이 끝난 후 학습된 모델을 엣지 단말로 이식하게 되는데, 본 연구에서는 NVIDIA 사의 소형 AI 컴퓨터 젯슨 나노(Jetson Nano)를 시스템의 엣지 단말로 선정하여 사용하였다. 젯슨 나노는 우분투(Ubuntu) 18.04 LTS 버전의 OS를 설치하여 학습된 모델을 실행할 수 있도록 구성되었다.

단말인 젯슨 나노에서 데이터 연산 및 처리가 완료되면 해당 결과를 서버로 전송하게 된다. 서버의 구축은 PHP 웹 프레임워크인 Laravel 5를 통해 진행하였다. 위의 과정을 유연하게 연결하여 하나의 시스템으로 구성함

으로써 도로 현장에서의 실시간 균열탐지가 가능할 것이다.

본 논문은 다음과 같이 구성된다. 제 2장에서는 본 논문과 관련된 선행 연구들에 대하여 알아보고, 제 3장에서는 균열 탐지를 위한 딥러닝 모델의 학습에 대해 분석한다. 제 4장에서는 서버의 학습 모델을 엣지 단말의 진단 모델로 배포하고, 엣지 단말의 신규 데이터와 분석 결과를 서버와 공유하는 전체 과정을 모바일 인터넷 환경 상에서 단일 시스템으로 구성한 결과를 제시한다. 제 5장에서는 본 논문에서 도출한 결과에 대해 토론하고 앞으로의 연구 방향과 개선할 점에 대해 기술한다.

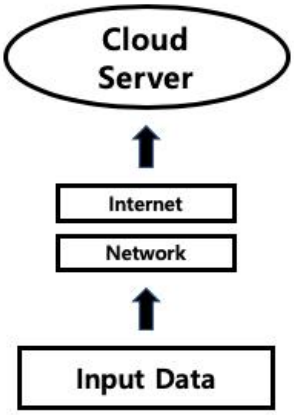
Ⅱ. 이론적 배경 및 선행연구

본 연구와 관련된 선행연구들을 몇 가지 주제로 묶어 정리하였다. 먼저 클라우드 컴퓨팅 환경과 본 연구에서 구현하는 엣지 컴퓨팅 환경의 정의와 장단점들에 대해 정리하였다. 그 다음 도로 균열 탐지와 관련된 선행연구들을 정리하고, 딥러닝의 기본 배경과 본 연구의 중요 기술인 객체탐지 기술의 선행연구들에 대해 정리하였다.

1. 클라우드 컴퓨팅 환경과 엣지 컴퓨팅 환경

클라우드 컴퓨팅(Cloud Computing) 환경은 중앙에서 데이터를 집중해서 처리하는 구조로써 원격서버를 통해 사용자의 데이터를 전달받아 주어진 처리가 이루어지는 환경이다. 사용자는 개인이 가진 단말기를 통해 주로 입력 혹은 출력 작업만 이루어지고, 데이터의 분석과 처리, 저장, 관리 등의 작업은 클라우드라고 부르는 원격서버에서 이루어지는 것이다. 클라우드 컴퓨팅 환경에서 업무나 작업을 처리함으로써 개인 서버에 대한 유지 및 보수에 대한 비용이 줄어들고 개인 서버에 대한 구축이 필요 없어지는 경우도 존재한다. 클라우드 컴퓨팅은 대표적으로 3가지의 서비스 유형 IaaS(Infra-structure as a Service), Paas(Platform as a Service) 그리고 SaaS(Software as a Service)로 구분된다(Sareen, P., 2013).

<그림 Ⅱ-1>은 클라우드 컴퓨팅 환경의 기본 설계도이다. 클라우드 컴퓨팅 환경에서는 발생하는 모든 데이터가 네트워크와 인터넷을 통해 중앙 집중 처리 장치인 클라우드 서버로 전송되어 데이터의 분석과 처리가 일어난다.



<그림 Ⅱ-1> 클라우드 컴퓨팅 환경의 기본 구조

반면 엣지 컴퓨팅은 네트워크 가장자리에서 클라우드 서비스를 대신하여 데이터의 다운스트림, 업스트림을 수행할 수 있는 기술을 의미한다. 여기서 '엣지'란 데이터 소스와 클라우드 데이터 센터 간의 경로를 따라가는 모든 컴퓨팅 및 네트워크 리소스로 정의할 수 있다(Shi, W. et al., 2016).

엣지 컴퓨팅(Edge Computing) 환경은 클라우드 컴퓨팅 환경과는 반대로 원격 중앙 서버가 아닌 사용자에게 가까이 있는 스마트폰 등의 단말을 통해 데이터의 분석과 처리가 이루어지는 환경을 말한다. 엣지 컴퓨팅은 정보 처리, 콘텐츠 수집 및 전달이 해당 정보의 출처, 보관 장소, 소비자에 인접한 곳에서 처리되는 컴퓨팅 토폴로지(Topology)이다. 엣지 컴퓨팅에서는 클라우드 컴퓨팅 환경에서 발생할 수 있는 처리 지연 시간을 줄이기 위해 네트워크 트래픽과 프로세싱을 로컬 단말에서 처리하고, 엣지 단말의 기능을 활용한다. 여기서 엣지 단말이란 사람들의 가까이에 존재하는 엔드포인트 단말을 지칭한다(Gartner, Inc., 2018).

개인이 소지하는 기기가 점점 늘어남에 따라서 데이터 량이 급증하게 되고 이 모든 데이터를 클라우드 컴퓨팅 환경에서 처리하는 것은 적절하지

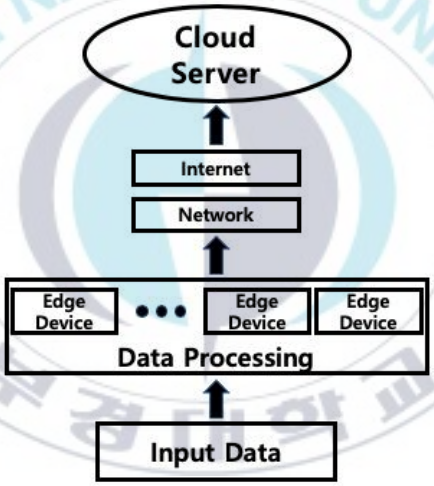
않다는 의견이 나오고 있으며, 이에 따라 엣지 컴퓨팅 환경의 구축에 대한 관심이 늘어나고 있다(Hao, Z et al., 2018). 또한 엣지 컴퓨팅 환경은 사물인터넷과 같은 기기들의 증가로 데이터의 트래픽이 과다하게 증가하여 클라우드 컴퓨팅 과정에서 문제가 생기는 것들을 해결할 수 있다(Stergiou, C et al., 2018). 미국의 IT 연구 및 자문 회사인 가트너 주식회사(Gartner, Inc.)에서 매년 발표하는 'Top 10 Strategic Technology Trends'에서는 비교적 최근에 발표된 2018년, 2019년, 2020년 10대 유망 기술에서 엣지 컴퓨팅이 매년 언급되고 있다(Gartner, Inc., 2017, Gartner, Inc., 2018, Gartner, Inc., 2019). 가까운 미래에는 이러한 엣지 컴퓨팅에 의해 사물인터넷을 포함한 데이터 처리 작업이 주도될 것이며, 필요에 의해 처리 작업은 클라우드 서버가 아닌 엣지 단말에서 이루어져 사용자 가까이에서 이루어질 것이다.

엣지 단말의 기능과 성능이 머신러닝, 딥러닝 모델을 실행가능할 만큼 강력해졌고, 각종 입력 센서들의 발달과 함께 정보를 클라우드 서버로 전송하여 처리해야할 필요가 줄어들었다. 엣지 컴퓨팅 환경을 통한 인공지능 모델 구축을 통해 클라우드 컴퓨팅 환경과 비교하여 보다 실시간에 가깝게 입력되는 정보에 반응할 수 있다. 또한 클라우드 서버와 정보를 주고받는 과정은 항상 보안 위협에 노출될 수 있다. 중요한 정보 처리 과정을 엣지 단말에서 실행함으로써 이러한 보안 위협을 최소화할 수 있다.

엣지 컴퓨팅은 크게 지역 근접성, 분산 효과, 모바일 지원의 3가지 특징을 가진다. 지역 근접성은 엣지 단말이 클라우드 서버와 사용자들 사이에 존재하면서 단말이 데이터 처리능력을 가지고 네트워크와 저장공간 등을 제공하기 때문이다(Hu, P et al., 2017). 클라우드 컴퓨팅 환경에서는 많은 사용자의 유입으로 인해 많은 데이터가 발생하고 이를 처리하는데 소요되는 시간이 길어지는 문제가 발생할 수 있지만 엣지 컴퓨팅 환경의 분산 효

과를 통해 이러한 점을 해결할 수 있다(Yu, W et al., 2018). 모바일 지원 특성의 경우에는 각각의 엣지 단말들이 네트워크로 연결될 수 있어 사용자가 이동 중에도 정보의 처리가 가능하게 할 수 있는 것이다.

<그림 Ⅱ-2는> 엣지 컴퓨팅 환경의 구조로서 데이터 입력과 데이터 처리를 위한 엣지 단말 그리고 처리된 결과를 전송하는 클라우드 서버로 이루어져 있다. 엣지 컴퓨팅 환경의 구조에서도 확인할 수 있듯이 엣지 컴퓨팅과 클라우드 컴퓨팅은 일부 상충되는 의미의 패러다임을 가지고 있지만 서로 공존하며 데이터 분석 및 처리에 대해 최적화할 수 있는 시스템이다(Shi, W et al., 2016).



<그림 Ⅱ-2> 엣지 컴퓨팅 환경의 기본 구조

2. 딥러닝과 객체탐지

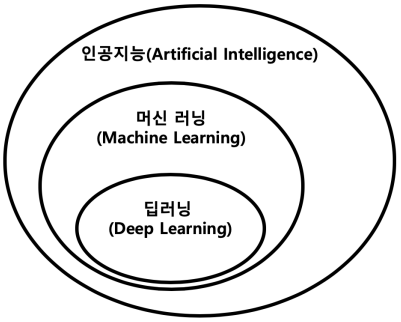
가. 딥러닝(Deep Learning)

딥러닝(Deep Learning)은 인공지능 분야에서도 특히 머신러닝 분야의

일부로서 현재 많은 연구가 진행되고 있다. 먼저 인공지능은 인간과 비슷한 수준의 능력을 컴퓨터로 구현하는 기술이다. 인공지능은 “인간처럼 생각하고 행동하며 이성적으로 생각하고 행동할 수 있는 시스템”으로 정의할 수 있다(S. J. Russell et al., 2003). 인공지능에 대한 연구는 머신러닝(Machine Learning)과 더 세부적으로 인공신경망에 대한 연구로 이어져오면서 현재는 4차 산업혁명의 중심 분야가 되었다.

머신러닝은 인공지능이라는 큰 분야의 일부분으로 머신러닝은 그 의미 그대로 기계학습을 의미하며 기계나 장치가 특정 알고리즘을 이용하여 기존 축적된 데이터를 통해 스스로 모델을 학습하여 새로운 데이터에 대해 예측을 하거나 분석을 할 수 있도록 하는 것을 의미한다. 이러한 과정을 거치면서 사람이 하나하나 세부적으로 해결할 수 없는 일들을 해결할 수 있고 사람보다 더 수준 높은 일을 처리할 수도 있다. 사람과 같은 혹은 사람보다 더 뛰어날 수 있다는 점에서 머신러닝에 대한 활발한 연구가 진행되어왔고 현재도 많은 연구가 진행되고 있다. 머신러닝 분야의 대표적인 모델로서는 딥러닝을 포함하여 회귀분석과 많은 분류모델들이 존재한다. 분류모델은 K-근접 이웃 알고리즘(K-Nearest Neighbors; KNN), 의사결정나무(Decision Tree), 서포트 벡터 머신(Support Vector Machine) 등이 있다.

인공지능, 머신러닝 그리고 딥러닝의 포함 관계는 <그림 Ⅱ-3>과 같다. 인공지능은 딥러닝 그리고 머신러닝을 포함한 가장 넓은 범위의 개념으로서 머신러닝을 제외하고도 많은 하위 분야들을 포함한다.



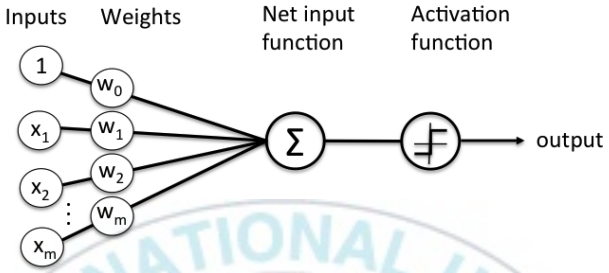
<그림 Ⅱ-3> 인공지능, 머신러닝, 딥러닝의 포함 관계

딥러닝은 인간의 두뇌를 컴퓨터로 모델링한 인공신경망(Artificial Neural Networks; ANNs)(LeCun et al., 1989)을 이용한 머신러닝의 한 분야이다. D. O. Hebb.(1949)에서는 생물학적인 신경망의 구조에서 신호가 발생하면서 나타나는 학습 효과를 인공신경망에 적용할 수 있음을 보여주면서 가중치(Weight)라는 개념을 설명하였다.

딥러닝은 다수의 인공신경망을 컴퓨터로 구현하여 심층 신경망(Deep Neural Network; DNN)을 모델링하고 해당 모델에 데이터를 입력해줌으로써 모델을 원하는 방향으로 학습해 나가는 것을 의미한다. 심층 신경망은 다양한 비선형 변환 기술의 조합을 통해서 추상화를 시도하려고 하는 머신러닝 기술이다(Bengio, Y. et al., 2013). 심층 신경망에서 특징의 추출이 수행되며 학습에 사용되는 데이터의 양이 많을수록 우수한 성능을 보일 수 있다(L. Deng et al., 2013).

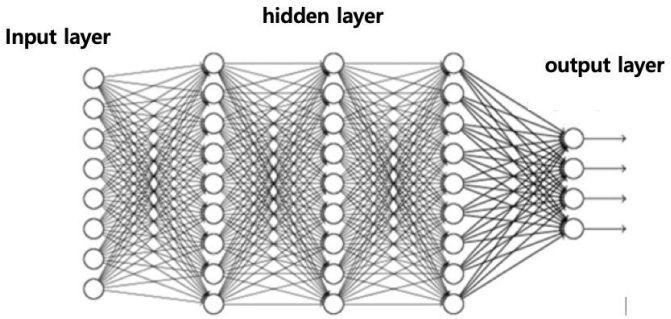
이러한 심층 신경망은 인간 신경계의 뉴런과 유사한 기능을 하도록 설계된 퍼셉트론(Perceptron)의 연속으로 이루어져있으며 <그림 Ⅱ-4>는 이 퍼셉트론의 기본 구조를 나타낸다. 퍼셉트론은 신호를 받아들여 가중치(Weight)를 곱하고 편향치(Bias)를 더한 값을 활성화 함수(Activation

Function)를 적용하여 값을 출력하는 방식을 가지고 있다. 퍼셉트론의 요소인 가중치와 편향이 곧 딥러닝 모델의 성능을 결정하게 되며, 딥러닝 모델의 학습은 곧 이 가중치와 편향의 값을 주어진 데이터에 적합하도록 수정해나가는 과정이라 할 수 있다.



<그림 Ⅱ-4> 퍼셉트론(Perceptron)의 기본 구조(Rosenblatt, F., 1958)

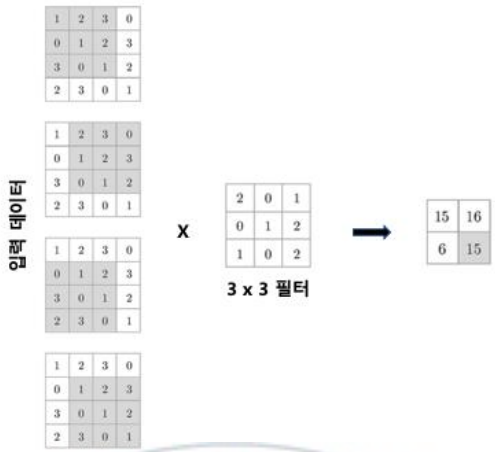
이러한 퍼셉트론을 다수의 층으로 연결하여 구성한 것이 인공신경망이라 할 수 있으며 구성하고 있는 층의 수가 많아짐에 따라 이를 심층 신경망이라 한다. 심층 신경망은 이러한 퍼셉트론을 연속적으로 설계하여 입력층(Input Layer)과 출력층(Output Layer) 그리고 그 사이에 다수개의 은닉층(Hidden Layer)을 두어 이루어진 모델이며 딥러닝 모델 중 가장 기본적인 구조라고 할 수 있다(J. Schmidhuber., 2015). <그림 Ⅱ-5>은 심층 신경망의 기본 구조를 나타낸다.



<그림 Ⅱ-5> 심층 신경망(DNN)의 기본 구조

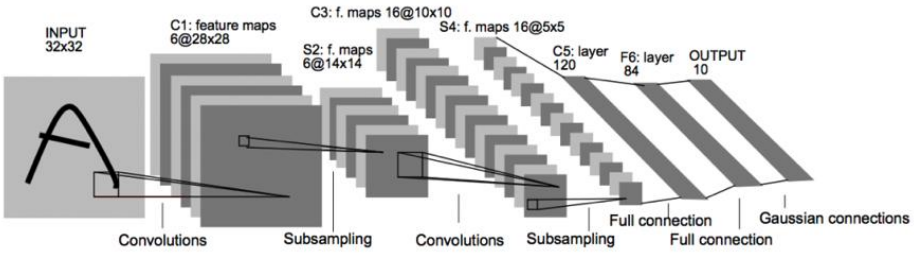
본 연구와 같이 이미지와 영상에 대한 분석과 처리를 위한 딥러닝 모델에서는 주로 합성곱 신경망(Convolution Neural Network; CNN)을 사용하게 된다. 영상, 이미지의 경우 일반적으로 3차원 행렬형태로 이루어져 있으며 각 이미지 픽셀에서 RGB와 같이 3개의 값을 가지게 되는데 이를 채널이라고 하며 RGB 이미지인 경우 3개의 채널을 가지고 있다고 할 수 있다. 합성곱 신경망은 이러한 3차원 형태의 데이터에서 특징 추출이 이루어질 때 효율적으로 수행하기 위해 등장하게 되었다.

합성곱 신경망은 일반적으로 입력 이미지의 형태와 같은 3차원의 인공신경망을 가진다. 예를 들어 입력 이미지의 형태가 224*224*3일 경우 모델의 입력층은 이와 같은 크기의 형태를 가지게 되는 것이다. 그리고 필터의 사이즈에 따라 합성곱을 수행해 나가면서 이미지의 특징을 추출해 나가고 모든 층에서의 연산이 완료되면 특징 지도(Feature Map)를 얻을 수 있다. <그림 Ⅱ-6>은 합성곱 연산의 방식을 나타낸다. 합성곱 신경망은 하나 이상의 합성곱 층과 완전 연결된 신경망 계층으로 구성되어있다.



<그림 Ⅱ-6> 합성곱 연산의 수행 과정

인공신경망의 이미지 및 영상 데이터에 특화된 형태인 합성곱 신경망을 이용하면 마찬가지로 가중치와 편향을 수정하는 과정을 거치면서 신경망의 학습을 진행해 나간다. LeCun, Y. et al.(1998)에서는 합성곱 신경망의 시초가 되는 <그림 Ⅱ-7>와 같은 신경망 구조를 제안하였으며 이는 실제로 우체국의 우편번호 인식 시스템에 적용되어 합성곱 신경망의 이미지 분석 성능을 보여주었다.

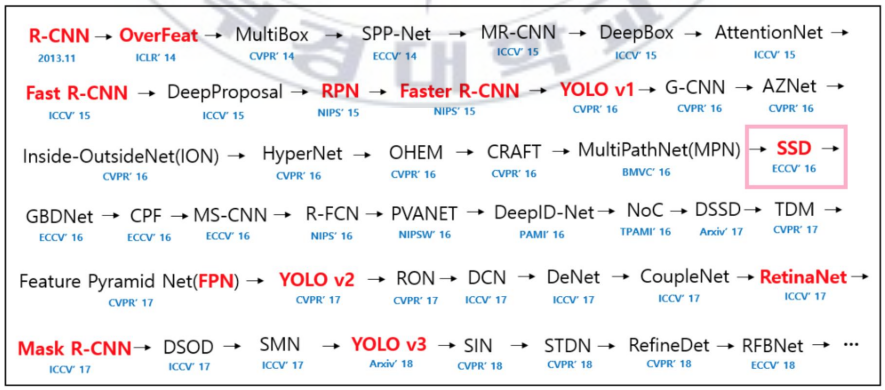


<그림 Ⅱ-7> 합성곱 신경망을 현실에 적용한 LeNet

본 연구에서는 위와 같이 이미지 처리 및 분석에 특화되어 있는 신경망인 합성곱 신경망을 활용한 객체 탐지 기반의 딥러닝 모델을 학습하여 연구를 진행하였다.

나. 객체탐지(Object Detection)

객체탐지(Object Detection) 분야는 이미지 속 여러 물체에 대해서 해당 물체가 이미지 속 어느 곳에 위치하고 어느 물체에 해당하는지에 대한 정보를 모두 추론해 내야하는 딥러닝 기반의 모델을 이용하는 분야이다. 이 객체탐지는 어떤 물체인지 추론해야하는 분류(Classification)문제와 Bounding Box라고 불리는 박스를 통해 해당 물체의 위치(Localization)까지 밝혀내야 하는 문제를 동시에 해결해야하는 분야이다. 현재 자율주행자동차와 같이 물체의 실시간 인식에 관한 연구들이 활발해지면서 객체탐지에 대한 연구 또한 활발한 상태이다. <그림 Ⅱ-8>는 객체탐지 연구의 동향을 나타낸다. 많은 연구들 중 사용빈도가 높고 객체탐지 분야의 중점이 되었던 연구들에 대해 정리해 보았다.

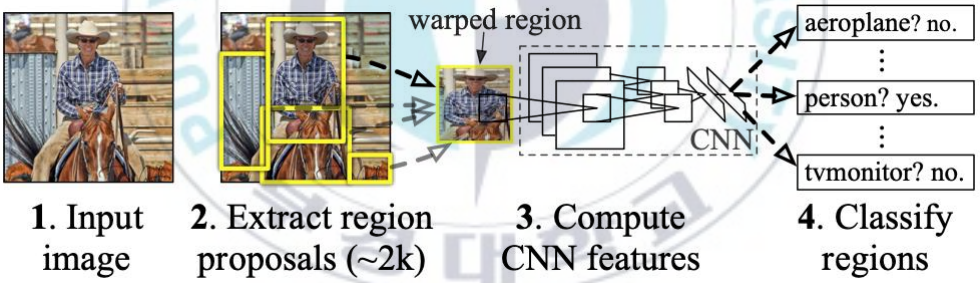


<그림 Ⅱ-8> 객체탐지 분야의 연구 동향

(1) R-CNN

R-CNN(Girshick, R. et al., 2014)은 이미지에 존재하는 객체가 어떤 물체인지 분류하는 합성곱 신경망과 Region Proposal 알고리즘을 연결해서 사용함으로써 딥러닝 이전의 객체탐지 연구에 비해 높은 성능을 보이는 딥러닝 기반의 객체탐지 모델을 제시하였다.

Region Proposal 알고리즘이란 이미지 상에서 객체 즉, 물체가 있을 확률이 높은 위치를 빠른 속도로 찾아내는 것을 의미한다. 이러한 Region Proposal을 위한 알고리즘은 대표적으로 Selective Search(Uijlings, J. R. et al., 2013)와 Edge Boxes(Zitnick, C. L. et al., 2014) 알고리즘이 있다. R-CNN의 경우 Region Proposal 알고리즘으로 Selective Search를 사용하여 합성곱 신경망과 결합한 후 모델을 제시하였다. <그림 Ⅱ-9>은 R-CNN 의 모델 구조를 나타낸다.



<그림 Ⅱ-9> R-CNN의 모델 구조

R-CNN은 먼저 이미지를 입력받은 후 Region Proposal 알고리즘을 통해 최대 2000개의 영역을 추출한다. 이미지에서 추출된 각 영역을 잘라낸 다음 모두 같은 크기의 이미지로 재설정 한 후 모두 합성곱 신경망을 통해 특징을 추출하게 된다. 특징을 추출한 후 각 영역에 존재하는 물체의 종류를 결정하기 위해 분류를 수행하여 물체의 종류를 예측한다. R-CNN을 제시한 논문의 저자는 물체의 종류를 예측하는 분류의 성능은

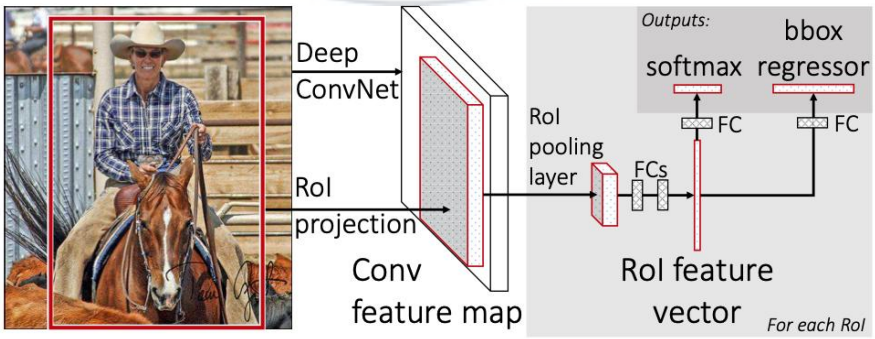
충분히 높다고 볼 수 있지만 이미지에서 물체의 위치를 정확히 잡아내는 것에 대해 취약하다고 하였는데, 이는 추출된 영역에서 탐지하고자하는 물체가 영역의 정중앙에 있지 않아도 합성곱 신경망은 높은 수준으로 물체의 종류를 맞춰 버리기 때문에 일어나는 현상이라고 할 수 있다.

이러한 현상을 해결하기 위해 Bounding-Box 회귀 방법을 제안하였으며 이는 예측한 위치와 실제 위치의 차이를 보정하여 정확한 위치에 가깝게 찾아낼 수 있도록 하는 방법이다. 이를 통해 예측 위치에 대한 취약점을 보완할 수 있었다.

R-CNN의 경우 한 장의 이미지에 대해 물체가 있을 확률이 높은 수많은 영역을 검출하여 이 모든 영역에 합성곱 신경망을 적용하므로 매우 높은 연산량이 요구된다. 이는 모델의 학습이 복잡해지고 매우 느려지는 원인이 된다.

(2) Fast R-CNN

Fast R-CNN(Girshick, R., 2015)는 이전의 R-CNN 모델의 단점인 학습 속도에 대한 개선방안을 제시하였다. <그림 Ⅱ-10>은 Fast-RCNN의 구조를 나타낸다.

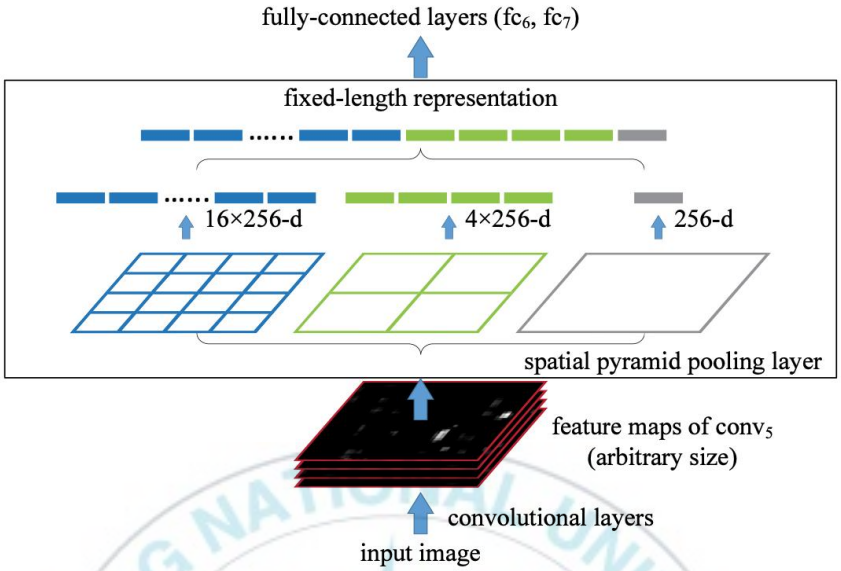


<그림 Ⅱ-10> Fast R-CNN 구조

이전의 R-CNN에서 Region Proposal 알고리즘에 의해 영역이 추출되면 이 영역을 합성곱 신경망에 입력하여 특징을 추출하는 과정을 거치게 된다. 이 과정에서 최대 2000개의 영역이 발생하면서 최대 2000번 까지 합성곱 신경망 연산이 일어나게 되는데 이는 매우 큰 속도 저하를 일으키게 된다. Fast R-CNN에서는 추출된 영역을 잘라내어 합성곱 연산을 수행하는 R-CNN과는 달리 합성곱 연산을 전체 이미지에 대해 먼저 수행한 후 영역을 잘라내는 방법을 사용함으로써 추출된 영역의 수가 몇 개가 될 지라도 합성곱 신경망의 연산은 1번만 수행하면 되므로 연산량이 크게 줄어들며 속도 차원에서의 문제점을 해결할 수 있다.

그러나 분류를 수행하기 위해서는 각 영역에서 추출된 특징의 크기가 같아야 하는데 R-CNN에서는 영역을 잘라낸 다음 크기를 모두 같게 수정하는 작업이 존재하지만 Fast R-CNN은 해당 과정이 존재하지 않아 각각 크기가 다양한 영역들에 대해서 일정한 크기의 특징을 추출해 내는 방법이 필요하다. 해결책으로 Spatial pyramid pooling networks(SPPnets)(He, K. et al., 2015)가 사용된다.

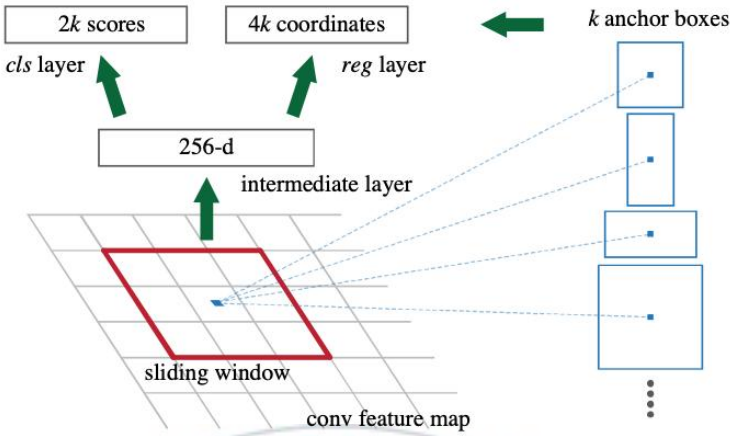
SPPnets은 SPP 계층을 합성곱 신경망에 적용시킨 기법으로 SPP 계층은 특정 영역을 일정한 개수로 나누고, 나뉘어진 부분에 대해 Pooling 작업을 수행함으로써 고정된 크기의 특징을 추출할 수 있도록 하는 것이다. <그림 Ⅱ-11>은 SPP 계층의 구조를 나타내며 Fast R-CNN에서는 이러한 SPP 계층을 RoI Pooling Layer라고 정의하였다. 이와 같은 방식을 사용하여 Fast R-CNN은 기존 R-CNN의 복잡한 학습과정을 해소하고 객체 검출의 학습속도를 크게 개선시켰다.



<그림 Ⅱ-11> SPP(Spatial Pyramid Pooling) 계층의 구조

(3) Faster R-CNN

Faster R-CNN(Ren, S. et al., 2015)은 Fast R-CNN 보다도 속도면에서 우월한 모델을 제시하였다. 앞선 연구에서는 Region Proposal 알고리즘을 합성곱 신경망과 분리하여 수행함으로써 속도저하의 큰 원인이 되었다. Faster R-CNN은 이러한 Region Proposal 알고리즘을 합성곱 신경망 외부가 아닌 내부에 신경망 설계를 통해 구현함으로써 속도와 함께 정확도까지 끌어올리게 되었다. 합성곱 신경망 내부에 존재하는 Region Proposal을 위한 신경망을 Region Proposal Network(RPN)이라고 정의하였으며 내부 수행 과정은 <그림 Ⅱ-12>과 같다.



<그림 Ⅱ-12> RPN의 수행 과정

특정 이미지 상에서 Sliding Window 방식으로 Window를 움직여가며 해당 영역에서 물체의 존재 유무를 판단하게 된다. 각각의 영역에서 물체가 모두 같은 비율로 존재하는 것이 아니므로 이를 보완하기 위해 k개의 비율을 사전에 정의하여 놓고 각 Sliding Window 단계에서 정의되어 있던 k개의 비율에 대해 확인하도록 설계하였다. k개로 정의된 물체의 비율들을 Anchor라고 명칭하였다.

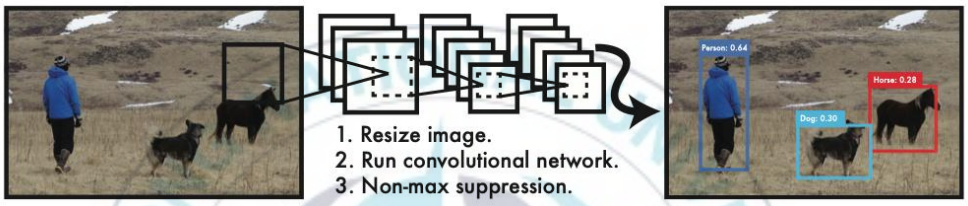
Faster R-CNN은 이전 연구들의 단점을 모두 보완하고 속도와 정확도 면에서 최고의 성능을 보였으나 학습과정이 매우 복잡한 단점을 가지고 있다.

(4) YOLO(You Only Look Once)

YOLO(Redmon, J. et al., 2016)는 이전 연구들과는 차별화된 새로운 방법으로 제시된 객체탐지 기술이다. 마찬가지로 이미지에서 물체의 위치와 종류에 대해 검출하는 것을 목표로 한다. YOLO는 입력 이미지에

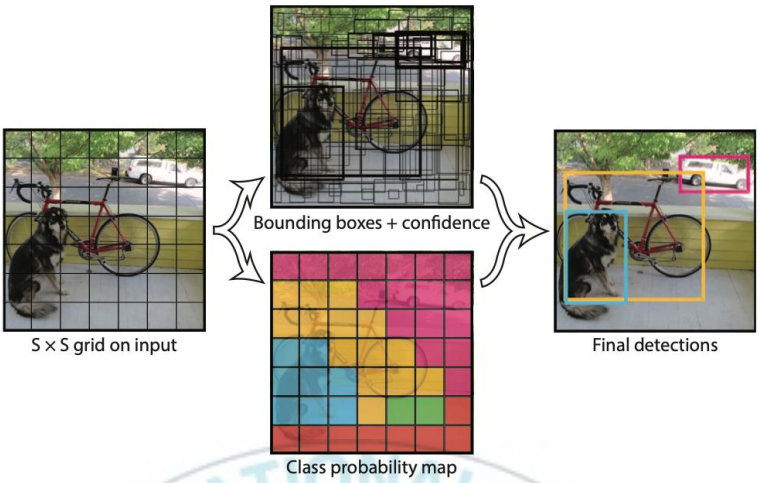
서 Bounding-Box와 물체의 종류를 분류하는 것을 하나의 문제로 간주하여 이미지를 신경망에 한번만 통과시키는 것으로 물체의 종류 그리고 위치에 대해 예측하게 된다.

<그림 Ⅱ-13>은 YOLO의 모델 진행 과정을 나타낸다. 그 과정이 매우 간단하며 직선적으로 이루어진다. 먼저 입력이미지의 사이즈를 448*448 크기의 이미지로 변환하고 합성곱 신경망에 통과시킨 후 Non-Maximum Suppression 알고리즘을 통해 객체탐지를 완료한다.



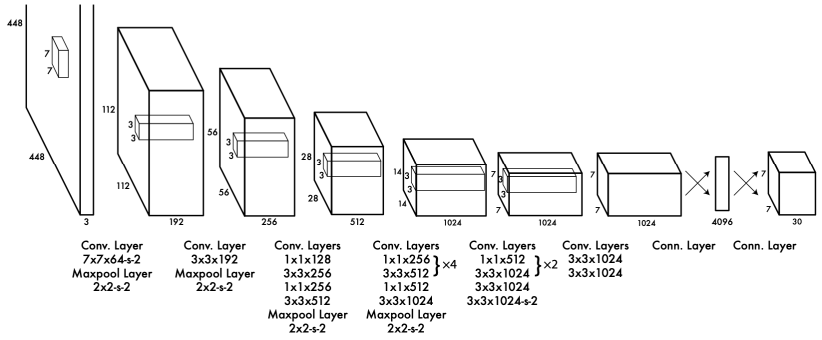
<그림 Ⅱ-13> YOLO의 기본 과정

YOLO의 장점은 과정에서도 볼 수 있듯이 매우 간단하며 실시간 객체탐지가 가능할 만큼 우수한 속도를 보인다. 속도가 중요시되는 실시간 객체탐지 모델 중에서도 가장 높은 정확도를 보였다. 앞선 R-CNN 연구들과는 달리 이미지의 특정 영역만 잘라서 보는 것이 아니라 전체 이미지를 한 번에 파악하기 때문에 실제 배경을 물체로 착각하여 탐지할 수 있는 오류가 줄어든다. 그러나 실시간 객체탐지 모델 중에서는 가장 높은 정확도를 보이고 있긴 하지만 R-CNN과같은 복잡한 모델에 비해 상대적으로 낮은 성능을 보여준다.



<그림 Ⅱ-14> YOLO의 객체탐지 과정

<그림 Ⅱ-14>은 YOLO의 객체탐지 과정을 나타낸다. 먼저 입력 이미지에 대해 S*S개의 격자로 구분하고 각각의 격자 영역에는 B개의 Bounding-Box를 가진다. 이 각각의 Bounding-Box는 물체를 포함하고 있을 확률에 따라 Confidence Score값을 가지게 된다. Confidence Score는 실제 물체가 존재하는 박스와 예측된 Bounding-Box의 겹치는 정도로써 IOU(Intersection Over Union)값이라 표현하기도 한다. 결국 한 격자에는 B개의 예측 가능한 Bounding-Box가 존재하고 이 박스는 곧 중심점의 좌표와 가로 세로 길이를 통해 구분된다.

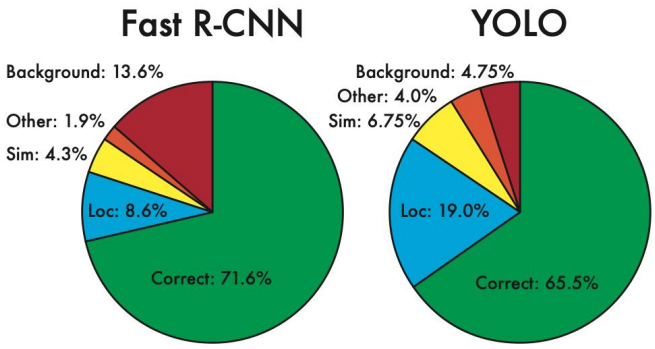


<그림 Ⅱ-15> YOLO의 신경망 모델 구조

<그림 Ⅱ-15>은 YOLO의 신경망 구조를 나타낸다. 입력 이미지를 받은 후 24개의 합성곱 신경망 층과 2개의 완전 연결층을 통과시켜 객체탐지 예측을 진행하는 구조이다.

YOLO는 입력 이미지를 격자로 구분지으면서 각각의 격자는 오직 하나의 물체만 예측할 수 있게 된다. 아주 작은 물체들이 격자 안에 좁은 간격으로 붙어서 존재한다면 잘 예측하지 못하는 경우가 일어난다. 따라서 YOLO는 큰 물체가 많은 이미지에 더욱 유리하다는 것을 확인할 수 있다.

<그림 Ⅱ-16>는 YOLO와 Fast R-CNN의 성능 비교를 보여준다. 정확도 측면에서는 Fast R-CNN이 역시 높은 값을 보여주지만 YOLO의 경우 배경 오류가 크게 낮은 것을 확인할 수 있다. 이러한 점에서 매우 높은 값의 정확도가 보장되지는 않지만 실시간 객체검출이 가능하다는 점과 합쳐 YOLO의 그 성과는 매우 뛰어나다고 할 수 있다.

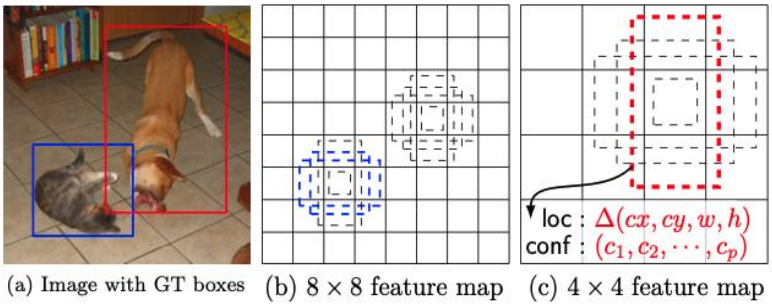


<그림 Ⅱ-16> YOLO와 Fast R-CNN의 성능 차트

(5) SSD(Single Shot multibox detector)

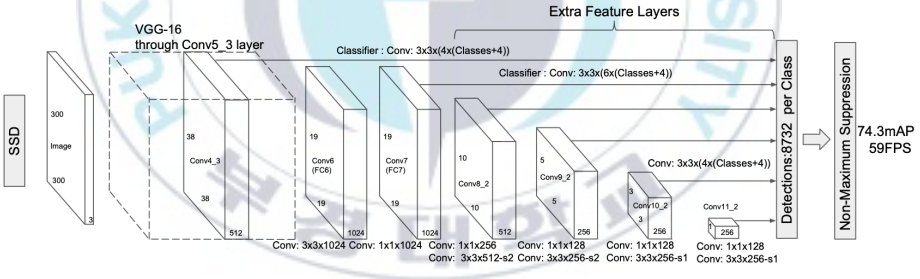
SSD(Liu, W. et al., 2016)는 여러 개의 특징맵을 활용하여 각 특징맵에서 다양한 크기의 Default Box를 생성해내고 물체가 존재할 확률이 있는 Default Box에 대해서 객체탐지를 수행하는 모델로서 당시 객체탐지 모델의 성능과 속도 면에서 모두 최고의 성능을 보여주었다.

다른 객체탐지연구와 마찬가지로 박스를 생성하고 각 박스에 대한 위치정보와 Score를 예측하게 되는데 SSD에서는 이러한 박스를 Default Box라 명칭하였으며, 고정된 크기의 박스를 사용한다. YOLO에서 작은 물체의 탐지가 불리한 것에 대한 단점이 존재하였는데 이를 해결하기 위해 여러 개의 특징맵을 사용하고 각각의 특징맵에서 비율이 다른 물체의 예측을 가능하도록 하였다. <그림 Ⅱ-17>은 SSD에서 서로 다른 크기의 특징맵을 사용하는 것을 보여준다. 해당 논문에서는 6가지 크기의 특징맵을 사용하여 작은 물체부터 큰 물체까지 차이를 두지 않고 객체검출을 이루어낼 수 있다. 큰 특징맵에서는 작은 물체를 탐지하고 작은 특징맵에서는 큰 물체를 탐지할 수 있다.



<그림 Ⅱ-17> SSD의 다양한 크기를 가지는 특징맵

SSD 또한 마찬가지로 입력 이미지에 대해 물체를 탐지하면 해당 물체에 대한 이미지 상에서의 위치와 어떤 종류의 물체인지가 출력되어야 한다. <그림 Ⅱ-18>은 SSD의 신경망 구조이다.



<그림 Ⅱ-18> SSD의 신경망 모델 구조

이미지가 입력되면 SSD에 맞는 해상도를 적용하기 위하여 300*300의 크기로 이미지가 조절된다. 조절된 이미지는 SSD의 Base Network로써 사용되는 VGG-16(Simonyan, K. et al., 2014) 네트워크를 통해 처리가 이루어진다. 그리고 점점 작아지는 합성곱 신경망들을 적용하면서 6개의 특징맵을 얻을 수 있다. 해당 6개의 특징맵에서 Default Box를 예측하게

되는데 이때 생성되는 Default Box의 개수는 한 이미지에 8,732개이다. 수많은 Default Box를 모두 사용하지는 않고 앞서 YOLO에서 언급했던 IOU값을 통해 임계치가 넘어가는 Default Box만 추출하여 학습과 예측에 사용하게 된다. 해당 방법을 통해 Default Box를 추출한 이후 YOLO와 비슷한 과정으로 객체탐지 과정이 진행된다.

SSD는 완전 연결층이 없어 가중치 수가 눈에 띄게 작으며 여러 크기의 특징맵을 사용해 다양한 크기의 물체를 탐지 할 수 있다는 점에서 YOLO의 단점도 보완하였다. 이러한 이유들로 속도와 정확도 모두 당시 최고의 성능을 보이고 있었다.

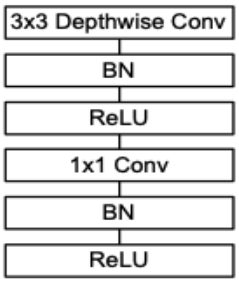
(6) MobileNet

MobiletNet(Howard, A. G. et al., 2017)은 심지어 모바일 기기의 CPU에서도 신경망의 연산이 실시간으로 이루어지는 것을 보여준다. MobileNet의 핵심 아이디어는 연산량이 많아 무거운 합성곱 계층을 줄이는 방법으로 Depthwise Separable Convolution을 사용한 것이다. <그림 Ⅱ-19>는 MobileNet의 신경망 구조를 나타낸다.

Type / Stride	Filter Shape	Input Size
Conv / s2	3 × 3 × 3 × 32	224 × 224 × 3
Conv dw / s1	3 × 3 × 32 dw	112 × 112 × 32
Conv / s1	1 × 1 × 32 × 64	112 × 112 × 32
Conv dw / s2	3 × 3 × 64 dw	112 × 112 × 64
Conv / s1	1 × 1 × 64 × 128	56 × 56 × 64
Conv dw / s1	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 128	56 × 56 × 128
Conv dw / s2	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 256	28 × 28 × 128
Conv dw / s1	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 256	28 × 28 × 256
Conv dw / s2	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 512	14 × 14 × 256
5× Conv dw / s1	3 × 3 × 512 dw	14 × 14 × 512
5× Conv / s1	1 × 1 × 512 × 512	14 × 14 × 512
Conv dw / s2	3 × 3 × 512 dw	14 × 14 × 512
Conv / s1	1 × 1 × 512 × 1024	7 × 7 × 512
Conv dw / s2	3 × 3 × 1024 dw	7 × 7 × 1024
Conv / s1	1 × 1 × 1024 × 1024	7 × 7 × 1024
Avg Pool / s1	Pool 7 × 7	7 × 7 × 1024
FC / s1	1024 × 1000	1 × 1 × 1024
Softmax / s1	Classifier	1 × 1 × 1000

<그림 Ⅱ-19> MobileNet 신경망 구조

Depthwise Separable Convolution은 <그림 Ⅱ-20>과 같은 구조로써 Depthwise Convolution Filter와 Pointwise Convolution Filter가 합쳐져 있는 구조이다. 이를 사용함으로써 모델의 성능을 1.1% 정도 떨어뜨리지만 반면에 연산량을 약 9배 줄이는 결과를 보여주었다.



<그림 Ⅱ-20> Depthwise Seperable Convolution의 구조

이외에 신경망의 크기를 더 작게 줄이기 위해 두 가지의 파라미터 Width Multiplier와 Resolution Multiplier를 사용하였다. 전자는 입력과 출력 데이터의 채널수를 일정한 비율로 조절하고 후자는 입력 이미지의 크기를 특정 비율만큼 조절하는 것이다. 모델의 성능과 연산량의 Tradeoff 관계에서 이 비율을 적절하게 조절하는 것이 중요한 것으로 판단된다.

(7) MobileNetV2

MobileNetV2(Sandler, M. et al., 2018)은 기존 MobileNet의 방식들을 사용하면서 Linear Bottlenecks를 추가하였다. 이는 1*1 합성곱 연산을 통해 데이터의 채널 수를 줄이고 3*3의 일반적인 합성곱 연산을 수행한다. 그 다음 다시 1*1의 합성곱 연산을 수행하여 늘어난 채널 수를 다시 줄이는 방식이다. 채널 수를 감소시키면서 앞선 MobileNet 보다도 메모리 측면에서 효율적인 신경망을 구성할 수 있다. <그림 Ⅱ-21>은 19개의 층으로 구성된 MobileNetV2의 신경망 구조를 나타낸다.

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

<그림 Ⅱ-21> MobileNetV2의 신경망 구조

실시간 객체 탐지 분야에서 특징 추출을 위해 MobileNetV2가 SSD 모델의 Base Network로 사용되었을 때 모델의 연산량에 비해 높은 성능을 보여 본 연구에서는 MobileNetV2를 SSD와 함께 사용하였다.

3. 도로 균열 탐지

자동차 도로로 사용되는 아스팔트의 파손은 사고와도 직결되는 중요한 안전문제가 될 수 있다. 이러한 문제를 예방하기 위하여 파손이나 마모에 대해 적시적인 탐지와 유지보수는 필수적이다. 아스팔트 혹은 콘크리트의 균열을 찾아내는 것에 대한 연구는 이미지 처리와 해석적 알고리즘들을 통한 균열 탐지 연구와 머신러닝 혹은 딥러닝을 이용한 균열탐지 연구로 나뉘게 되는데 이 두 가지 방법의 관련 연구들에 대해 정리해 보았다.

가. 이미지 처리와 알고리즘을 이용한 균열 탐지 연구

이미지 처리와 알고리즘 방식의 균열 탐지 연구는 딥러닝 이전 과거부터 진행되어오던 전통적인 균열 탐지 연구 방식으로 보통 이미지를 행렬화한 데이터에 특정한 알고리즘을 적용하거나 특징적인 부분을 추출해 내는 것으로 균열을 탐지해 왔다.

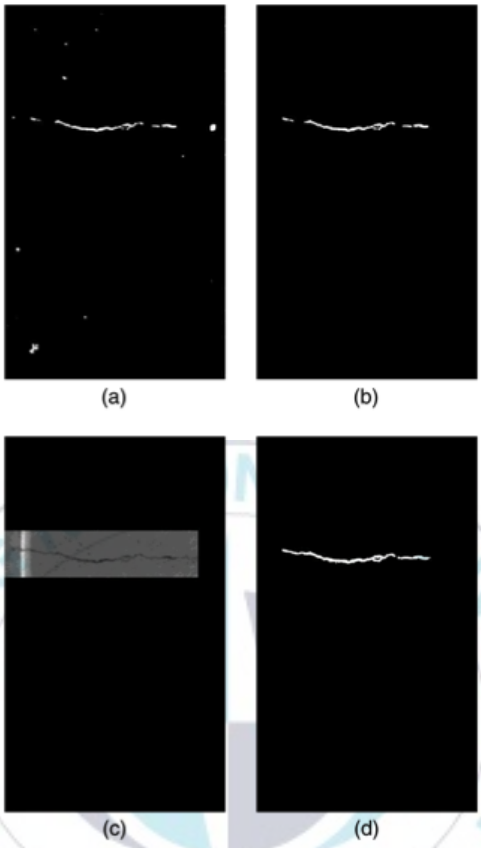
Haas, C. et al.(1991)에서 균열 탐지를 자동화하기 위한 연구를 제안하기 시작하면서 도로의 포장상태를 효율적으로 점검하기 위해 패칭이나 균열과 같은 도로의 결함들을 탐지하는 연구가 활발히 일어나기 시작했다.

Radopoulou, S. C. et al.(2015)에서는 도로 결함인 패칭이 윤곽선으로 둘러쌓여 있고 표면의 질감이 주변 환경과 유사하게 보인다는 시각적 특성을 인지하여 패칭을 자동으로 탐지하였다.

패칭이나 포트홀이아닌 거북등균열, 종방향균열, 횡방향균열과 같은 경우 얇은 선으로 그 균열이 이루어져있다. 이러한 선형 특성 때문에 균열 탐지가 상대적으로 어렵다. Cheng, H. D. et al.(2003)에서는 일반 도로 표면보다 균열이 이미지 상에서 더 어둡게 나타난다는 점을 기반으로 밝기가 유사한 이미지들의 픽셀값 평균과 표준 편차를 계산하여 균열을 식별할 수 있는 임계값을 결정하는 방법을 제안하였다.

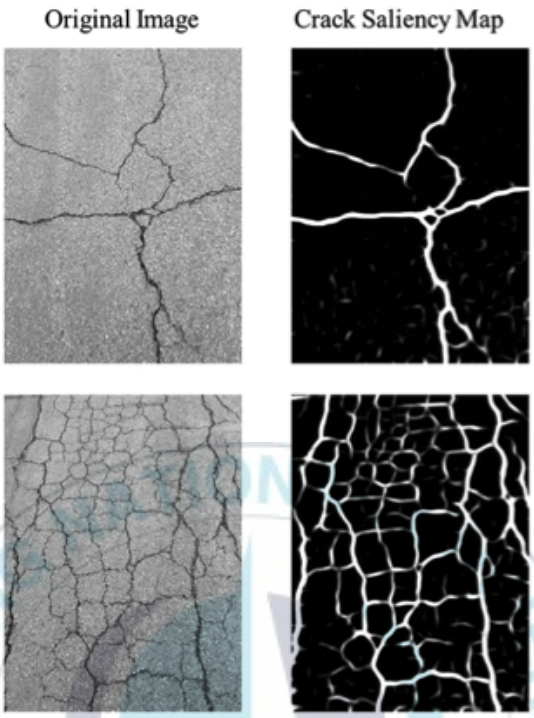
Oliveira, H. et al.(2012)는 Otsu의 임계치 방법을 적용하여 2단계 균열 탐지와 특성화를 통해 도로 이미지에서 균열을 탐지하였다.

Sun, L. et al.(2015)에서는 기존 임계치 기반의 균열 탐지 연구에서 균열 검출 능력의 향상을 위해서 균열일 수 있는 후보를 위한 임계치와 실제 균열에 대한 임계치라고 정의한 두 개 값의 임계치를 이용한 방법을 통해 <그림 Ⅱ-22>와 같이 균열을 탐지하였다.



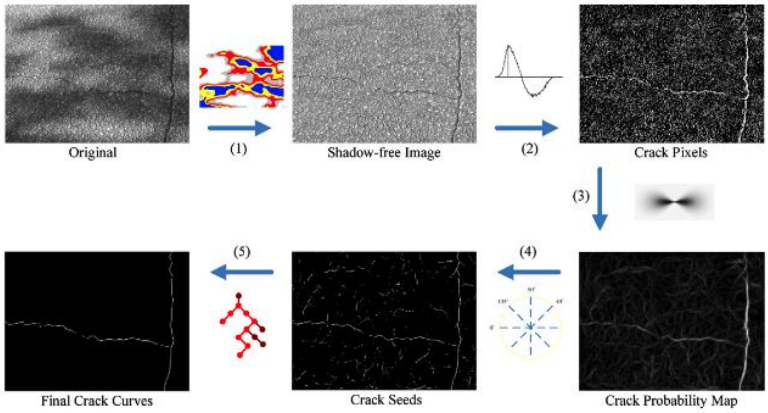
<그림 Ⅱ-22> 균열 탐지 결과(Sun, L. et al., 2015)

Li, S. et al.(2017)에서도 또한 Otsu의 임계값을 이용한 방법을 활용하였다. 균열이 두드러지게 나타나는 부분을 확인하고 Saliency Map을 생성하여 <그림 Ⅱ-23>와 같이 균열을 가시화하였다.



<그림 Ⅱ-23> Crack Saliency Map을 통한 균열 탐지

균열이 존재하는 이미지에서 도로상에 그림자가 생겨 밝기가 불균일 하거나 도로 표면상에 노이즈가 많은 경우 균열 탐지에 영향을 끼칠 수 있다. Zou, Q. et al.(2012)에서는 포장도로상의 그림자를 제거하는 알고리즘과 Crack Probability Map을 사용하는 균열 탐지 모델인 CrackTree를 제시하였다. <그림 Ⅱ-24>는 CrackTree의 균열탐지 과정에 대해 나타낸다.



<그림 Ⅱ-24> CrackTree의 균열 탐지 과정

나. 머신러닝 및 딥러닝을 이용한 균열 탐지 연구

머신러닝 기법들과 딥러닝이 발전하면서 이를 이용한 균열 탐지 기술들이 제안되기 시작했다. 특히 합성곱 신경망을 사용한 이미지 및 영상 분석 연구들에서 높은 성능을 보임으로써 균열 탐지 분야에서도 축적된 균열 데이터를 통해 딥러닝 모델을 구성하고 균열의 형태와 같은 특징 패턴을 학습하여 균열을 탐지하는 기술들이 연구되고 있다. 머신러닝과 딥러닝을 이용한 몇 가지 균열 탐지 모델을 정리해 보았다.

Cord, A. et al.(2012)에서는 도로 포장 상태의 결함 탐지를 자동화하기 위해 텍스트 패턴 인식을 사영한 방법을 제시하였다. Jahanshahi, M. R. et al.(2012)에서는 판별분석과 SVM(Support Vector Machine) 그리고 신경망을 이용하여 균열 검출법을 제시하였다.

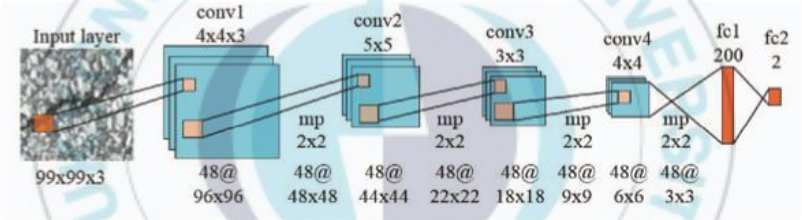
Radopoulou, S. C. et al.(2016)는 주차장의 카메라 이미지를 수집하여 Semantic Texon Forests방식을 통한 포장 결함 탐지모델을 제시하였다. Random Forest 기법으로 이미지의 균열을 검출한 연구 또한

존재한다(Shi, Y. et al., 2016).

많은 연구들이 진행되어 왔지만 위와 같은 연구들을 바탕으로 한 방법에서 실제 균열이 아닌 영역을 균열로 분류하는 경우가 빈번하게 일어났으며 여전히 그림자와 같은 외부 환경에 따라 균열 탐지가 힘들어질 수 있다.(Gopalakrishnan, K. et al., 2017).

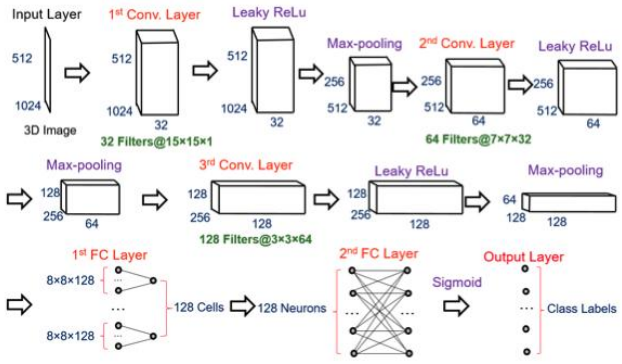
이러한 균열 탐지 관련 연구 동향에서 컴퓨터 성능의 발달 그리고 오랜 기간 축적된 균열 데이터를 통해 딥러닝을 활용한 균열 탐지 연구가 활발히 일어나기 시작하였다.

Zhang, L. et al.(2016)은 <그림 Ⅱ-25>와 같은 합성곱 신경망 구조가 이미지 상에서 균열 이미지를 잘 분류해 낼 수 있다는 것을 보여주었다.



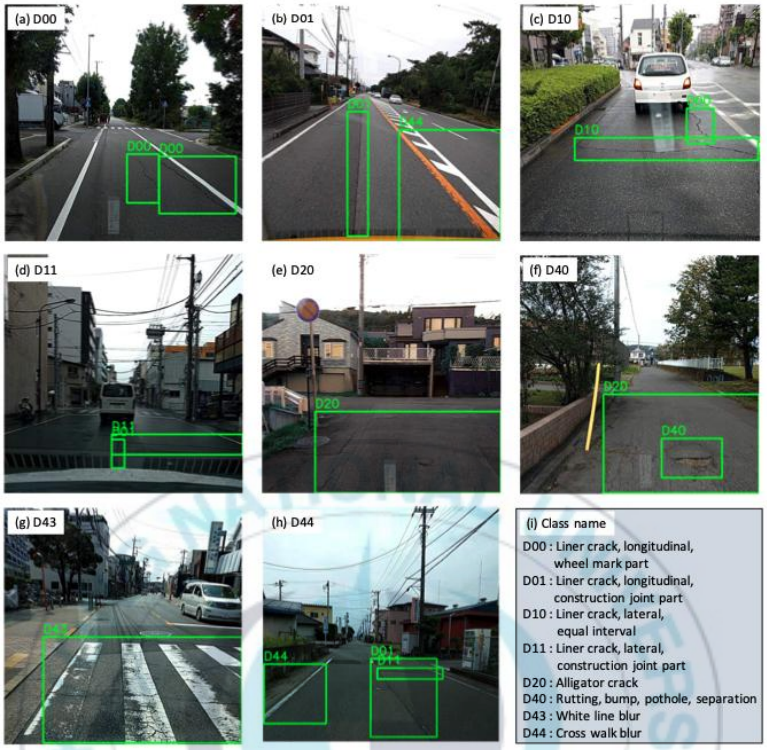
<그림 Ⅱ-25> 제안된 신경망 구조(Zhang, L. et al., 2016)

Wang and Hu (2017, August)에서는 종방향균열, 횡방향균열, 그리고 거북등균열에 대한 탐지를 위해 주성분분석과 64*64 크기의 패치를 사용하였다. Wang, K. C. et al.(2017)은 아스팔트 도로상의 균열을 탐지하기 위해 3개의 합성곱 층과 2개의 완전 연결층으로 구성된 <그림 Ⅱ-26>와 같은 합성곱 신경망을 제안하였다.



<그림 Ⅱ-26> 제안한 신경망 구조(Wang, K. C. et al., 2017)

Maeda, H. et al.(2018)은 차량에 스마트폰을 장착하여 8가지의 종류의 도로 포장 결함에 대해 탐지하는 모델을 제안하였다. <그림 Ⅱ-27>는 제안된 모델에 의한 탐지 결과를 나타낸다. 균열 이미지의 품질이 높아 어느 정도 학습이 진행되었으나, 8가지 중 4가지의 유형에서 낮은 Recall 값을 보이며 높은 성능을 보여주지는 못하였다.

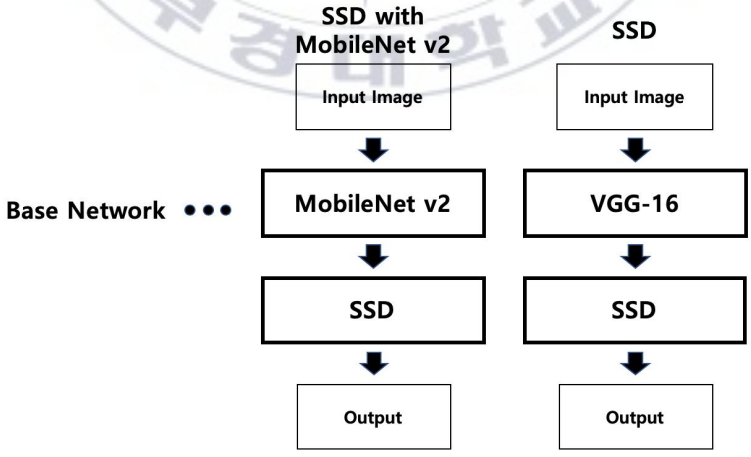


<그림 Ⅱ-27> 균열 탐지 결과(Maeda, H. et al., 2018)

Ⅲ. 파일럿 시스템의 균열 탐지 설계

1. 제안된 DNN(Deep Neural Network)의 구조

균열탐지를 위한 딥러닝 객체탐지 모델로서 SSD 모델을 선택하여 학습을 진행하였다. 기본이 되는 SSD 모델은 VGG-16을 Base Network로 하여 제안되었다. 그러나 VGG-16 모델은 매우 깊은 신경망 구조로 이미지 학습을 위해 높은 성능의 컴퓨터 연산을 필요로 한다. 이는 엣지 단말에서 이미지를 처리해야하는 엣지 컴퓨팅 환경에 적합하지 않은 모델로 판단하여 Base Network로 MobileNetV2(Sandler, M. et al., 2018)가 적용된 SSD 모델을 학습하여 사용하였다. <그림 Ⅲ-1>은 MobileNetV2를 활용한 SSD 모델의 구조를 나타낸다. 기존의 SSD 모델을 유지하면서 Base Network만 MobileNetV2로 변경한 것으로 모바일 기기 상에서 SSD 객체탐지가 가능하도록 한다. 엣지 컴퓨팅 환경의 설계를 위해서는 모바일 기기 상에서 구동되는 모델을 선택하는 것이 필수적이다.



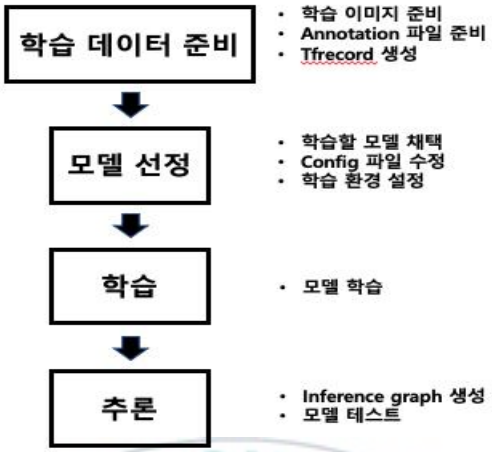
<그림 Ⅲ-1> MobiletNetV2를 활용한 SSD 모델의 구조

본 연구에서는 엣지 단말로 사용되는 젯슨 나노의 성능과 환경에 맞도록 VGG-16의 깊은 신경망 구조를 사용하는 대신 엣지 단말에서도 학습된 모델의 사용이 가능하도록 경량화된 MobileNetV2를 이용하는 SSD 모델을 채택하였다.

2. DNN의 학습 및 테스트 결과

객체탐지를 위한 모델로서 SSD 모델을 채택하여 학습을 진행하였다. 채택한 신경망 모델의 학습에 필요한 학습 데이터를 준비하고 학습 환경을 설정한 뒤 학습을 완료하는데 까지 있어 Python 3와 Tensorflow 라이브러리를 통해 직접 코드를 구현한 프로그램을 통해 진행하였으나, 학습 장비의 성능적인 한계가 있어 학습이 원활히 이루어지지 않는 현상이 발생하였다.

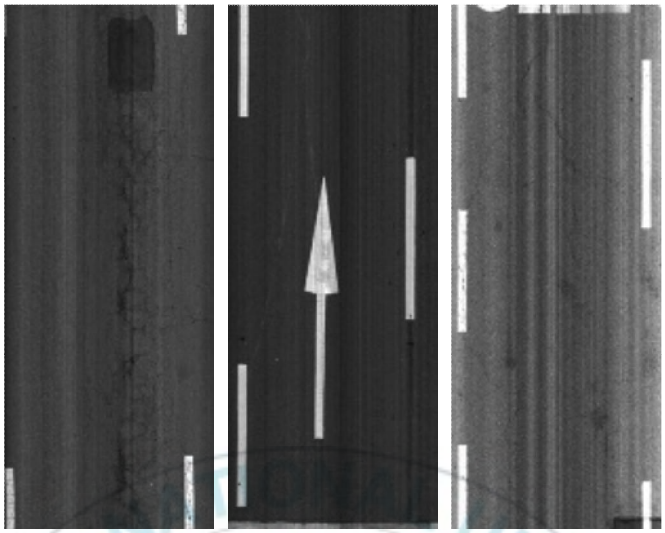
이에 Tensorflow Object Detection API를 사용하여 학습을 진행하였으며 이는 Google의 Protocol Buffer를 사용하며 C언어 기반으로 작성된 스크립트 코드를 통해 학습을 진행할 수 있으므로 최적화가 매우 잘 이루어져 있어 비교적 빠른 시간에 학습을 진행할 수 있었다. 학습과정에 대해 정리하고 학습된 모델의 결과에 대해 정리해보았다. Tensorflow Object Detection API를 이용한 학습의 과정은 <그림 Ⅲ-2>와 같다. 각 단계별로 본 연구에서 적용한 방법을 정리해보았다.



<그림 Ⅲ-2> Tensorflow Object Detection API 사용과정

가. 학습 데이터 준비

신경망의 학습을 위한 이미지 데이터를 준비하는 과정으로 본 연구에서는 10000*3739 크기의 도로 이미지를 수집하였다. <그림 Ⅲ-3>는 학습 데이터의 샘플 이미지이다. 도로 균열의 종류는 균열의 모양에 따라 종방향균열(LC), 횡방향균열(TC), 거북등균열(AC) 그리고 포트홀과 패칭이 있다. 본 연구에서는 포트홀과 패칭을 하나의 균열 종류로 간주하고 학습을 진행하였다. 따라서 모델이 학습하는 균열의 종류는 LC, TC, AC 그리고 포트홀과 패칭을 하나로 묶은 POTPTC이다. 따라서 모델의 학습을 통해 위와 같은 총 4가지 종류의 균열에 대해 탐지하는 모델을 학습하고자 하였다.



<그림 Ⅲ-3> 10000*3739 크기의 학습 이미지 샘플

원본 이미지를 그대로 학습에 사용하기에는 이미지의 크기가 매우 커 Python의 이미지 및 영상처리 라이브러리인 OpenCV를 활용하여 Image Pyramid 방식으로 이미지를 축소하였다. SSD 모델의 입력이미지 크기가 300*300이므로 그와 근접하도록 이미지의 크기를 축소해 나가 최종적으로 학습을 위해 600*220의 크기로 축소하였다.

또 객체탐지 모델의 학습을 위해서는 이미지의 기본 정보나 포함하고 있는 객체에 대한 정보가 저장되어 있는 Annotation이라 불리는 파일이 이미지 1개마다 하나씩 존재하여야한다. <그림 Ⅲ-4>는 본 연구에서 생성한 Annotation 파일의 샘플이며 이는 Pascal VOC의 Annotation 양식에 맞추어 생성하였다. Pascal VOC 양식의 Annotation 파일에는 파일의 이름과 크기 그리고 해당 이미지에 포함되어 있는 객체에 대한 정보를 담고 있다.

```
<?xml version="1.0" encoding="UTF-8"?>
<annotation>
  <folder>CRACK</folder>
  <filename>공항로_s005900000.jpg</filename>
  <size>
    <width>3739</width>
    <height>10000</height>
    <depth>3</depth>
  </size>
  <object>
    <name>AP종방향균열</name>
    <pose>Unspecified</pose>
    <difficult>0</difficult>
    <bndbox>
      <xmin>520</xmin>
      <ymin>9500</ymin>
      <xmax>820</xmax>
      <ymax>9800</ymax>
    </bndbox>
  </object>
</annotation>
```

<그림 Ⅲ-4> Annotation 파일 샘플

이미지와 Annotation이 모두 준비되었을 경우, 학습을 위한 데이터와 테스트를 위한 데이터를 분리시킨다. 본 연구에서는 전체 이미지의 무작위한 80%를 학습용 데이터로 사용하였다.

먼저 이미지와 Annotation을 통해 <그림 Ⅲ-5>와 같이 CSV 파일을 생성한다. 마지막으로 학습을 진행하기 전 준비한 데이터들을 TFRecord 형식으로 변환하였다. TFRecord는 Tensorflow 학습을 위해 최적화되어 데이터들을 이진화 형태로 변환하여 저장하는 방법을 의미한다. 생성한 CSV

파일을 바탕으로 본 연구를 위한 TFRecord 파일을 생성하였다. 이를 이용하면 학습 코드가 간결해지고 메모리 활용도를 높혀 효율적인 학습이 가능하다.

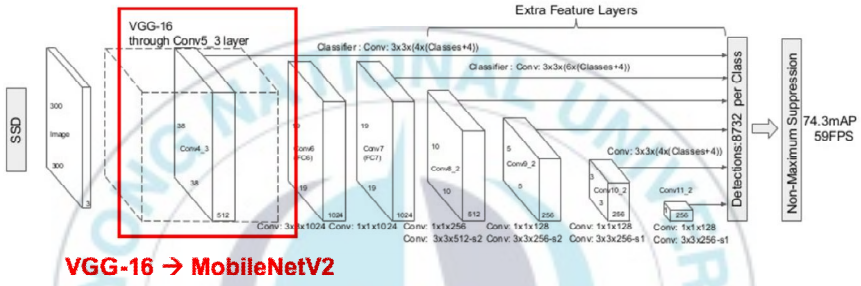
filename	width	height	class	xmin	ymin	xmax	ymax
경인로-상행1_s004260000.jpg	220	600	POTPTC	171	54	189	72
경인로-상행1_s004260000.jpg	220	600	POTPTC	171	90	189	108
경인로-상행1_s004260000.jpg	220	600	POTPTC	189	54	207	72
경인로-상행1_s004260000.jpg	220	600	POTPTC	189	252	207	270
경인로-상행1_s004260000.jpg	220	600	AP횡방향균열	207	486	220	504
경인로-상행1_s004260000.jpg	220	600	AP횡방향균열	189	486	207	504
경인로-상행1_s004260000.jpg	220	600	AP횡방향균열	154	504	171	522
경인로-상행1_s004260000.jpg	220	600	AP횡방향균열	30	522	48	540
경인로-상행1_s004260000.jpg	220	600	AP횡방향균열	171	486	189	504
공항로_s005910000.jpg	220	600	AP종방향균열	154	510	171	528
공항로_s005910000.jpg	220	600	AP종방향균열	154	582	171	600
공항로_s005910000.jpg	220	600	AP종방향균열	154	492	171	510
공항로_s005910000.jpg	220	600	AP종방향균열	154	564	171	582
공항로_s005910000.jpg	220	600	AP종방향균열	136	402	154	420
공항로_s005910000.jpg	220	600	AP종방향균열	154	474	171	492
공항로_s005910000.jpg	220	600	AP종방향균열	136	384	154	402
공항로_s005910000.jpg	220	600	AP종방향균열	154	456	171	474

<그림 Ⅲ-5> 학습 데이터 CSV 파일

나. 학습 모델 선정 및 학습

학습 데이터를 준비한 후 데이터와 모델의 용도에 맞도록 학습할 모델을 선정한다. 본 연구에서는 엣지 컴퓨팅 환경의 구현을 위해 비교적 가벼운 SSD with MobileNetV2 모델을 선택하였기 때문에 Tensorflow Object Detection API에서 해당 모델의 설정파일을 가져와 학습 환경을 설정하였다. SSD는 입력이미지에 대해 신경망을 한 번만 통과하는 것

만으로도 객체 탐지를 이루어낼 수 있다. 이미지 상에서 수많은 Default Box들을 정의한 후 각각의 박스에서 객체가 있는지 없는지, 존재한다면 어떤 종류의 객체인지 확인하는 작업을 거치게 된다. 이 때 Default Box들의 크기가 여러가지 비율에 의해 생성되는데 이를 통해 SSD는 객체의 크기에 민감하게 반응할 수 있으므로 이미지 속 객체의 크기가 다양하게 존재 하더라도 유리한 조건에서 객체 탐지를 수행할 수 있다. <그림 Ⅲ-6>는 본 연구에서 사용된 SSD 모델의 신경망 구조를 나타낸다.



<그림 Ⅲ-6> SSD 모델의 신경망 구조

SSD 모델의 학습과정도 타 모델과 마찬가지로 비용함수를 최소화 하는 방향으로 진행된다. 본 연구에서 사용되는 SSD는 최종적으로 균열의 위치에 대한 예측과 균열의 종류에 대한 예측 값을 출력하게 된다. 따라서 SSD의 비용함수는 위치에 대한 비용함수(Loss_loc)와 균열의 종류를 판단하는 분류에 대한 비용함수(Loss_conf)로 이루어져 있으며 이 두 비용함수의 값을 더하여 모델 전체의 Loss 값을 구할 수 있다. 수식 (1), (2)는 각각 Loss_loc과 Loss_conf을 나타낸다. x_{ij}^{p}는 i번째 Default Box와 j번째 Ground Truth에서 p 종류의 객체를 탐지했다면 1, 반대의 경우 0의 값을 가지게 된다. N은 생성한 Default Box의 개수이며, I는 예측한 객체의 개수이다. g는 실제 정답을 알고있는 데이터인 Ground Truth를

의미하고, d는 생성한 Default Box를 의미한다. cx, cy, w, h는 해당 Box의 중앙 좌표와 가로, 세로 길이를 나타낸다. Loss_loc과 Loss_conf를 각각 계산하여 두 값을 더한 값이 모델의 전체 Loss 값이라 할 수 있다.

$$Loss_{loc}(x,l,g) = \sum_{i \in Pos}^{N} \sum_{mINcx,\,cy,\,w,\,h} x_{ij}^{k} smooth_{L1}(l_i^m - \hat{g}_j^m)$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$$

$$\hat{g}_j^w = \log(\frac{g_j^w}{d_i^w}) \qquad \hat{g}_j^h = \log(\frac{g_j^h}{d_i^h})$$

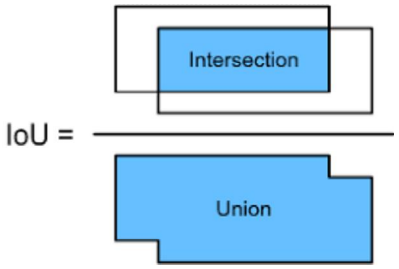
(1)

$$Loss_{conf}(x,c) = -\sum_{i \in Pos}^{N} x_{ij}^p \log(\hat{c}_i^p) - \sum_{i \in \neg} \log(\hat{c}_i^0)$$

$$\hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$$

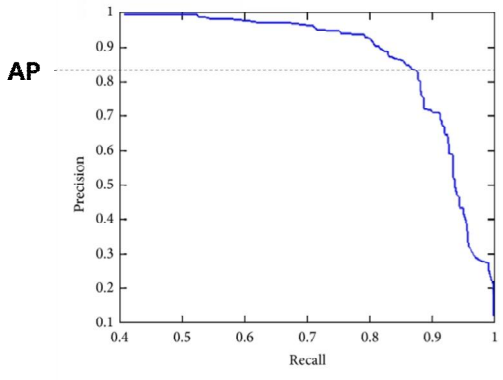
(2)

그리고 모델이 예측하여 표시한 객체의 박스와 실제 정답과의 IoU 값을 통해 모델이 객체를 올바르게 예측했는가를 판단하게 되는데 먼저 IoU의 임계값을 지정하고 이를 넘을 경우 맞게 예측하였다고 판단한다. 임계값은 주로 0.5 또는 0.7을 사용하며 해당 수치는 모델의 성능에 큰 영향을 미칠 수 있다. <그림 Ⅲ-7>은 예측한 객체 별 IoU 값의 정의를 나타낸다.



<그림 Ⅲ-7> 예측 위치와 실제 정답간의 IoU 값

객체 탐지 모델의 성능판단은 mAP(Mean Average Precision)을 통해 이루어진다. 이는 모델이 예측할 수 있는 모든 종류의 객체에 대해서 AP(Average Precision) 값을 구하고 AP 값들의 평균을 통해 모델의 mAP를 도출할 수 있다. 각 객체 종류에 대한 AP 값은 정밀도-재현율 곡선(Precision-Recall Curve)를 통해 계산할 수 있다. <그림 Ⅲ-8>는 P-R 곡선의 예를 보여준다. 정밀도(Precision)란 모든 검출 결과 중 옳게 검출한 비율을 의미하며 재현율(Recall)은 검출해내야하는 객체들 중 옳게 검출된 것의 비율을 의미한다. 수식 (3), (4)는 정밀도와 재현율을 나타낸다.



<그림 Ⅲ-8> Precision-Recall 곡선의 예시

$$\text{정밀도}(precision) = \frac{TP}{TP+FP} = \frac{TP}{all\ \ detections} \quad (3)$$

$$\text{재현율}(recall) = \frac{TP}{TP+FN} = \frac{TP}{all ground truths} \quad (4)$$

<표 Ⅲ-1> 실제 값과 예측 값에 다른 분류

실제 값 (Ground Truth)	예측값(Predicted)	
	Positive	Negative
Positive	TP (True Positive)	FN (False Negative)
Negative	FP (False Positive)	TN (True Negative)

학습 환경은 Inten Xeon 4116 2.10GHz의 CPU와 32GB RAM 그리고 GTX 1050Ti의 GPU를 사용하였으며, OS는 Ubuntu 16.04LTS 버전을 사용하였다.

우선적으로 SSD with MobileNetV2에 대한 .Config 파일을 수정한 후 API에 포함되어 있는 train.py 파이썬 파일의 실행을 통해 학습을 실행하였다. 학습 과정에서 실시간으로 모델의 Loss 값을 확인해 볼 수 있으며 이를 통해 학습이 원활히 이루어지고 있는 것인지 파악할 수 있다.

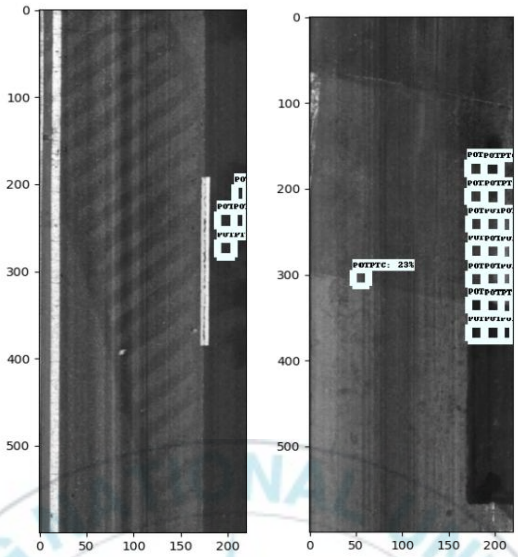
(1) 모든 데이터셋을 이용한 학습

학습에 이용한 데이터셋에서 모든 이미지가 보유하고 있는 객체의 수는 <표 Ⅲ-2>와 같다.

<표 Ⅲ-2> 학습에 이용된 객체의 수

균열 종류	AC	LC	TC	PAT/POTHOLE
Object 수	129,068	1,089,806	297,270	494,793

수집된 데이터셋을 모두 이용하여 SSD 모델의 학습을 진행하였다. 학습과정에서 Loss 값이 6 < Loss < 7 의 값을 벗어나지 못하는 현상이 있어 학습을 중단하고 엣지 단말에서의 모델 실행 전에 서버 상에서 객체탐지의 테스트와 성능평가를 진행해보았다. <그림 Ⅲ-9>는 임의의 테스트 이미지에 대한 균열 탐지 결과를 나타낸다.



<그림 Ⅲ-9> 학습한 모델의 테스트 결과

<표 Ⅲ-3>는 학습된 모델의 성능 평가표를 나타낸다. 객체 탐지 모델의 성능 지표인 mAP 값이 58.8% 정도로 높은 성능을 보이지는 않았다. 이에 학습 데이터 중 LC(종방향균열)의 객체의 수가 너무 많아 모델이 LC 유형의 균열 탐지에 치우쳐 학습되고 있는 것이 아닌지에 대한 확인이 필요하여 데이터셋을 보정한 후 재학습을 진행하였다.

<표 Ⅲ-3> 모델의 성능 평가

균열 종류	AC	LC	TC	PAT/POTHOLE	mAP
AP	58.3%	60.2%	52.6%	64.1%	58.8%

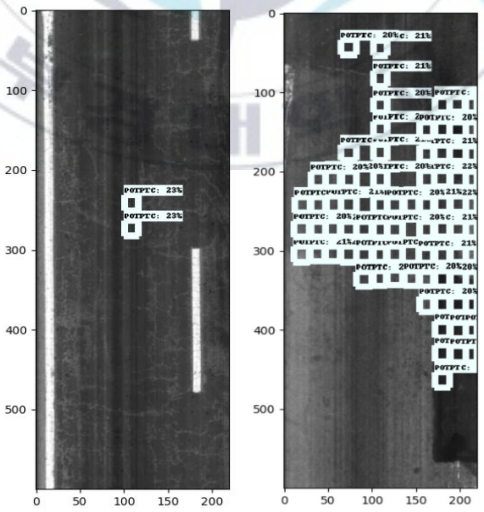
(2) 보정한 데이터셋을 이용한 학습

LC 유형의 균열에 치우친 학습을 예방하기 위해 데이터셋의 보정작업을 진행하였으며, 객체의 수가 가장 적은 AC(거북등균열) 유형의 객체 수에 맞추어 모든 균열 유형에서 객체의 수를 임의 추출한 129,000로 통일하였다. <표 Ⅲ-4>는 보정한 데이터셋에서 객체의 수를 나타낸다.

<표 Ⅲ-4> 보정한 데이터셋에서의 균열 유형별 객체의 수

균열 종류	AC	LC	TC	PAT/POTHOLE
Object 수	129,000	129,000	129,000	129,000

모든 균열 유형에서 객체의 수가 통일된 학습데이터를 통해 같은 모델의 학습을 진행하였다. 학습 진행과정에서 4 < Loss < 5 의 Loss 값을 유지하였으며 더 이상의 학습은 무의미하다고 판단하여 학습을 종료한 후 이전 학습한 모델의 테스트에 사용하였던 같은 이미지에 대해 테스트를 진행해 보았으며 그 결과는 <그림 Ⅲ-10>과 같다.



<그림 Ⅲ-10> 수정된 데이터로 학습한 모델의 테스트 결과

마찬가지로 같은 방법을 통해 모델의 성능 평가를 진행해 보았으며 그 결과는 <표 Ⅲ-5>와 같다.

<표 Ⅲ-5> 보정한 데이터셋으로 학습한 모델의 성능 평가

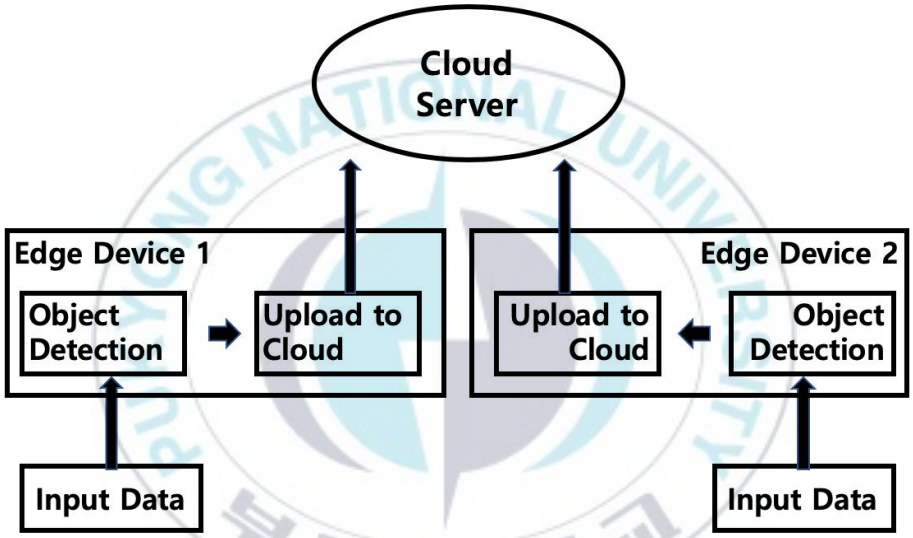
균열 종류	AC	LC	TC	PAT/POTHOLE	mAP
AP	59.6%	62.9%	53.9%	68.8%	61.3%

새로 학습한 모델이 객체를 더 많이 검출해 내었으나, 탐지 성능은 분류 모델에 비해 상당히 낮은 수준에 머물러 있기 때문에 균열 탐지 모델의 성능을 끌어올리는 후속적인 연구가 지속되어야할 것으로 보인다. 학습을 진행하는 PC와 모델을 실제로 실행할 엣지 단말의 성능이 보장된다면 Base Network로써 VGG 계열의 네트워크나 더 깊은 네트워크를 사용하고 입력으로 300*300 크기의 이미지를 받는 SSD300 모델을 사용하는 것 대신에 500*500 크기의 입력 이미지를 받는 SSD500 모델을 사용한다면 더 높은 성능을 얻을 수 있을 것으로 판단된다.

Ⅵ. 도로 균열 탐지를 위한 모바일 시스템

1. 파일럿 시스템 구조

본 연구에서는 실험을 위해 엣지 컴퓨팅 환경의 파일럿 시스템을 구축하였다. <그림 Ⅳ-1>은 실험을 위한 파일럿 시스템의 구조도를 나타낸다.



<그림 Ⅳ-1> 엣지 컴퓨팅 환경 구조의 파일럿 시스템

엣지 컴퓨팅 환경의 정의와 구조에 적합하도록 엣지 단말을 사용하고 클라우드 서버를 사용하여 파일럿 시스템을 구축하였다.

엣지 단말에서는 균열 탐지가 필요한 도로의 표면 이미지를 입력받아 모델을 수행하여 데이터를 분석 처리하는 역할을 수행하게 되며 처리된 결과는 네트워크와 인터넷을 통해서 클라우드 서버로 전송되게 된다. 사용자는 처리된 결과 즉, 균열이 탐지된 결과를 클라우드 서버에서 확인할 수 있게

된다. 엣지 컴퓨팅 환경에 맞게 여러 개의 엣지 단말에 대해 시스템이 원활하게 동작하여야 한다.

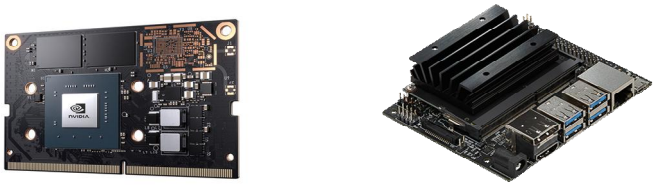
2. 구축 환경 및 기능 구현

엣지 컴퓨팅 환경의 시스템 구축에 있어서 사용되는 대표적인 요소들에 대해 구축환경과 기능에 대해 정리하였다. 본 연구에서는 엣지 단말과 해당 기기에서 처리한 결과 데이터를 전송받아 클라우드 서버 역할을 하는 웹 서비스가 시스템의 요소로 볼 수 있다.

가. 엣지 단말(Edge Device)

본 연구에서 실험을 위해 사용되는 엣지 단말은 NVIDIA 사에서 AI, 딥러닝 용으로 출시된 소형 컴퓨터인 젯슨 나노(Jetson Nano)이다. 젯슨 나노의 개발자 킷(Developer Kit)을 통해 파일럿 시스템을 구성하였다.

<그림 Ⅳ-2>는 젯슨 나노 보드의 형태에 대해 나타낸다. 젯슨 나노는 1.43GHz의 쿼드 코어 ARM A57 CPU와 딥러닝에 특화된 소형 컴퓨터인 만큼 128 코어의 Nvidia Maxwell GPU를 장착하고 있으며 4GB의 RAM를 보유하고 있다. 이를 통해 젯슨 나노는 소형 컴퓨터임에도 불구하고 많은 딥러닝 모델을 엣지 컴퓨팅 환경으로 테스트해 볼 수 있다.



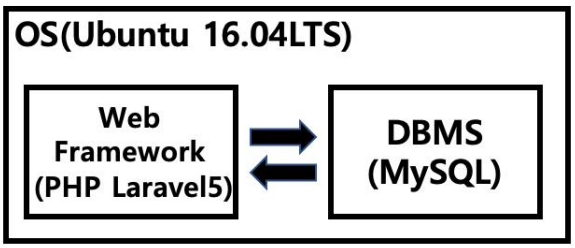
<그림 Ⅳ-2> Jetson Nano 보드의 형태

파일럿 시스템의 구축을 위해 젯슨 나노에 Ubuntu 18.04LTS 버전의 OS를 탑재하였고 본 연구를 통해 학습한 딥러닝 모델을 이식하여 엣지 단말로서의 역할을 적절히 수행할 수 있도록 하였다.

파일럿 시스템 상에서 엣지 단말은 우선 균열 탐지가 필요한 도로의 표면 이미지들을 입력받게 되고 해당 이미지들을 사전 학습되어 단말에 이식되어 있던 객체탐지 기반의 균열 탐지 모델인 MobilenetV2를 활용한 SSD에 입력 값으로 전달하게 된다. 입력을 받은 균열 탐지 모델은 해당 입력 이미지를 모델에 통과시켜 균열을 탐지하게 된다. 모델 통과가 끝나면 균열 탐지 결과를 시스템의 클라우드 서버로 전송하면서 엣지 단말의 역할은 끝이난다.

나. 클라우드 서버(Cloud Server)

본 연구의 실험을 위한 파일럿 시스템에서 클라우드 서버로서의 역할을 수행하는 웹 서버를 구축하였다. 웹 서버는 웹 개발 언어인 PHP 기반의 웹 개발 프레임워크인 Laravel 5를 이용하여 구축하였으며 MySQL 데이터베이스를 사용하였다. 서버의 OS는 Ubuntu 16.04LTS 버전을 탑재하였다. 그림 Ⅳ-3은 클라우드 서버의 환경과 구조에 대해 나타낸다.

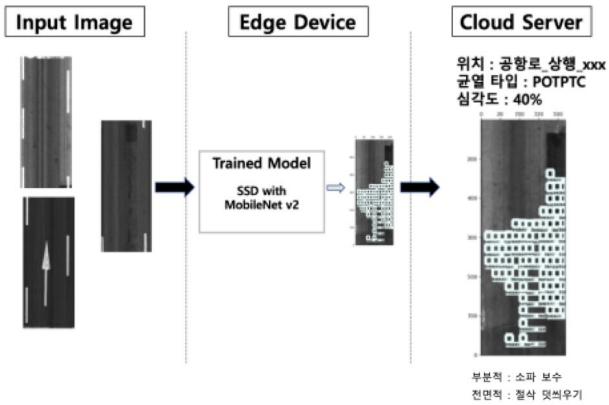


<그림 Ⅳ-3> 클라우드 웹 서버 구축 환경

파일럿 시스템의 엣지 단말인 젯슨 나노에서 결과 데이터를 클라우드 서버로 전송하게 되면, 클라우드 서버는 전달받은 결과 데이터를 바탕으로 사용자에게 균열 탐지의 결과를 제공하는 역할을 한다. 서버는 전달받은 균열 탐지 결과 데이터를 바탕으로 사용자가 유지보수에 대한 판단을 내리는데 있어서 도움을 줄 수 있도록 균열 탐지의 최종 결과를 생성하게 된다.

3. 테스트 운용 결과 및 활용 방안

엣지 컴퓨팅 환경의 파일럿 시스템 구축을 통한 테스트를 진행하여보았다. 시스템의 운용 테스트는 테스트용으로 필요한 도로의 표면 이미지와 균열 탐지 모델을 수행할 수 있는 엣지 단말인 젯슨 나노 그리고 클라우드 서버 역할을 수행하는 웹 서버를 모두 구축한 후 테스트를 진행하였다. <그림 Ⅳ-4>는 샘플 입력 이미지에 대한 시스템 운용 테스트 과정을 나타낸다.



<그림 Ⅳ-4> 특정 샘플 이미지에 대한 시스템 운용 과정

장비에 대한 제약으로 엣지 단말에 카메라를 설치하여 현장에서의 테스트를 진행해보지는 못하였으나 사전에 캡쳐된 이미지를 사용하여 파일럿 시스템이 잘 작동하였으며 입력이미지에 대한 균열 탐지 결과를 적절히 출력해 내는 것을 확인할 수 있었다. 엣지 단말인 젯슨 나노에 카메라를 장착하고 와이파이나 셀룰러 데이터를 통하여 실제 현장에서의 작업을 수행한다면 하나의 시스템으로서 도로 균열 탐지 분야에서 더욱 의미 있는 역할을 수행할 수 있을 것이다.

본 연구에서 제안한 엣지 시스템 환경에서의 자동화된 도로 균열 탐지 시스템 구축과 관련된 기술의 사업화 방안 및 활용 방안에 대해 모색해 보았다. 먼저 엣지 시스템 환경에서의 균열 탐지 시스템 구축 방법에 대한 특허권의 확보와 함께 현업에서 포장관리를 실시하는 기업에 실시권을 부여할 수 있다. 해당 특허권의 확보를 위해서는 균열 탐지 모델의 지속적인 연구를 통해 모델자체의 독립성을 확보하여야 한다.

본 논문에서 제시한 시스템의 사용을 통해 포장관리 업체의 경쟁력을 확보할 수 있다는 점도 고려해 볼 수 있다. 매년 지자체에서는 도로의 포장

관리를 위해 기업을 선정하게 되는데, 본 시스템의 프로토타입 제작을 통한 기술력을 바탕으로 포장관리 사업의 기업선정에서 경쟁력을 확보할 수 있다. 기업은 엣지 단말을 통한 균열 탐지작업을 진행하면서 고가의 로드 스캐너를 장비한 차량의 추가에 대한 부담을 경감할 수 있으며 이로써 경제적인 경쟁력 또한 확보할 수 있을 것이다.

각 지자체는 본 시스템을 운영함으로써 축적된 데이터를 이용하여 각 도로 구간별 통계적 데이터를 얻을 수 있다. 이를 활용하면 도로 포장관리의 예산 관리 척도로써 본 시스템을 사용할 수 있다. 시간의 경과에 따라서 도로의 균열이 어떻게 변형되어 가는지를 파악하고 다음 포장관리 기간의 효율적인 예산 확보 계획을 수립할 수 있을 것이다.

Ⅴ. 결론

1. 연구 요약

본 논문에서는 자동차 도로 균열 탐지 작업을 자동화하고 인력과 시간적인 비용을 절약하고자 하는 점에서 엣지 단말을 이용한 모바일 환경에서의 도로 균열 탐지 시스템 구축에 관한 연구를 진행하였다. 먼저 도로 균열 탐지를 위한 모델 학습을 위해 적절한 모델을 선택하는 과정을 거쳤으며 엣지 단말의 제한적인 계산성능에서 수행할 수 있도록 딥러닝의 객체탐지 분야에서 MobileNetV2 기반의 SSD 모델을 선택하여 학습을 진행하였다. 두 가지의 데이터셋을 통해 학습을 수행해 보았다. 먼저 모든 데이터를 이용한 학습에서는 mAP 값이 58.8% 정도로 높은 성능을 보이지는 않았다. 두 번째로는 데이터셋을 보정하여 각 균열 종류별 객체의 수를 129,000개로 통일한 후 학습을 재시도 해보았다. 두 번째 학습에서는 mAP 값이 61.3%로 앞서 학습한 첫 번째 모델보다는 높은 성능을 보였으나 매우 높은 성능을 보이지는 않았다. 현재까지도 분류 모델에 비해 객체탐지 모델은 높은 성능을 끌어내야만 하는 과제가 존재하기 때문에 전체 시스템을 위한 균열 탐지 모델에 있어서 성능 향상에 대한 연구가 지속적으로 필요할 것으로 판단된다.

학습된 모델의 실험을 위해 엣지 컴퓨팅 환경의 파일럿 시스템을 구축하고자 하였으며 적절한 엣지 단말을 선택하고 클라우드 서버 역할의 웹 서버를 환경에 맞게 구축하여 파일럿 시스템을 완성하였다. 샘플 테스트 이미지를 이용하여 파일럿 시스템의 운용 테스트를 진행하였으며 적절한 결과 값의 출력을 확인할 수 있었다. 엣지 단말에 이미지를 입력하면 내장하고 있는 균열 탐지 모델을 이용해 해당 입력이미지에서 균열 탐지를 수행

하게 되고 해당 탐지 결과를 클라우드 서버로 전송하여 관리자에게 균열 탐지에 대한 결과를 제공한다. 따라서 현장에서 취득된 도로 이미지를 엣지 단말 상에서 딥러닝 모델을 통해 즉각적으로 분석하여 도로 균열을 탐지하고 유지보수 계획의 수립을 보조할 수 있을 것이다.

엣지 컴퓨팅 환경에서의 균열 탐지 작업을 시도함으로써 클라우드 서버 측의 분류 작업자의 부담을 경감할 수 있을 것이다. 최종적으로 균열 탐지를 위한 객체 탐지 모델의 성능에 있어서는 다소 실험적인 부분이 존재하지만, 장기적인 목표를 두고 파일럿 시스템을 구현하여 엣지 시스템 환경에서의 실시간 균열 탐지의 가능성을 타진해 보았다.

2. 연구의 한계점과 향후 연구 방향

카메라나 네트워크에 필요한 장비들이 모두 갖추어진 상황에서 파일럿 시스템 상에서의 테스트가 아니라 엣지 단말에 카메라 모듈과 각종 센서 그리고 셀룰러 네트워크 장비를 적용시켜 실제 현장에서의 테스트를 진행해 볼 수 있을 것이다. 이를 통해 시스템을 현업에서 운용할 시 더 필요한 개선사항에 대해 모색할 수 있고 시스템 상에서 불필요한 요소들을 제외시킬 수 있을 것이다.

학습용 장비와 엣지 단말의 성능 측면의 한계에 도달하여 균열 탐지 모델의 성능이 아쉬운 결과를 보였다. 향후 엣지 단말의 성능을 보완하면 더 무겁고 높은 성능의 딥러닝 모델의 학습이 가능할 것이고 이는 곧 균열 탐지에 있어서 성능을 끌어 올리는 효과를 가져 올 것이다. 균열 탐지 성능의 향상은 곧 엣지 컴퓨팅 환경에서의 균열 탐지 시스템 전체의 성능을 높이는 것을 의미한다고 볼 수 있어 클라우드 서버의 효율적인 설계와 함께

균열 탐지의 성능을 높이는 것이 향후 첫 번째 과제가 되어야 할 것이다.

또한 낮은 품질의 이미지 데이터에 대한 해결책을 모색하여 데이터의 품질을 끌어올림으로써 모델의 성능을 향상시킬 수 있는 방법도 존재한다. 모든 방법을 효율적으로 구성하여 균열 탐지의 성능을 최대한 끌어올리는 연구가 지속되어야 한다.

참고 문헌

1. 국내 문헌

국토교통부. (2013). 도로 포장 유지보수 실무 편람.

국토교통부. (2019). 2019년 예산 개요.

2. 해외 문헌

Bengio, Y., Courville, A., and Vincent, P., “Representation learning: A review and new perspectives,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 35, pp. 1798 - 1828, March 2013.

Cheng, H. D., Shi, X. J., & Glazier, C. (2003). Real-time image thresholding based on sample space reduction and interpolation approach. Journal of computing in civil engineering, 17(4), 264-272.

Cord, A., & Chambon, S. (2012). Automatic road defect detection by textural pattern recognition based on AdaBoost. Computer Aided Civil and Infrastructure Engineering, 27(4), 244-259.

D. O. Hebb. (1949). “The Organization of Behavior”, New York : Wiley & Sons.

Gartner, Inc. (2017). Top 10 Strategic Technology Trends for 2018

Gartner, Inc. (2018). Top 10 Strategic Technology Trends for 2019

Gartner, Inc. (2019). Top 10 Strategic Technology Trends for 2020

Girshick, R. (2015). Fast r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 580-587).

Gopalakrishnan, K., Khaitan, S. K., Choudhary, A., & Agrawal, A. (2017). Deep Convolutional Neural Networks with transfer learning for computer vision-based data-driven pavement distress detection. Construction and Building Materials, 157, 322-330.

Haas, C., Hendrickson, C., & McNeil, S. (1991). A Design for Automated Pavement Crack Sealing. In Preparing for Construction in the 21st Century (pp. 222-227). ASCE.

Hao, Z., Yi, S., & Li, Q. (2018). Edgecons: Achieving efficient consensus in edge computing networks. In {USENIX} Workshop on Hot Topics in Edge Computing (HotEdge 18).

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE transactions on pattern analysis and machine intelligence, 37(9), 1904-1916.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

https://github.com/tensorflow/models/tree/master/research/object_detection

https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/?nvid=nv-int-mn-78458

Hu, P., Dhelim, S., Ning, H., & Qiu, T. (2017). Survey on fog computing:architecture, key technologies, applications and open issues. Journal ofNetwork and Computer Applications, 98, 27-42.

J. Schmidhuber, “Deep learning in neural networks: An overview,” Neural Netw vol. 61, pp. 85 - 117, Jan 2015.

Jahanshahi, M. R., & Masri, S. F. (2012). Adaptive vision-based crack detection using 3D scene reconstruction for condition assessment of structures. Automation in Construction, 22, 567-576.

L. Deng, J. Li, J. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams, Y. Gong, and A. Acero. “Recent advances in deep learning for speech research at Microsoft,” ICASSP, vol. 26, pp. 64, May 2013.

LeCun et al. (1989). Backpropagation Applied to Handwritten Zip Code Recognition,Neural Computation, 1: 541 - 551.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

Li, S., Cao, Y., & Cai, H. (2017). Automatic pavement-crack detection and segmentation based on steerable matched filtering and an active contour model. Journal of Computing in Civil Engineering, 31(5), 04017045.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016, October). Ssd: Single shot multibox detector. In European conference on computer vision (pp. 21-37). Springer, Cham.

Maeda, H., Sekimoto, Y., Seto, T., Kashiyama, T., & Omata, H. (2018). Road damage detection using deep neural networks with images captured through a smartphone. arXiv preprint arXiv:1801.09454.

Oliveira, H., & Correia, P. L. (2012). Automatic road crack detection and characterization. IEEE Transactions on Intelligent Transportation Systems, 14(1), 155-168.

Radopoulou, S. C., & Brilakis, I. (2015). Patch detection for pavement assessment. Automation in Construction, 53, 95-104.

Radopoulou, S. C., & Brilakis, I. (2016). Automated detection of multiple pavement defects. Journal of Computing in Civil Engineering, 31(2), 04016057.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only

look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems (pp. 91-99).

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review, 65(6), 386.

S. J. Russell, P. Norvig. (2003). “Artificial Intelligence : A Modern Approach”, 2nd ed., Prentice Hall)

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 4510-4520).

Sareen, P. (2013). Cloud computing: types, architecture, applications, concerns, virtualization and role of it governance in cloud. International Journal of Advanced Research in Computer Science and Software Engineering, 3(3).

Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. IEEE Internet of Things Journal, 3(5), 637-646.

Shi, Y., Cui, L., Qi, Z., Meng, F., & Chen, Z. (2016). Automatic road

crack detection using random structured forests. IEEE Transactions on Intelligent Transportation Systems, 17(12), 3434-3445.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Stergiou, C., Psannis, K. E., Kim, B. G., & Gupta, B. (2018). Secureintegration of IoT and cloud computing. Future Generation Computer Systems,78, 964-975.

Sun, L., Kamaliardakani, M., & Zhang, Y. (2015). Weighted neighborhood pixels segmentation method for automated detection of cracks on pavement surface images. Journal of Computing in Civil Engineering, 30(2), 04015021.

Uijlings, J. R., Van De Sande, K. E., Gevers, T., & Smeulders, A. W. (2013). Selective search for object recognition. International journal of computer vision, 104(2), 154-171.

Wang, K. C., Zhang, A., Li, J. Q., Fei, Y., Chen, C., & Li, B. (2017, August). Deep learning for asphalt pavement cracking recognition using convolutional neural network. In Proc. Int. Conf. Airfield Highway Pavements (pp. 166-177).

Wang, X., & Hu, Z. (2017, August). Grid-based pavement crack analysis using deep learning. In 2017 4th International Conference on Transportation Information and Safety (ICTIS) (pp. 917-924). IEEE.

Yu, W., Liang, F., He, X., Hatcher, W. G., Lu, C., Lin, J., & Yang, X. (2018).A survey on the edge computing for the Internet of Things. IEEE access, 6,6900-6919.

Zhang, L., Yang, F., Zhang, Y. D., & Zhu, Y. J. (2016, September). Road crack detection using deep convolutional neural network. In 2016 IEEE international conference on image processing (ICIP) (pp. 3708-3712). IEEE.

Zitnick, C. L., & Dollár, P. (2014, September). Edge boxes: Locating object proposals from edges. In European conference on computer vision (pp. 391-405). Springer, Cham.

Zou, Q., Cao, Y., Li, Q., Mao, Q., & Wang, S. (2012). CrackTree: Automatic crack detection from pavement images. Pattern Recognition Letters, 33(3), 227-238.

감사의 글

본 석사 학위 논문을 작성하는데 주변의 많은 지원과 독려와 가르침이 있었습니다. 바쁘신 가운데에도 학부생 시절부터 석사과정의 끝 무렵까지 항상 배려와 격려로 지도하여 공학도로서 좋은 방향을 제시해주신 김민수 지도교수님께 먼저 진심으로 감사를 드립니다.

연구실에서 공부하며 많은 점을 느끼며 성장할 수 있는 기회를 마련해 주신 점 또한 진심으로 감사드립니다.

그리고 석사 과정의 끝을 본 논문으로 완성할 수 있도록 부족한 점을 지도해 주시고 더 좋은 방향을 조언해주며 논문의 심사를 맡아주신 이지환 교수님, 옥영석 교수님 그리고 천동필 교수님께도 감사의 말씀드립니다.