공학석사 학위논문

라즈베리파이 기반 SSD를 이용한 화재 검출 시스템 구현

Implementation of Fire Detection System Using Raspberry Pi-based SSD

지도교수 손 경 락

2020 년 2 월

한국해양대학교 대학원

전자통신공학과

양 승 호

본 논문을 양승호의 공학석사 학위논문으로 인준함.

위원장 김 정 창 (인)

위 원 손 경 락 (인)

위 원 고 승 우 (인)

2019년 12월 20일

한국해양대학교 대학원

목 차

List of Tables

List of Figures

라즈베리파이 기반 SSD를 이용한 화재 검출 시스템 구현

Yang, Seung Ho

Division of Electronics and Communications Engineering
Graduate School of Korea Maritime and Ocean University

Abstract

최근 국내와 국외에 화재가 연달아 일어나 이슈가 되었다. 사람이 자주 찾지 않는 지역은 화재가 일어날 경우 초기 대응까지 긴 시간이 걸리기 때문에 대형 화재로 커지는 것을 막지 못하게 된다. 때문에 화재가 대형으로 커지기 전에 감지하는 것에 대해 관심을 가지게 되었다. 본 학위 논문에서는 넓은 지역의 화재감지 시스템을 구성하기 위하여 라즈베리파이와 딥러닝 기술인 객체감지와 전력선 통신 기술을 사용하였다. 검출속도가 느린 F-RCNN과 빠른 SSD을 사용하여 학습 모델을 형성하여 라즈베리파이에서 더 적합한 모델인지를 검출속도를 비교하였다. 또한 시스템 구성에 사용된 전력선 통신은 고전압과 고전류 환경에서도 통신이 잘 이루어지는 가를 확인하기로 하였다. 마지막으로 라즈베리파이에서 파이카메라를 통해 화재를 감지하면 유도형 전력선 통신 기술을 사용하여 모니터링 PC로 화재가 일어났음을 알리는 텍스트와 이미지를 전송하는 실험에 성공하였다.

KEY WORDS: Image Deep learning 이미지 딥러닝; Object detection 객체검출; Power line communication 전력선 통신

Implementation of Fire Detection System Using Raspberry Pi-based SSD

Yang, Seung Ho

Division of Electronics and Communications Engineering
Graduate School of Korea Maritime and Ocean University

Abstract

The recent domestic and foreign fires have caused issues. Areas that are rarely visited do not prevent large fires because they take a long time to respond early in the event of a fire. Because of this, we became interested in detecting fire before it became large. In this thesis, raspberry pi, deep learning technology and power line communication technology are used to construct a fire detection system in a large area. A training model was formed using F-RCNN and SSD to compare the detection speed to see if it is a more suitable model in Raspberry Pi. In addition, the power line communication used in the system configuration is to confirm whether the communication is well in high voltage and high current environment. Finally, when Raspberry Pi detects a fire through the Pi camera, it succeeded in sending texts and images to the monitoring PC using inductive power line communication technology.

KEY WORDS: Image Deep learning; Object detection; Power line communication

제 1 장 서 론

1.1 연구 배경 및 연구 내용

오랜 세월 인간은 화재를 막기 위한 노력을 계속해왔고 지금도 계속되고 있다. 특히 최근 국내와 국외에 화재가 연달아 일어나 이슈가 되었다. Fig. 1은 최근 10년 간 국내에서 일어난 화재 피해 건수와 면적이다. 일시적으로 피해 건수 대비 피해 면적이 줄어들지만 다시 건수와 피해 면적이 늘어나는 것을 확인 할 수 있다. 때문에 화재가 대형으로 커지기 전에 감지하는 것에 대해 관심을 가지게 되었다. 사람이 자주 찾지 않는 지역은 화재가 일어날 경우 초기 대응까지 긴 시간이 걸리기 때문에 조기 진화는 물론이고 대형 화재로 커지는 것을 막지 못하게 된다. 특히 산간 지역과 같은 넓은 지역에서 이와 같은 문제가 자주 보이기 때문에 본 논문에서는 화재 감지를 위하여 딥러닝의 객체검출 기술과 전력선 통신 기술을 사용한 화재 검출 시스템을 구현하였다.

Fig. 1. Forest Fire Damage for 10 Years [1].

딥러닝 기술이 발전하기 이전까지는 주로 센서에 의한 화재 감지 기술 (Fire detection technology)이 주로 연구되어 왔다. 팬 / 틸트 작동이 가능한 카메라의 파노라마 이미지 [2], MODIS (Moderate Resolution Imaging Spectroradiometer) 위성 이미지 [3], IoT (Internet of Things) 기반 드론 [4], 무인 헬리콥터 및 센서 네트워크 [5]를 이용한 산불 모니터링 시스템이 발표되었다. 하지만 앞서 설명한 방법은 대개 센서나 이미지에서 온도를 계산하는 (위성이미지의 경우 지표 온도를 계산한다.) 방식을 사용하기 때문에 비가 오거나 날씨가 흐린 경우 화재를 찾지 못하는 경우가 발생한다.

하드웨어의 발전과 여러 기술의 발달로 4 차 산업 혁명이 일어난 덕분에 딥러닝이 대두되게 된 이후에는 딥러닝을 활용한 화재 감지에 대한 연구가 주를 이루고 있다 [6]. 최근에는 합성곱 신경망 (CNN, Convolutional Neural Network)을 이용한 화재 및 화염 탐지에 관한 연구가 도입되었다. [7-9]. 하지만 대부분의 화재 감지를 위한 연구에서는 실내 또는 건물 근처에서 감지되었다. 또한 위성이나 드론 헬리콥터를 사용할 경우 매우 큰 비용이 요구되기 때문에 조금 더 저렴한 시스템이 필요하다고 생각되었다. 산간 지역과 같은 매우 넓은 지역에서 딥러닝 기반 화재 감지를 위한 시스템이 필요하다.

딥러닝이란 인공 신경망을 이용한 머신러닝 일종으로, 컴퓨터들이 인간의 두뇌와 비슷한 모양의 대형 인공 신경망을 구성하여 복수의 레이어를 통해 학습을 진행하는 기법을 말한다. 과거와 달리 딥러닝이 가능해진 이유로는 먼저 기존 인공신경망 모델의 단점이 극복되었다는 점이다. 과적합 (Over fitting) 문제와 같은 구조적 한계들이 교차 검증이나 드롭아웃 (dropout), 정규화 (regularization)등과 같은 여러 기법이 추가되면서 해결되었다. 그리고 GPU (Graphics Processing Unit)라는 복잡한 3D 그래픽 처리에 적합한 뛰어난 연산 능력과 병렬 처리 기술을 가진 하드웨어의 발달에 의해 학습 시간이 크게 줄어들게 되었다. 마지막으로 정보화 시대가 진행함에 따라 빅 데이터라 불리는 대량의 데이터가 생기게 되고 SNS 사용자들에 의해 이런 자료들이 정리되어 학습에 사용되었기 때문이다. 딥러닝의 부활 이후 다양한 분야, 특히 자동 음성 인식과 컴퓨터비전 분야에서 최고 수준의 성능을 보여주고 있으며 최근에는

CNN (Convolutional Neural Network:합성곱 신경)망 기반의 딥러닝 알고리즘이 뛰어난 성능을 발휘하고 있다 [10]. 또한 이러한 딥러닝의 발달은 이미지에서 물체를 찾아내는 객체검출 기술의 발달도 함께 이루어져 아래의 Fig. 2와 같이 개와 고양이의 데이터를 학습 시키는 것으로 두 동물을 구별해내는 것이 가능하게 되었다. 사진에 라벨을 붙여 놓으면 각각의 이미지에서 스스로 특징을 추출하여 이미지 내에서 객체를 찾아낼 수 있게 된 것이다. 이를 이용하여 화재 또한 검출 할 수 있을 것이라 기대할 수 있다.

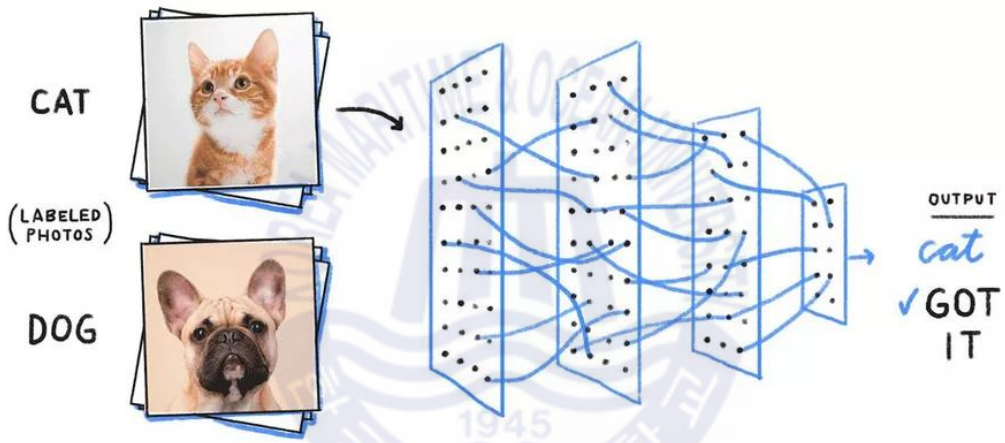


Fig. 2. Deep Learning Overview [11].

본 논문에서는 딥러닝 진행 상황을 보다 명확히 알기 위하여 손실함수(Loss function) 만이 아니라 IOU (Intersection Over Union) 또한 사용하여 정확도를 판단하기로 하였다. IOU는 교차 영역의 넓이를 이용하여 정확도를 평가하는 지표로서 Fig. 3과 같이 딥러닝 객체 검출에서는 예측된 경계 상자 (predicted bounding boxes)와 사물의 실제 위치를 나타내는 실제 참값 (ground truth) 경계 상자의 교차영역의 넓이를 합 영역의 값으로 나눈 값을 이용해 '정확도' 를 나타낸다 [12, 13].

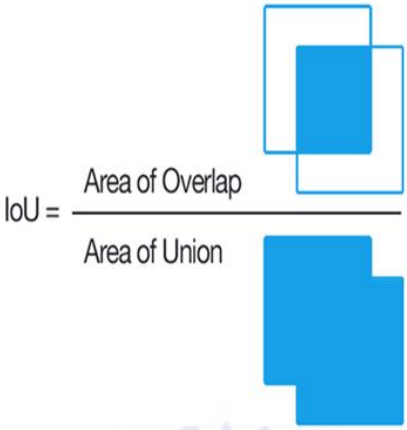


Fig. 3. IOU (Intersection Over Union).

본 논문에서는 넓은 지역의 화재감지 시스템을 구성하기 위하여 라즈베리파이와 딥러닝 기술인 객체감지와 전력선 통신 기술을 사용하였다. 객체 감지를 사용하기 위하여 텐서플로우라는 오픈소스 라이브러리의 딥러닝 모듈을 사용하여 학습을 진행하였고 학습 데이터를 얻기 위하여 인터넷에서 화재 동영상을 얻어 동영상을 일정 간격으로 분할하여 라벨링하는 방식으로 다양한 화재 데이터를 모았다. 라벨링 과정에서 생성되는 xml 파일을 텐서플로우에 맞춰 tfrecord 파일로 변형하였으며 학습이 완료된 모델은 freeze 파일로 만들어 쉽게 적용할 수 있도록 하였다.

학습에 사용된 모델은 성능 차이로 인해 자주 비교되고는 하는 사진을 분할하기 때문에 작업처리량이 많아 검출속도가 느린 F-RCNN (Fast-RCNN, Fast Region-based Convolutional Neural Network)과 사진을 분할하지 않아 검출속도가 빠른 SSD (Single Shot Multibox Detector)을 사용하여 학습 모델을 형성하였다. 소형컴퓨터인 라즈베리파이에서 어느 쪽이 더 적합한 모델인지 검출속도를 비교하여 F-RCNN의 경우 0.05 fps의 검출속도를 가지는데 반해 SSD는 1.88가량으로 수십 배나 빠른 속도를 갖는 것을 확인 하였다.

또한 시스템 구성에 사용된 전력선 통신은 건물 내부 배선은 물론 송전선이나 용접선과 같은 매우 높은 고전압 환경에서도 전력선 통신 기술이 적용 되어야한다. 전력선 통신 기술 (PLC : Power-Line Communications)은, 전력을 공급하는 전력선을 이용해서 음성과 데이터를 수백 kHz에서 수십 kHz 이상의 고주파 신호에 실어 통신 하는 기술이다. 주변에 광범위하게 설치되어 있는 전력선을 통신 매체로서 이용하는 통신 방식으로서, 단순 신호 전달 및 원격 제어에서부터 초고속 데이터 네트워크에 이르기까지 여러 응용분야에 활용될 수 있는 유선 통신 기술이다 [14]. 일상적으로 사용하고 있는 전기에너지는, 대개 전력선이라는 매체를 통해 한 곳에서 다른 한 곳으로 전달된다. 즉, 전기에너지를 사용하고 있다면, 광범위한 전력선 인프라를 갖추고 있다고 볼 수 있다. 이러한 전력선을 정보 전달에도 활용하고자 하는 것이 전력선통신의 기본 개념이다 [15]. 이러한 전력선 통신 기술은 기존 인프라에 적용이 가능하기 때문에 시스템을 구성하는데 매우 저렴하고 효율적일 것이라 기대된다.

때문에 시스템 구성에 포함하고자 고전압과 고전류 환경에서도 통신이 잘 이루어지는가를 확인하였다. 통신 성능을 테스트하기 위해 Jperf 프로그램이 사용되었고 실험 결과 전류의 유무 자체에는 영향을 받지만 전류의 크기에 크게 영향을 받지 않는 것을 확인 할 수 있었다.

마지막으로 구현한 화재 검출 시스템을 테스트한 결과 라즈베리파이에서 파이카메라를 통해 화재를 감지하면 유도형 전력선 통신 기술을 사용하여 모니터링 PC로 화재가 일어났음을 알리는 텍스트와 이미지를 전송하는 실험을 진행하였다. 테스트 결과 일반적인 전선에서 뿐만 아니라 용접중의 고전류 환경에서도 텍스트와 이미지를 전송하는데 성공하였다.

1.2 논문 구성

본 논문의 구성은 다음과 같다. 제 1 장 서론에 이어 제 2 장에서는 CNN의 신경망 구조에 대해 알아보도록 하고 실험에 사용된 학습 기법 F-RCNN과 SSD에 대해 소개한다. 또한 하이퍼파라미터에 대해 간단히 소개한다. 제 3 장 1절에서는 시스템 구성을 위해 준비된 학습 환경에 대해 알아본다. 화재 데이터를 학습에 사용하는 방법을 설명하고 두 학습 기법을 사용하였을 때의 학습 결과를 비교한다. 2절에서는 학습된 모델을 소형컴퓨터에 사용하여 검출속도를 비교하였다. 제 4 장 1절은 전력선 통신이란 무엇인가에 대해 알아본다. 또한 사용된 커플러와 전력선 통신 모뎀을 소개한다. 또한 PLC를 사용하여 통신 성능을 테스트한 결과를 알아본다. 2절에서는 전력선 통신 기술을 사용하여 화재감지 시스템을 테스트한 결과를 보인다. 일반적인 송전선에서와 용접용 고전압 케이블에서 진행하였다. 마지막으로 제 5 장은 결론으로 본 논문에서 소개한 것들을 정리한다.

제 2 장 객체 검출 기법

합성곱 신경망 이 나오기 이전의 풀리 커넥티드 (Fully Connected) 레이어만으로 구성된 인공 신경망의 입력 데이터는 1차원 형태로 한정된다. 이는 3차원 사진 데이터를 사용할 경우 1차원으로 평면화 시켜야하기 때문에 사진 데이터를 평면화 시키는 과정에서 공간 정보가 손실될 수밖에 없다. 이미지 공간 정보 유실로 인한 정보 부족으로 인공 신경망이 특징을 추출 및 학습하는데 비효율적이고 정확도를 높이는데 한계가 있다. 때문에 이미지의 공간 정보를 유지한 상태로 학습이 가능한 모델이 바로 합성곱 신경망이라고도 불리는 CNN이다.

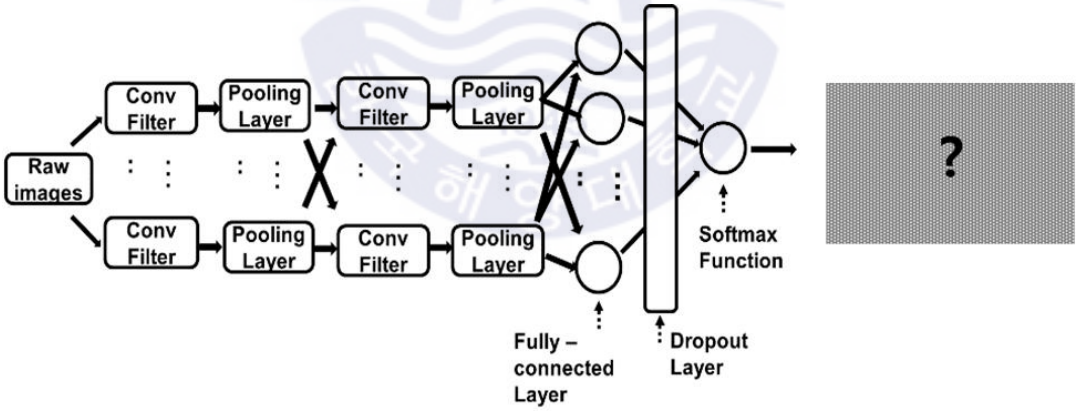


Fig. 4. Basic CNN Structure.

Fig. 4는 합성곱 계층 및 풀링 계층과 같은 여러 계층을 통해 데이터를 추출하는 CNN 의 기본 구조이다. CNN 구조는 이미지의 특징을 추출하는 부분과 클래스를 분류하는 부분으로 나눌 수 있다. 특징 추출 영역은 합성곱 레이어와

풀링레이어를 여러 겹 쌓는 형태로 구성된다. 컨볼루션 레이어는 입력 데이터에 필터를 적용 후 활성화 함수를 반영하는 필수 요소이다. 합성곱 레이어 다음에 위치하는 풀링 레이어는 선택적인 레이어이다. CNN 마지막 부분에는 이미지 분류를 위한 풀리 커넥티드 레이어가 추가된다. 이미지의 특징을 추출하는 부분과 이미지를 분류하는 부분 사이에 이미지 형태의 데이터를 배열 형태로 만드는 플래튼 (Flatten) 레이어가 위치한다 [16]. 이와 같은 합성곱 신경망이 모든 딥러닝의 근간이라 할 수 있으며 이를 바탕으로 이미지 딥러닝과 같은 활용이 가능하게 되었다.

2.1 F-RCNN

실험에 사용한 F-RCNN에 대해 설명하기에 앞서 R-CNN에 대해 알아보기로 한다. R-CNN은 CNN을 객체 감지에 적용한 첫 번째 연구이다. 이미지를 분류하는 것보다 이미지 안에 어떤 물체들이 들어 있는지를 구분해내는 것이 훨씬 어려운 작업이기 때문에 R-CNN은 이를 위해 몇 단계를 거쳐 임무를 처리한다.

먼저 가능한 이미지 영역을 찾아내는 지역 제안 (region proposal) 혹은 바운딩 박스 (bounding box)를 만드는 단계가 있다. 바운딩 박스를 찾기 위해 셀렉티브 서치 (selective search) 알고리즘을 사용한다. 가령 색상이나 강도 패턴 등이 비슷한 인접한 픽셀을 합치는 방식이다. 그런 다음 추출한 바운딩 박스를 (대략 2,000여개) CNN의 입력으로 주입하기 위해 강제로 사이즈를 일원화시킨다. CNN의 마지막 단계에서 서포트 벡터 머신 (SVM, support vector machine)을 사용하여 이미지를 분류한다. 그리고 최종적으로 분류된 오브젝트의 바운딩 박스 좌표를 더 정확히 맞추기 위해 선형 회귀 (linear regression) 모델을 사용한다.

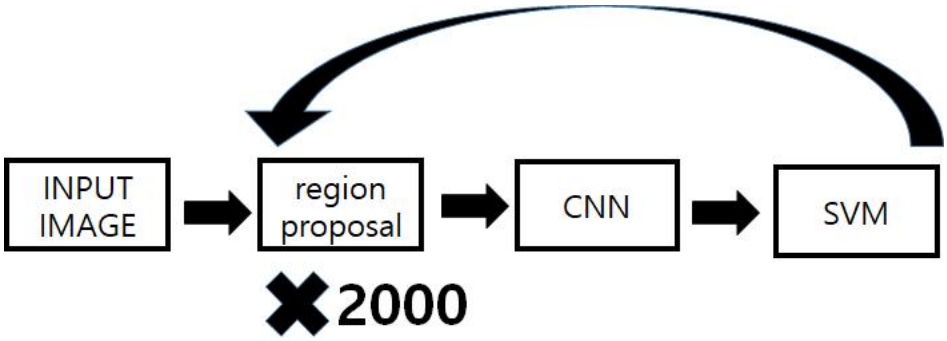


Fig. 5. R-CNN : Regions with CNN features.

Fig. 5는 R-CNN 개념을 설명하는 그림이다. 자세히 살펴보면, (1) 입력이미지를 받아서 (2) 셀렉티브 서치로 2000개의 지역 제안을 추출한 다음, (3) CNN으로 각 제안의 특징을 계산하고 (4) 각 영역의 분류 결과와 바운딩 박스 회귀를 계산한다. 분류기로는 서포트 벡터 머신을 사용한다. R-CNN은 각 제안을 독립적으로 전체 CNN 추출물을 계산하기 때문에 많은 계산 량이 소요되고 서포트 벡터 머신과 바운딩 박스 회귀자의 학습이 분리 되어 있는 것도 처리 시간을 늘리는 요인이다 [17].

기존의 R-CNN은 모든 바운딩 박스를 CNN을 돌려야하기 때문에 바운딩 박스들끼리 겹치는 영역이 많고 따로 따로 CNN을 통과시켜 긴 시간이 소요되는 문제점을 가졌다. F-RCNN은 이러한 문제점을 해결하고자 R-CNN 모델에 ROI Pooling (Region of Interest Pooling)의 개념이 도입되어 소요되는 시간을 획기적으로 단축한 모델이다.

Fig. 6의 F-RCNN의 기본 구조를 보면 셀렉티브 서치에서 찾은 바운딩 박스 정보를 CNN을 통과하면서 유지시키고 최종 CNN 특성 맵으로 부터 해당 영역을 추출하여 풀링한다. 이렇게 하면 바운딩 박스마다 CNN을 돌리는 시간을 획기적으로 단축할 수 있다 [18]. R-CNN의 경우 CNN과 분류를 위한 서포트 벡터 머신, 바운딩 박스를 위한 선형 회귀, 이 세 가지 모델을 모두 훈련시키기 어려운데 F-RCNN은 서포트 벡터 머신과 선형 회귀 모델을 모두 하나의 네트워크

에 포함시켜 훈련을 시킨다. 서포트 벡터 머신 대신 CNN 뒤에 소프트맥스(softmax)를 놓고, 선형 회귀 대신 소프트맥스 레이어와 동일하게 CNN에 뒤에 따로 추가했다. 이렇게 해서 R-CNN 보다 160배가량 빠른 모델이 나왔지만 Fast R-CNN에도 지역 제안을 CNN Network가 아닌 셀렉티브 서치 외부 알고리즘으로 수행하여 병목현상 발생하여 처리시간이 늘어나게 되는 문제점이 있다.

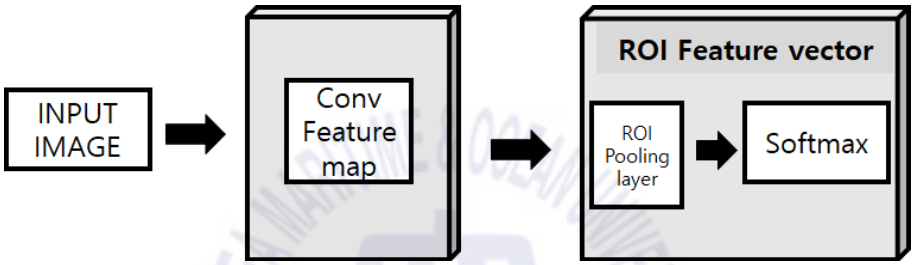


Fig. 6. F-RCNN Basic structure.

2.2 SSD

SSD는 테두리 상자를 조정하기 위해 픽셀이나 특징들을 재 추출하지 않는 것으로 기존의 학습 기법들에서 뒤떨어졌던 속도와 정확성의 반비례 문제를 해결한 모델이다. 이전까지 사용된 딥러닝 디텍터들은 원하는 사진에서 객체가 있는지 없는지 확인하기 위해서는 그림을 자르거나 변형하여 알아내야 했다. 이러한 과정을 위해 이미지 피라미드 및 슬라이딩 윈도우 (Image Pyramid & Sliding Window) 또는, Region Proposal Network (RPN) 등으로 입력 이미지를 변형시켜 네트워크에 집어넣게 된다. 이미지 처리 과정에서 나온 샘플들을 모두 네트워크에서 제각각 객체 검출이 일어나게 되므로 매우 많은 연산 량을 필요로 하게 된다. SSD는 한 장, 한 장의 사진을 변형 없이 한 번의 네트워크 조성만으로 현지화와 여러 물체를 검출하는 모델로 매우 빠른 검출 능력을 가진다.

Fig. 7은 SSD의 기본 구조로 CNN 기반의 기본 네트워크 모델인 VGG-16 (Visual Geometry Group) 네트워크를 특징 추출기로 사용한다. VGG-16은 2014년에 ILSVRC (ImageNet Large Scale Visual Recognition Challenge)라는 이미지 인식 대회에서 준우승한 딥러닝 구조로 13개의 컨볼루션 레이어와 3개의 풀리 커넥티드 레이어로 이루어져있다. VGG-16으로부터 특징 피라미드를 만드는데, 특징 맵은 합성곱 층이 쌓임에 따라 출력 크기가 점점 작아지는 구조이다. 이를 통해 “여러 스케일 = 여러 레이어” 라는 가정 하에 네트워크 구조를 설계된다. 각각의 특징 레이어에서는 합성곱 필터를 적용해 카테고리 별 점수와 기본 상자 (default box)에 상대적인 오프셋을 출력해 검출을 하도록 한다. SSD는 이 과정에서 추출된 모든 특징 맵들을 추론과정에 사용하여 객체를 인식한다. 입력 영상으로부터 가까이 있는 층에서 추출되어 크기가 큰 특징 맵은 작은 물체들을 검출할 수 있고 입력 영상으로부터 멀리 있는 층에서 추출되어 크기가 작은 특징 맵은 큰 물체들을 검출 할 수 있다.

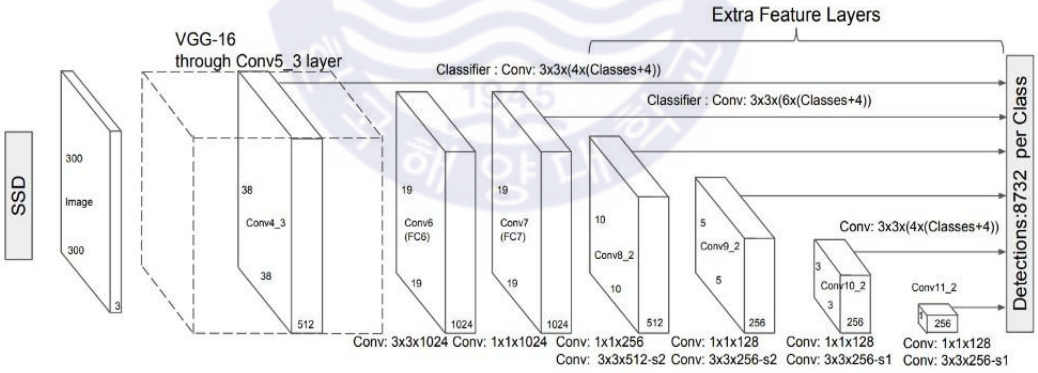
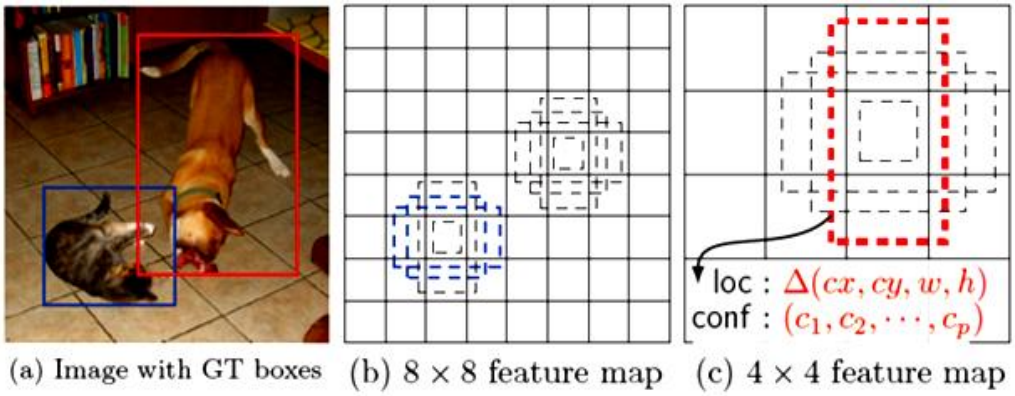


Fig. 7. SSD general structure [19].



(a) detected image **(b)** small figure detector **(c)** large figure detector

Fig. 8. SSD inference method [19].

Fig. 8은 SSD가 가지는 추론 방식으로 멀티 스케일 기능 맵 (Multi-scale feature maps)에 대해 보여준다. Fig. 8(a)는 이미지에서 객체를 검출한 모습이고 Fig. 8(b)와 (c)는 각각 작은 물체와 큰 물체를 찾기 위한 맵을 나타낸다. SSD는 단 한 장의 사진만으로 여러 종류와 크기를 지닌 물체를 검출해야하기 때문에 멀티 스케일 기능 맵이라는 형태를 취해 이를 해결한다. One-Stage 구조의 객체 검출 기법인 SSD는 후보영역을 생성하기 위한 RPN을 따로 학습 시키지 않고 서로 다른 크기를 가지는 객체를 구별하기 위해 기능 맵을 여러 개의 크기로 만들어서, 큰 맵에서는 작은 물체의 검출을, 작은 맵에서는 큰 물체의 검출을 하게 만든다. 이러한 방식은 위치 추정 및 입력 이미지의 리샘플링을 없애면서도 정확도 높은 결과를 도출하게 된다.

SSD의 학습 과정에서 어떤 디폴트 박스가 실제 트루 값과 대응되는지 결정하여야 이에 따라 신경망을 학습할 수 있다. 각 트루 박스는 위치, 종횡비, 스케일이 변하는 디폴트 박스 중 하나를 선택해야 한다. 이를 위해 먼저 트루 박스와 가장 많이 겹치는 박스를 먼서 선택한 뒤, 그 다음 디폴트 박스 중 IOU값이 0.5이상이 되는 트루 박스를 선택한다.

$$L(x,c,l,g) = \frac{1}{N}(L_{conf}(x,c) + \alpha L_{loc}(x,l,g)) \quad (1)$$

Fig. 8(c)에 보이는 loc(로컬라이제이션 로스)와 conf(컨피던스 로스)는 물체를 학습하는데 사용되는 손실 함수이다. 최종 판단에 사용되는 전체 손실 함수는 로컬라이제이션 로스와 컨피던스 로스의 가중치 합으로 식 (1)과 같이 계산된다. 여기서 $x_{ij}^{p} = \{1,0\}$는 i번째 디폴트 박스가 클래스 p의 j번째 그라운드 트루 박스 인디케이터인지 아닌지를 나타내는 인디케이터이다. (IOU 가 0.5 이상이면 1 아니면 0을 의미한다.) N은 매칭 된 디폴트 박스의 수로 N = 0 이라면 손실은 0으로 사용한다. 알파는 교차 검증으로부터 얻어진 1을 의미하며 로스 평션은 크게 2부분, 클래스 점수에 대한 손실과 바운딩 박스의 오프셋에 대한 손실로 나뉜다. N은 검출된 박스의 개수이며 g는 그라운드 트루스 박스를 d는 디폴트 박스를 의미한다. c는 카테고리이며 l은 프레딕 박스, cx, cy는 센터를 w,h는 길이와 높이를 의미한다.

식(2)의 로컬라이제이션 로스는 예측된 박스 l과 실제 트루 박스 g 파라미터 사이의 Smooth L1 loss를 이용하여 계산한다. 디폴트 박스 (d)의 중심의 오프셋 (cx, cy) 디폴트 박스의 크기 (w, h)를 사용한다.

$$L_{loc}(x,l,g) = \sum_{i \text{inPos} m \in \{\text{cx,cy,wh}\}}^{n} \sum x_{ij}^{k} \{\text{smooth}(\text{l}_{\text{i}}^{\text{m}} - \hat{\text{g}}_{\text{j}}^{\text{m}})$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^{w}, \qquad \hat{\text{g}}_{\text{j}}^{\text{cx}} = (\text{g}_{\text{j}}^{\text{cy}} - \text{d}_{\text{i}}^{\text{cy}})/\text{d}_{\text{i}}^{\text{h}} \quad (2)$$

$$\hat{g}_j^{w} = \log(\frac{\text{g}_{\text{j}}^{\text{w}}}{\text{d}_{\text{i}}^{\text{w}}}), \qquad \hat{\text{g}}_{\text{j}}^{\text{h}} = \log(\frac{\text{g}_{\text{j}}^{\text{h}}}{\text{d}_{\text{i}}^{\text{h}}})$$

예측해야할 프레딕 박스의 l_i^m (cx, cy, w, h)값들은 특이한 $\hat{g}$ 값들을 예측한다. 이때 $\hat{g}$의 cx, cy는 디폴트 박스의 cx와 w, h로 노멀라이즈 된다. 예측된 l 값들의 박스를 표현할 때 역시 디폴트 박스의 오프셋 정보가 필요하다.

$$L_{conf}(x,c) = -\sum_{i\in Pos}^{N} x_{ij}^{p}\log(\hat{c}_i^{p}) - \sum_{i\in Ncg}\log(\hat{c}_i^{0}) \text{where } \hat{c}_i^{p} = \frac{\exp(c_i^{p})}{\sum_{p}\exp(c_i^{p})} \quad (3)$$

식(3)의 컨피던스 로스는 포지티브 클래스에 대해서는 소프트 맥스 함수를 사용한다. 네거티브 클래스를 예측하는 값은 $\hat{c}_i^{0}$ 값이고 백그라운드이면 1, 아니면 0의 값을 가져야한다. 최종적인 예측된 클래스 점수는 예측할 클래스 + 배경 클래스를 나타내는 지표이다 [20]. 이와 같은 손실 함수를 사용하여 학습 정확도를 증가시킨다.

본 연구에서는 소형컴퓨터인 라즈베리파이를 사용하여 시스템을 구성하였으므로 적합한 모델을 사용하여야한다. 이를 위해 실제 비교에 앞서 아래와 같은 비교 사례를 조사하여 F-RCNN과 SSD를 비교하기로 하였다.

Table 1. Result on Pascal VOC 2007 test [20].

Model	VOC 2007 test map	FPS	Input resolution
Fast-RCNN	73.2	7	1000 x 600
SSD	77.2	46	300 x 300

Table 1은 Pascal VOC 2007이라는 데이터 셋을 사용하여 두 모델의 FPS를 비교한 모습이다. 상자 안의 숫자를 보면 SSD가 F-RCNN보다 몇 배나 빠른 속도를 보이는 것을 알 수 있다. Table 2를 보면 MSCOCO 데이터 셋을 사용할 경우에도 오른쪽 위의 오각형인 F-RCNN은 정답률은 높지만 검출속도가 느린 모습을 보이는 반면 왼쪽 아래의 SSD는 정답률이 낮지만 매우 빠른 검출속도를 가지는 것을 확인 할 수 있다. 이와 같은 현상은 Fig. 9와 같이 매우 많은 연산을 필요로 하는 F-RCNN과 같은 두 단계 검출 방법들과 비교하여 단일 단계 검출 기법을 사용하는 SSD는 적은 연산 량을 필요로 하여 학습 속도가 향상되는데, 이와 같은 두 모델의 검출 속도가 라즈베리파이에서 어떤 식으로 적용되는지를 조사하였다.

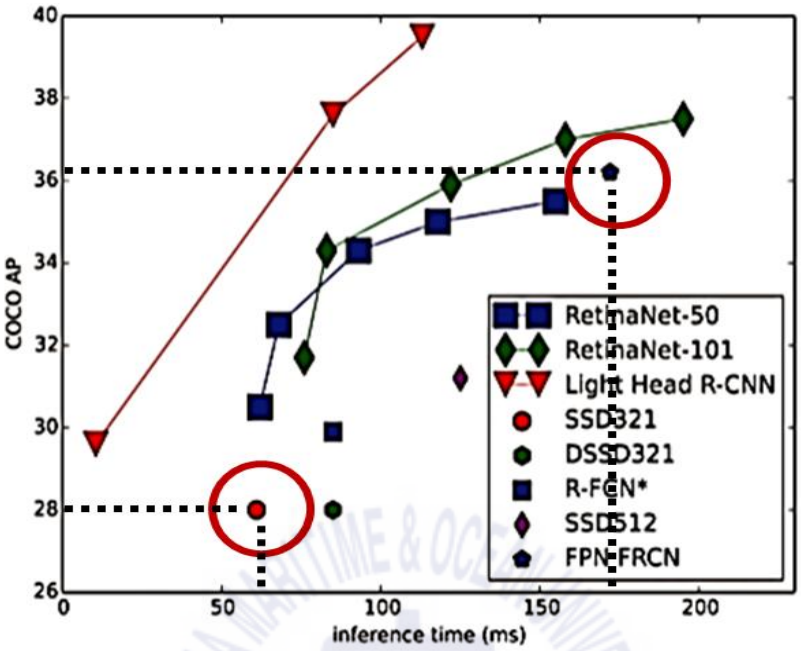


Fig. 9. Object Detection Algorithm Performance Comparison [20].

Table 2. Single stage detector vs. two stage detector.

	단일 단계 검출 방법	두 단계 검출 방법
방법	모든 영역에서 Localization + Classification	1) 대략적인 Localization 2) Classification + 세밀한 Localization
정확도	합리적인 수준의 성능	매우 높은 성능
속도	매우 빠름	느림
예	SSD	FAST-RCNN

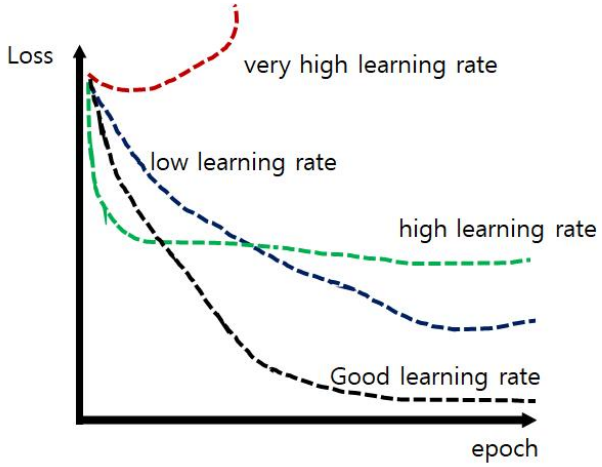


Fig. 10. Hyperparameters summary.

Fig. 10은 하이퍼 파리미터를 최적화함에 따라 정확도가 변화하는 것을 나타내기 위하여 파라미터 중 가장 자주 언급되는 학습률을 어떻게 설정하는가에 따라 학습에 최적화 되는지를 보여준다. 하이퍼 파라미터에서 설정한 값에 따라 모델 파라미터를 최적화하므로 머신러닝과 딥러닝 모델의 성능에 큰 영향을 미치게 된다. 모델 파라미터 최적화는 하이퍼 파라미터로 설정된 범위 내에서만 최적화를 시도하기 때문에 올바르게 설정되지 않은 하이퍼 파라미터에서는 좋은 성능을 보여주기 힘들다. 따라서 모델 파라미터 최적화보다 중요한 것은 하이퍼 파라미터 최적화라고 할 수 있다. Table 3은 딥러닝 학습 시에 조절되는 다양한 하이퍼 파라미터의 예시이다 [21]. 본 연구에서는 학습율과 배치사이즈 등의 하이퍼 파라미터를 조정하는 것으로 보다 높은 학습률을 가질 수 있도록 활용하였다.

Table 3. Hyper Parameter Types.

종류	설명
Learning Rate	학습 진도율은 “gradient”의 방향으로 얼마나 빠르게 이동을 할 것인지를 결정한다. 학습 진도율이 너무 작으면 학습의 속도가 너무 느리게 되고, 반대로 너무 크면 학습이 안 되고 진동할 수 있다.
Cost function	일반적인 최소자승법을 사용할 수 도 있고, cross-entropy 함수를 사용할 수도 있다.
Regularization parameter	overfitting 문제를 피하기 위해 L1이나 L2 regularization 방법을 사용할 수도 있고 거기서 사용하는 일반화 변수는 weight decay의 속도를 조절하기 위한 용도로 사용할 수가 있다.
Mini-batch 크기	Mini-batch크기가 큰 경우 병렬연산 구조를 사용할 때 효과적일 수 있으며, 크기가 작으면 더 많은 update를 할 수가 있다.
Training 반복횟수	학습의 조기 종료를 결정하는 변수가 된다.
Hidden Unit의 개수	Hidden 레이어가 많아질수록 특정 훈련 데이터에 더 최적화 시킬 수가 있다. 또한 모든 hidden 레이어의 뉴런의 개수를 동일하게 유지하는 것이 같은 hidden 레이어의 개수에 뉴런의 개수를 가변적으로 하는 것보다 효과적이다.
가중치 초기화 Weight initialization	바이어스는 일반적으로 0으로 초기화가 된다. 하지만 가중치의 경우는 초기화가 하습결과에 큰 영향을 미치기 때문에 주의가 필요하다.

제 3 장 화재 이미지 학습 및 성능 시험

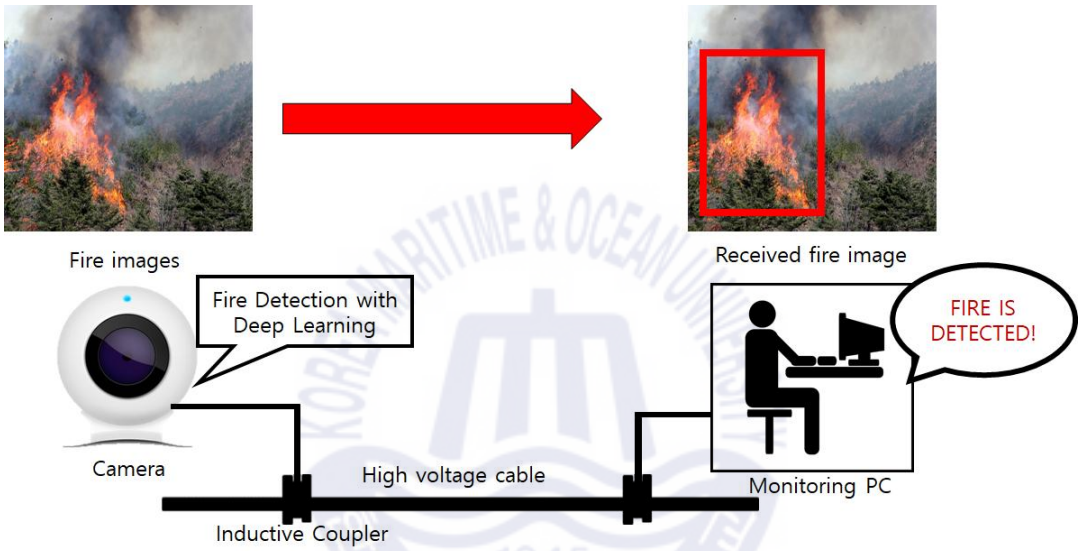


Fig. 11. Fire Detection System Overview.

Fig. 11은 본 논문에서 제안하는 화재 검출 시스템의 개요이다. 카메라에서 딥러닝을 활용한 객체 감지가 실행되면 실시간으로 화재를 감지한다. 화재가 검출될 경우 카메라에서 모뎀으로 데이터를 보내고 유도형 커플러가 장착되어 있는 고전압 전송선을 사용하여 전력선 통신이 이루어지게 된다. 모니터링 PC에 화재가 검출되었음을 알리는 문자와 이미지 데이터가 전송되면 관측자는 받은 데이터를 통해 실제 화제가 일어났음을 확인하면 화재 진압을 위한 조치를 취하게 된다.

3.1 데이터 셋 준비 및 학습

화재 영상을 모을 때 산불뿐만 아니라 건물이나 자동차와 같은 곳에서 일어난 다양한 종류의 화재 데이터를 모았다. 총 20 개의 비디오를 나누어서 5217 개의 이미지를 준비하였고 그 중 2639 개의 화재 이미지 데이터를 사용하여 학습에 이용하였다. 학습에 사용된 시스템 사양은 window10 운영 체제, 인텔 코어 i7 CPU 및 NVIDIA GeForce GTX 1050 GPU를 장착한 컴퓨터이다 [22-25].

Fig. 12는 수집한 이미지에서 화재 부분을 찾아 라벨링하여 데이터 셋을 형성하는 과정이다. 라벨링이란 이미지에 사용자가 이름을 붙이는 것을 의미하는데 준비된 이미지에서 불꽃을 찾은 다음 상자를 만들어 영역 안의 이미지에 이름을 붙이게 되는 것을 말한다. 라벨링된 상자는 이미지에서 어느 위치에 존재하는지 xml 파일형식으로 만들어져 기록되고 이렇게 모인 데이터 셋이 학습에 사용된다.

(a) Original image
학습을 위해 준비된 화재 데이터의 원본 이미지이다.

(b) Labeling
labelimg라는 프로그램을 사용하여 이미지에서 화재 부분을 라벨링한다.

(c) Generated xml file
라벨링하여 생성된 내용이 xml 파일에 기록된다.

Fig. 12. Dataset creation process.

```
</source>
- <size>
    <width>640</width>
    <height>360</height>
    <depth>3</depth>
</size>
<segmented>0</segmented>
- <object>
    <name>fire</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
  - <bndbox>
      <xmin>262</xmin>
      <ymin>124</ymin>
      <xmax>459</xmax>
      <ymax>298</ymax>
    </bndbox>
</object>
```

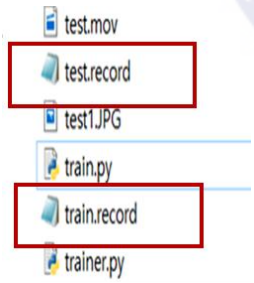
(a) xml file contents

labeling 프로그램을 사용하여 생성된 xml 파일에는 원본 이미지의 사이즈와 붙여진 라벨링 네임, 라벨링된 부분의 위치가 저장된다.

width	height	class	xmin	ymin	xmax	ymax
640	360	fire	262	124	459	298
640	360	fire	315	92	464	239
640	360	fire	172	151	225	203
640	360	fire	299	228	340	273
640	360	fire	305	218	353	273
640	360	fire	254	206	344	277
640	360	fire	240	197	331	268
320	240	fire	99	98	161	146
320	240	fire	99	90	156	144
320	240	fire	132	107	189	149

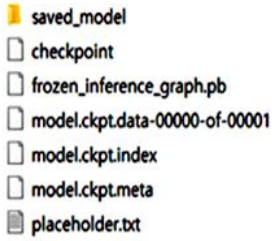
(b) csv file

csv 파일에는 xml 파일의 데이터를 모아 화재 데이터셋을 구성한다.



(c) tfrecord file

텐서플로우는 tfrecord 형식의 데이터를 학습에 사용하므로 csv 파일 형태에서 record 형식으로 변환한다.

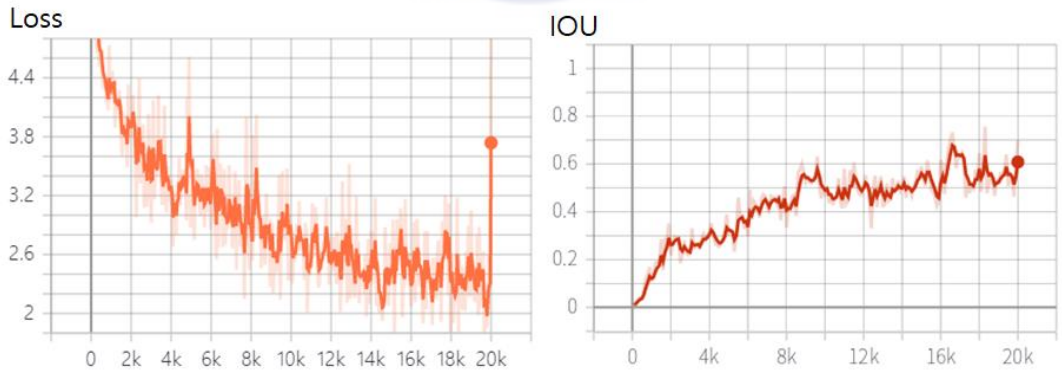


(d) create freeze graph file

학습이 완료되면 최종 학습 단계 정보를 저장하여 객체 검출에 사용되는 프로즌 그래프를 형성한다.

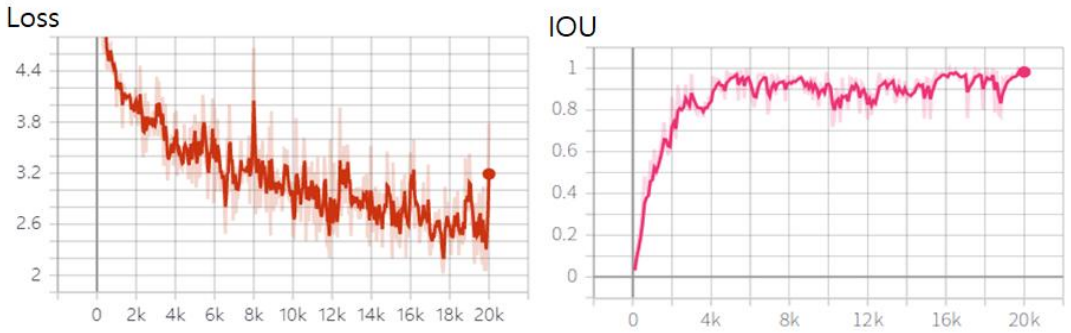
Fig. 13. Data conversion process and training result.

Fig. 13은 앞서 모은 화재 데이터를 그대로 사용할 수 없으므로 텐서플로우에 맞춰 변환하는 과정이다. 먼저 사전에 모은 데이터를 train 셋과 test 셋으로 나누었다. train에 준비된 파일로 학습이 진행되는 동안 test 폴더 안에 들어있는 파일을 사용하여 검증과정을 거치기 때문이다. 이를 위해 데이터를 텐서플로우에 맞춰 변형할 필요가 있다. Fig. 13(a)를 보면 준비된 사진에 대한 크기와 사진 내부에 있는 라벨링된 위치, 라벨링된 이름이 기록되어 있는 xml 파일에서 데이터를 읽어 들인다. 그 다음 Fig. 13(b)와 같이 csv 파일로 변환한 뒤 Fig. 13(c)와 같은 텐서플로우에서 사용되는 tfrecord 파일을 형성하게 된다. 이후 학습에 사용될 학습 기법을 정하고 학습 횟수나 학습 진도율, 정규화 매개 변수(Regularization parameter), Mini-batch 크기 등의 하이퍼 파라미터를 적절히 조절하여 학습을 진행한다. 실험 환경 여건상 batch size는 40으로 잡았고 학습률이나 인풋사이즈를 조절하는 것으로 최적화를 찾는 학습을 진행하였다. Fig. 13(d)은 학습 결과를 저장하는 프로즌 그래프 파일이다. 학습이 진행되고 끝나게 되면 “ckpt” 확장자를 가지는 체크포인트 파일이 형성되게 되는데 이대로도 객체 검출에 사용될 수 있지만 사용되는 환경에 따라 용량도 작고 보다 간편한 프로토버프 파일로 바꿀 필요가 있다. 이 때 프로즌 그래프 파일은 PC나 라즈베리파이에 넣어 객체 검출에 사용된다.



(a) 10 video test Loss (b) 10 video test IOU

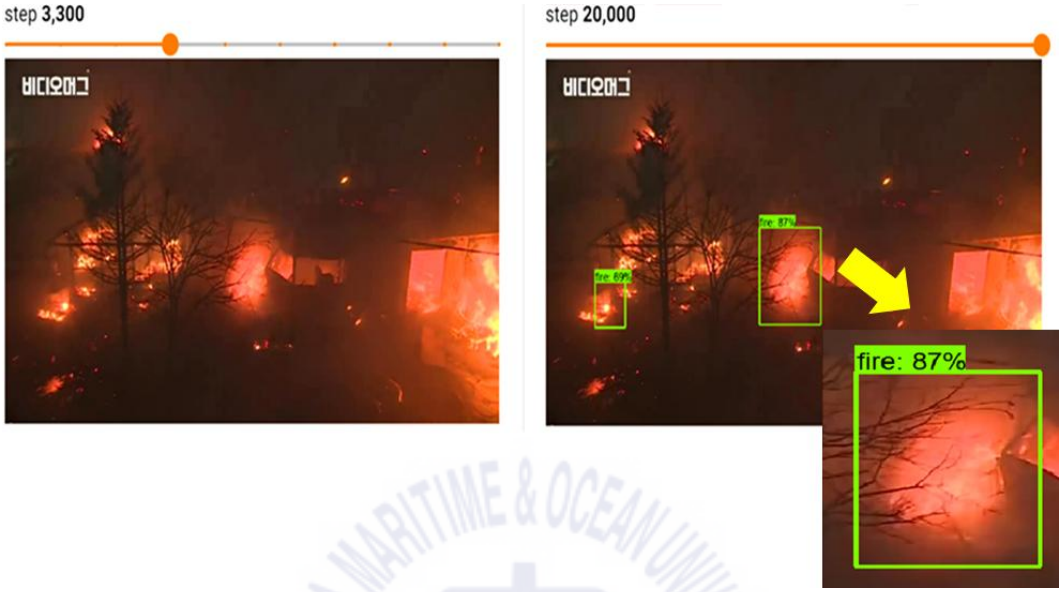
Fig. 14. 10 video learning results.



(a) 17 video test Loss (b) 17 video test IOU

Fig. 15. 17 video learning results.

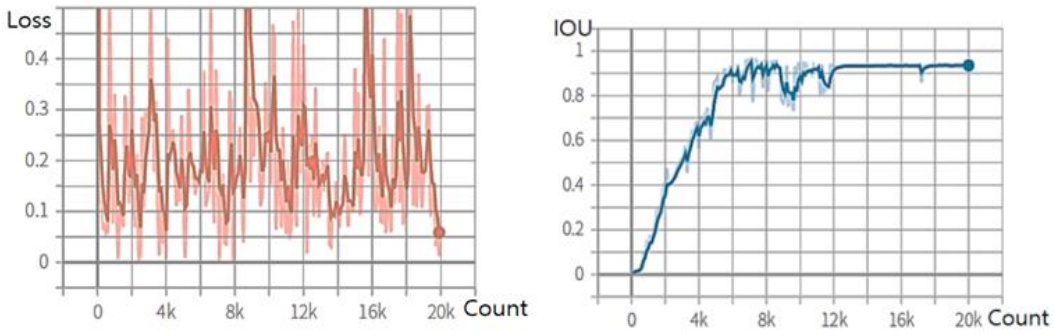
Fig. 14와 Fig. 15는 여러 차례에 걸쳐 SSD를 사용하여 학습한 결과 중 진행이 잘 된 것과 그렇지 않은 것을 비교한 모습이다. 학습 결과를 비교하기 위해 각각 10개, 17개 동영상을 사용하여 학습을 진행하였을 때 정확도가 어떻게 달라지는가를 확인하였다. Fig. 14(a)와 Fig. 15(a)는 학습률을 알아보기 위하여 사용된 손실함수 값을 의미하는데 손실함수는 모델의 출력 값과 사용자가 원하는 출력 값의 차이, 즉 오차를 말한다. 두 그래프를 보면 마지막에 값이 뛰긴 하지만 모두 약 2에 수렴하려는 모습을 보이는데 이와 같이 수렴되어야 학습이 이루어졌다고 판단 내리게 된다. 두 그래프 모두 손실함수는 비슷하게 떨어지지만 Fig. 14(b)와 Fig. 15(b)는 학습 과정에 따른 IOU 정확도를 나타내는데 IOU값이 0.5를 넘는 경우 트루로 판단하여 학습이 진행됨에 따라 정확도가 올라가게 된다. 두 그래프를 확인하면 학습하는 데이터의 종류가 부족한 경우 정확도가 60%정도밖에 나오지 않는 모습을 보인다. 하지만 데이터의 종류를 늘릴 경우 정확도가 올라간 것을 알 수 있었다. 이를 통해 손실함수나 IOU 정확도 한 쪽만 가지고 모델 성능을 판단 할 것이 아니라 두 지표 모두 사용할 필요를 알 수 있다.



(a) step 3,300 (b) step 20,000

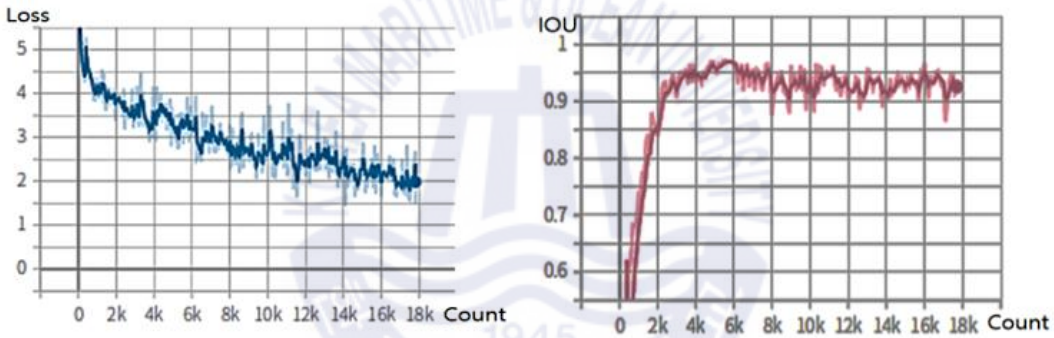
Fig. 16. Detection comparison according to the learning progress.

Fig. 16은 텐서 보드에서 일정 학습 횟수 마다 valid 파일을 사용하여 학습 진행 단계에 따라 이미지 내에서 화재 검출을 할 수 있는가를 확인하는 모습이다. Fig. 16(a)와 (b)를 보면 학습 진행단계 3,300에서는 검출되지 않았던 불꽃이 학습이 진행됨에 따라 검출 할 수 있게 되고 최종적으로 학습 20,000번째 단계에서는 87%의 유사도를 가지는 화재가 검출되는 것을 볼 수 있다. 이와 같은 경우, 학습이 잘 진행된 경우이다. 학습이 진행됨에 따라 모델 내에서의 불꽃에 대한 특성이 변화하게 되어 찾지 못하던 불꽃을 찾을 수 있었다. 이와 같이 학습 진행률을 확인하는 평가 지표뿐만 아니라 중간 중간에 학습 정도를 확인하여 정확도를 높일 수 있다.



(a) F-RCNN Loss (b) F-RCNN IOU

Fig. 17. F-RCNN Learning Results Graph.



(a) SSD Loss (b) SSD IOU

Fig. 18. SSD Learning Results Graph.

Fig. 17과 Fig. 18은 각각 F-RCNN과 SSD를 사용하여 동일한 데이터 세트를 사용할 때의 학습 결과를 보여준다. 텐서플로우의 텐서보드를 사용하여 두 모델 훈련 결과를 확인하면 손실 값이 감소하고 IOU 값이 증가함을 알 수 있다. Fig. 17(a)와 Fig. 18(a)의 두 그래프는 학습과정에서 나온 손실[1] 값을 나타낸다. F-RCNN의 손실은 학습 횟수가 증가함에 따라 거의 0으로 감소하고 SSD는 거

1) 손실 함수는 학습률을 판단하는 지표 중 하나로서 학습이 진행 되면서 수렴하는 모습을 보인다. Tensor Flow HELP-TOOL를 사용하는 경우 손실함수는 학습모델에 따라 수렴 값이 달라진다. 손실 함수는 최종 수렴 값과는 별개로 수렴한다는 사실 자체가 중요하다. 위와 같은 경우 SSD와 F-RCNN의 수렴 값이 다르더라도 학습이 진행되었다고 판단내릴 수 있다.

의 2로 수렴한다. 손실 함수의 값과 별개로 위와 같이 수렴하게 되면 손실 함수가 최소화가 되었고 학습이 정상적으로 이루어지고 있다고 볼 수 있다. 또한 두 그림의 우측에 표시된 것처럼 정확도를 뜻하는 IOU가 90% 이상으로 높기 때문에 학습이 잘 진행되었다고 판단하였다. 충분히 학습이 진행되었기 때문에 형성된 freeze 그래프 파일을 사용하여 PC와 라즈베리파이에서 화재 감지를 시험하였다.

(a) fire video 1 (b) fire video 2

Fig. 19. Fire detection test on PC.

Fig. 19는 여러 차례 학습이 실행된 것들 중에 가장 잘 나온 학습 모델과 학습에 사용한 동영상 중 2 개를 사용하여 PC에서 화재를 검출 하는 모습이다. 가장 잘 되었다고 판단하지만 Fig. 17(a)가 99%인데 비해 Fig. 17(b)는 71%로 비교적 낮은 정답률을 보였는데 이는 학습에 사용된 화재 중에서도 비디오 2의 경우 화재 부분에 연기나 건물이 비교적 많이 포함되어 인식률이 떨어진 것으로 보인다. 테스트 결과 다른 학습모델들에 비해 동영상 속 화재가 잘 검출되었기 때문에 라즈베리파이에서도 동일한 결과를 기대하였다.

3.2 객체 검출 속도 비교

(a) F-RCNN (0.05 FPS) (b) SSD (1.88FPS)

Fig. 20. F-RCNN vs SSD Speed Comparison.

Fig. 20은 F-RCNN과 SSD를 사용하여 만든 학습 모델을 라즈베리파이에 넣어 화재 감지 속도를 비교한 모습이다. Fig. 20(a)는 F-RCNN 모델을 사용하여 화재 감지를 수행 할 때의 모습으로 휴대폰 화면에서 동영상의 화재를 감지하여 0.05의 초당 프레임 수 (FPS)가 나오는 것을 확인하였다.

Fig. 20(b)은 SSD 모델을 사용하였을 경우인데 파이카메라가 노트북에 표시된 화재 이미지를 읽어 이미지에서 검출된 화재 이미지는 모니터 화면의 상자 형태로 감지된다. SSD를 사용하였을 경우의 초당 프레임 수는 1.88로 실시간 처리에 가깝다. 본 논문에서는 위에서 언급 한 F-RCNN과 SSD가 물체 감지에 사용되었다. F-RCNN의 물체 감지에는 많은 계산이 필요하므로 라즈베리파이와 같은 소형 컴퓨터에는 적합하지 않다. 따라서 소형 컴퓨터 리소스에 적합한 SSD를 사용하여 F-RCNN 모델의 화재 감지 속도와 비교했다. 실제 검출 속도를 비교한 결과 SSD가 라즈베리파이에 적합하다는 것을 확인 할 수 있었다.

제 4 장 전력선 통신 기반 화재 검출 시스템 구현

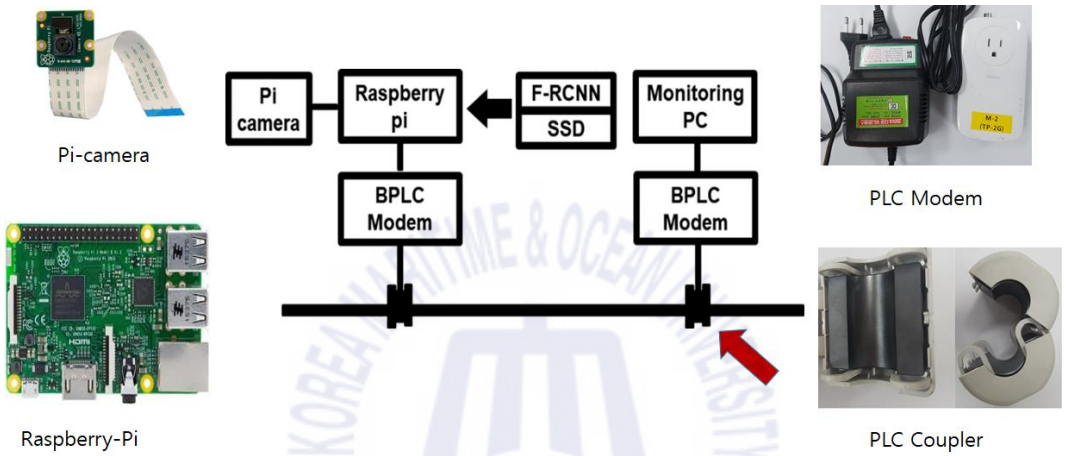


Fig. 21. Fire Detection System Outline.

Fig. 21은 전력선 통신 기반 화재 감시 시스템 개요도이다. 라즈베리파이에 파이카메라를 장착한 후 앞서 비교한 F-RCNN과 SSD로 학습된 파일을 라즈베리파이 넣어 화재검출 성능을 실험하였다. 또한 화재가 검출될 경우에는 전력선통신을 사용하여 텍스트와 검출된 이미지가 원격지의 모니터링 PC로 전송되도록 한다. 객체 검출을 위하여 라즈베리파이 내부에는 텐서플로우 모듈과 OpenCV 모듈이 설치된다. 영상 내에서 화재 이미지를 찾지 못하면 새로운 영상을 반복하여 객체검출을 실시한다가 화재가 감지되면 전력선 통신을 이용하여 텍스트와 이미지를 보내고 라즈베리파이는 일시 정지하게 된다. 전송받은 데이터는 모니터링 PC에 출력되고 사용자가 이를 확인하면 PC내부에 저장하게 되고 파이에 탐색을 재개 하라는 명령을 내려 화재감지가 반복된다.

4.1 전력선 통신 적용시험

전력선 통신 기술 (Power Line Communication : PLC)이란 전력을 공급하는 전력선을 이용해서 음성과 데이터를 수백 kHz에서 수십 kHz 이상의 고주파 신호에 실어 통신 하는 기술이다. Fig. 22는 기본적인 전력선 통신의 구성도이다. 접촉식 전력선 통신은 전력송신을 목적으로 하는 전력선의 전원파형 (60Hz)에 고주파로 변조된 통신신호를 실어서 음성, 문자, 영상 및 데이터를 전송하는 유선통신을 의미한다.

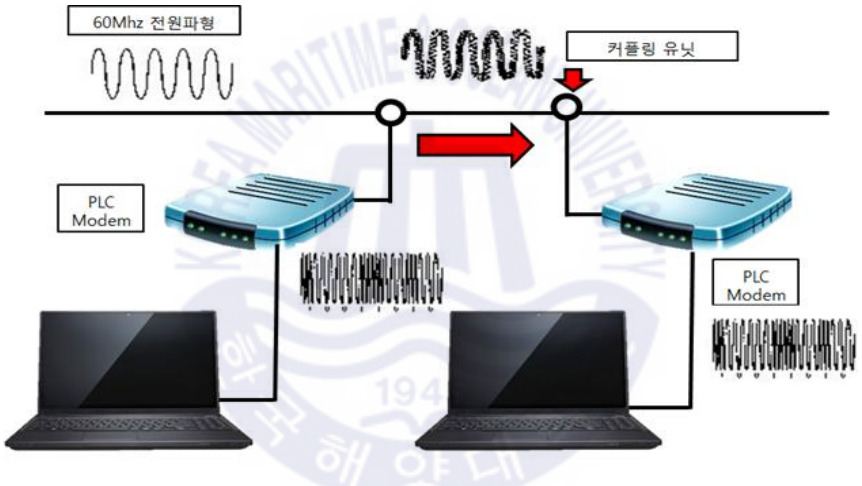


Fig. 22. Power Line Communication Configuration.

유도 전력선 통신은 전선을 분리하거나 다시 배선하지 않고도 두 개의 유도형 결합기 (커플러)를 전선에 체결하여 통신을 수행하는 방식이다. 유도형 결합기를 사용한 전력선 통신은 전선을 분리하거나 새로 설치하지 않고 전선에 결합기를 직접 장착하여 60 Hz 상용주파수와 변조된 고주파신호가 동시에 흐를 때 자기유도 원리에 따라 통신 신호만을 유도해 낸다. 접촉식 커플러에 비해 설치 작업이 용이하고 안정성이 우수하며 전압에 관계없이 사용할 수 있으며, 저속/고속 통신 모두 적용 가능하다 [26-28].

(a) Coupler (b) Modem

Fig. 23. PLC components.

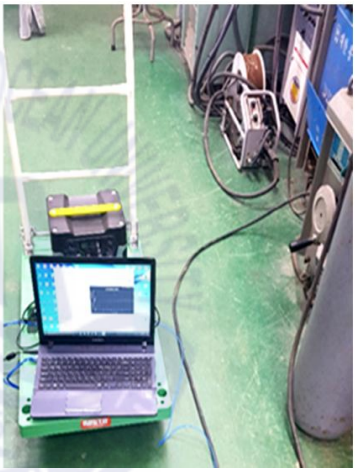
Fig. 23(a)은 실험에 사용된 유도형 페라이트 커플러와 전력선 모뎀이다. 전형적인 Mg-Zn계 페라이트 코어로 만들어진 커플러의 주성분은 Fe, Mn, Mg, Zn 등의 금속 이온들로 이루어진 자성체 재료이며 세라믹과 유사하다. 코어 외형은 결합하기 용이하도록 원통 기둥 형태에서 절반으로 가른 형태이다. 본 연구에서 사용한 페라이트 코어는 1.7 ~ 35 MHz의 주파수에서 통신이 이루어진다. Fig. 23(b)은 기가 광대역 전력선모뎀이다. 이러한 Gbps 모뎀은 전력선에서 최대 2000 Mbps, 이더넷에서는 10/100/1000 Mbps의 전송 속도를 가진다. 또한 옥내에서 300 미터의 통신 가능 거리를 가지며 두 개의 Gigabit Ethernet Port와 한 개의 Power Socket이 설치되어있다.

화재감지 시스템을 구성하는 전력선 통신성능을 알아보기 위하여 통신 속도 측정용 프로그램인 Jperf 프로그램을 사용하였다. Jperf는 클라이언트와 서버간의 네트워크 대역폭을 측정하는 Iperf 프로그램에서 시각적 자료를 더하고 사용자가 프로그램을 보다 용이하게 사용 할 수 있도록 인터페이스를 제공한 것이다. Jperf는 Iperf의 특징인 다양한 플랫폼에서 사용가능하며 IPv4는 물론 IPv6에서도 사용이 가능하다. 간단한 명령어를 사용하여 원하는 옵션을 선택하여 네트워크 스트림 측정이 가능하다. 사용방법은 간단하여 모드를 선택할 때 한쪽은 서버로 열어두고 다른 한쪽은 클라이언트로 설정하여 서버의 주소를 기입한다. TCP Window Size는 300k로 고정한 상태에서 10초 동안 통신 속도를

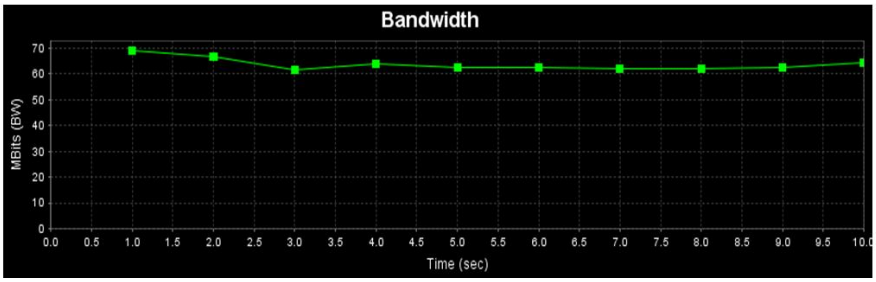
측정하였다.

Fig. 24(a)와 (b)는 아크 용접에 사용되는 고전류 케이블을 사용하여 전력선 통신을 시험하는 모습이다. Fig. 24(c)는 고전류 케이블에서 Jperf를 사용하여 통식속도를 측정한 모습으로 그 결과는 Fig. 25와 같다. 고전류에서의 성능을 알아보기 위하여 거리에 따른 통신 속도를 측정한 용전전과 용접 중에 통신 속도의 변화를 확인 하였다. 실험 결과 용접전이 용접중보다 통신 속도가 높은 것을 확인하였고 거리가 늘어남에 따라 통신 속도는 줄어들지만 90m 까지도 통신이 이루어지는 것을 확인 할 수 있었다.

(a) transmitter



(b) receiver



(c) Jperf bandwidth measurement

Fig. 24. Communication performance measurement on high current cables.

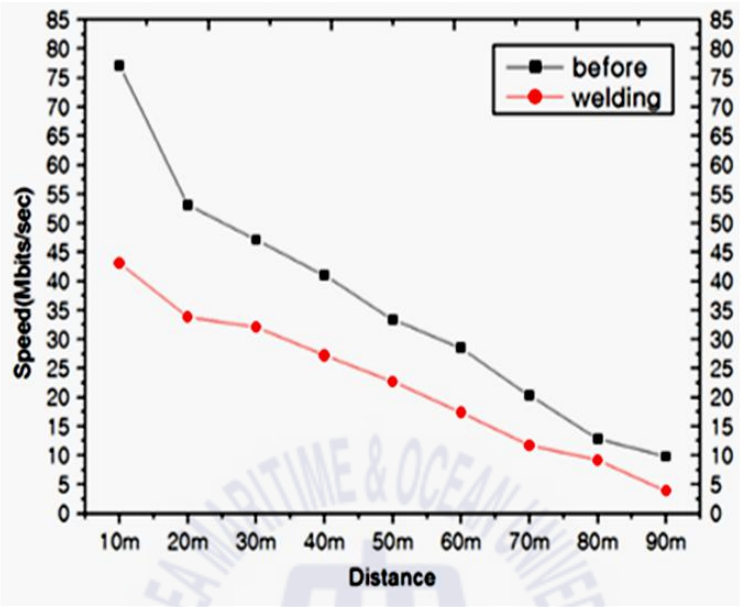
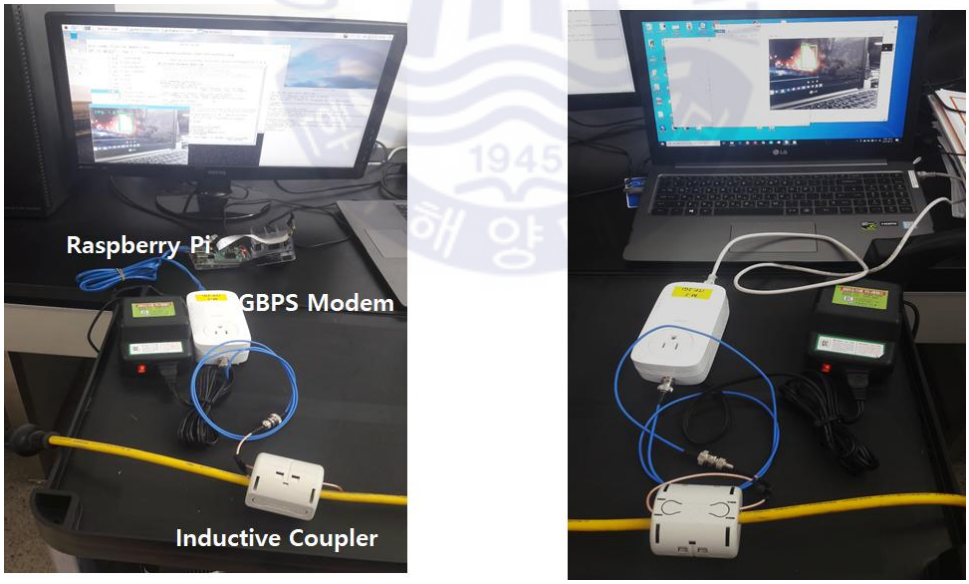


Fig. 25. Transmission rate by distance for cut-core coupling unit.

4.2 화재 검출 실험 결과

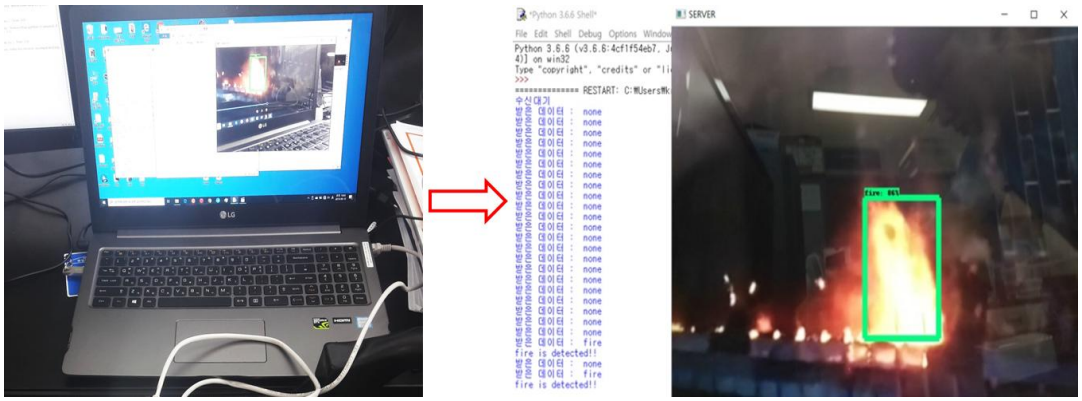
Fig. 26은 라즈베라파이와 전력선으로 구현한 화재 감지 시스템의 실물사진이다. 그림의 왼쪽은 송신 단으로 라즈베리파이에 기가비트 전력선모뎀과 유도성 커플러와 함께 설치되어 있으며 오른쪽은 모니터링 PC에도 기가비트 전력선모뎀이 연결되어 있다. 본 실험에서는 2m 길이의 전원 케이블에 전력선 통신 장치를 설치하였다. 라즈베리파이에서 화재가 감지되면 불꽃 영역에 상자가 형성되고 이렇게 형성된 이미지는 전력선 통신을 이용하여 텍스트와 이미지 데이터가 모니터링PC로 전송되도록 하였다.

Fig. 27은 라즈베리파이에서 전송한 텍스트와 이미지 데이터를 모니터링 PC에 전송한 모습이다. 화재가 감지되지 않은 평소에는 라즈베리파이는 “none”이라는 문자를 계속해서 보내다가 학습된 객체 감지 모듈에서 불꽃을 감지하고 화재가 일어났음을 알리는 “fire” 문자와 문자화 된 이미지 데이터가 라즈베리파이에서 모니터링 PC에 전송된다. 문자가 “fire” 일 경우 뒤따라오는 문자화 된 이미지 데이터는 모니터링 PC측에서 디코딩하여 화면에 띄우게 되고 파이썬 쉘에는 “fire is detected!!” 라는 문자열을 띄우게 된다. 관측자가 이를 확인하면 PC에 저장되게 된다. 라즈베리파이는 데이터를 전송한 후 대기 상태가 되는데 PC에서는 관측자가 확인하면 라즈베리파이에 “watch” 라는 문자를 보내어 라즈베리파이가 화재 감지를 재개하도록 한다. 전송 받은 이미지를 확인하면 라즈베리파이에서 화재라고 판단하여 검출된 영역이 박스로 가두어지고 “fire” 라고 라벨링된 것을 확인 할 수 있다.



(a) transmitter (b) receiver

Fig. 26. Fire Detection System Configuration.



(a) monitoring PC (b) Detailed image

Fig. 27. Text, image transfer experiment.

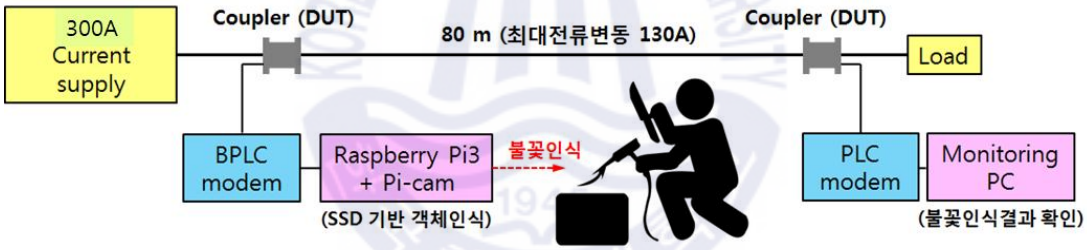


Fig. 28. Diagram of transmit test under current fluctuation conditions.

Fig. 28은 전류변동 선로환경에서 객체인식 시스템의 정상 동작 여부를 시험하여 본 연구에서 개발한 광대역 결합기의 전송한계를 보여주고자 한다. SSD 학습모델이 장착된 라즈베리파이에서 불꽃을 검출하면 130A의 전류변동이 있는 선로를 통하여 캡처한 이미지와 상황을 전달하는 메시지를 모니터링 PC로 전달한다.

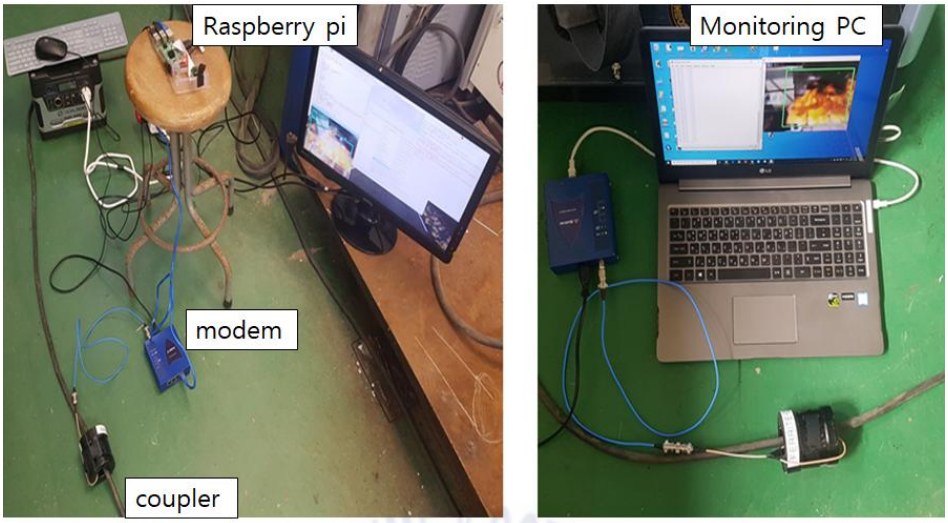


Fig. 29. System configuration in high current wire.

Fig. 29는 아크 용접에 사용되는 고전류 전선에서 화재 검출 시스템을 설치한 모습이다. 실험에 사용된 전선은 300A 이하의 직류 고전류가 흐르는 150m 용접용 전선으로 양 옆 약 5m에 송신단과 수신단을 설치한다. 용접 중에 생기는 노이즈로 라즈베리파이에서 보내는 텍스트 데이터와 이미지 데이터가 영향을 받아 손상되는지를 확인하기 위하여 이와 같은 실험을 진행하였다. 이러한 실험 구성은 모니터링 PC에 위치한 관측자가 작업자가 업무를 보고 있는지를 주기적으로 확인하는데 용이할 것으로 보고 있다.

Fig. 30은 고전류 용접선에서 텍스트와 이미지 데이터를 전송하는 실험을 한 모습이다. 실험에서는 작업자가 사용하는 아크 용접기가 사용 될 때 일어나는 빛을 라즈베리파이에서 불꽃이라 인식하여 관측자에게 불꽃이 감지되었다고 전하도록 하였다. 위 사진과 같이 고전류의 전선에서도 전력선 통신이 무사히 일어나 텍스트와 이미지를 전송하는 것을 볼 수 있었다. 확대된 이미지를 확인하면 라즈베리파이에서 98%의 확률로 불꽃으로 판단하고 있음을 알 수 있다.

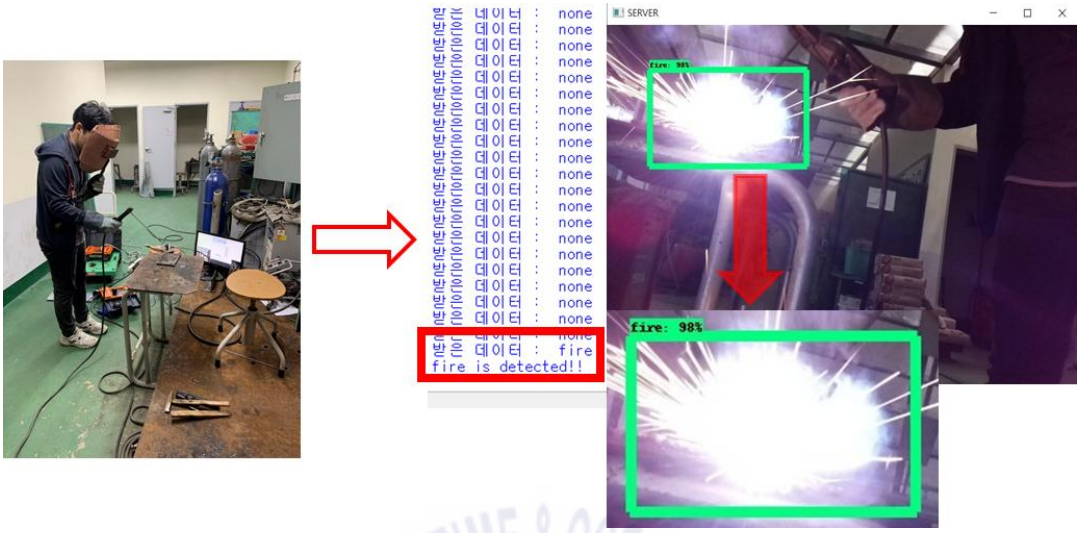


Fig. 30. Text and Image Transmission Experiments on High Current Wires.

제 5 장 결 론

사람이 자주 찾지 않는 지역은 화재가 일어날 경우 초기 대응까지 긴 시간이 걸리기 때문에 대형 화재로 커지는 것을 막지 못하게 된다. 때문에 화재가 대형으로 커지기 전에 감지하는 것에 대해 관심을 가지게 되었다.

본 논문에서는 라즈베리파이에 딥러닝 기술인 객체감지를 적용하고 전력선 통신 기술을 사용하여 넓은 지역의 화재감지 시스템을 구성하였다. 객체 감지를 사용하기 위하여 텐서플로우라는 오픈소스 라이브러리의 딥러닝 모듈을 사용하여 학습을 진행하기로 하였다. 라즈베리파이와 같은 소형컴퓨터에 적합한 학습 기법을 찾기 위하여 자주 비교되는 F-RCNN모델과 SSD모델을 사용하여 라즈베리파이에서 검출 속도를 비교하였을 때 F-RCNN이 0.05로 매우 느린데 반해 SSD는 1.88로 수십 배나 빠르게 검출되어 SSD가 적합하다는 것을 확인 할 수 있었다. 또한 고전압과 고전류 환경에서도 전력선 통신은 통신이 잘 이루어지는지 확인하였다. 라즈베리파이에서 파이카메라를 통해 화재를 감지하면 유도형 전력선 통신 기술을 사용하여 모니터링 PC로 화재가 일어났음을 알리는 텍스트와 이미지를 전송하는 실험을 진행하였다. 테스트 결과 고전류가 흐르는 150m 길이의 용접선에서도 환경에서도 텍스트와 이미지를 전송하는데 성공하였다.

이와 같은 시스템은 산간 지역과 같은 인적인 드문 장소에서 송전선에 카메라를 설치하는 것으로 효율적으로 화재감시를 이룰 수 있을 것으로 기대된다. 향후 라즈베리파이에는 SSD를 사용하고 모니터링 PC에는 F-RCNN을 적용하여 검출된 이미지가 화재임을 한 번 더 확인하는 모델을 구현해볼 예정이며 YOLO나 Darknet과 같은 여러 감지 방법을 사용하여 화재 감지 시스템을 시도할 것이다.

참고문헌

[1] http://www.index.go.kr/potal/main/EachDtlPageDetail.do?idx_cd=1309

[2] S-H. Lee, B.-J. Shin, B.-D. Song, S.-J. An, J.-D. Kim, H.-J. Lee, "Wild Fire Monitoring System using the Image Matching," J. Korea Contents Society, vol. 13, no. 6, pp. 40-47, 2013.

[3] J. H. Son, Y. Huh, Y. G. Byun, K. Y. Yu, Y. I. Kim, "Designing and Building a Fire Monitoring Web GIS System Using MODIS Image -Using ArcIMS 4.0," J. GIS Association of Korea, vol. 14, no. 1, pp. 151-161, 2006.

[4] K. Y. Kim, K. E. Lee, J. H. Han, "Forest fire Detection System using IoT-based drones," Proceeding of Korea Information Science Society, pp. 157-159, Dec. 2016.

[5] D. Y. Yun, S. H. Kim, "A Design of Fire Monitoring System Based On Unmaned Helicopter and Sensor Network", J. Korean Institute of Intelligent Systems, vol. 17, no. 2, pp. 173-178, 2007.

[6] K. Muhammad, J. Ahmad, S. W. Baik, "Early fire detection using convolutional neural networks during surveillance for effective disaster management", ,vol.288,pp.30-42,May2018

[7] Y-J. Kim, E.-G. Kim, "Real-Time Fire Detection based on CNN and Grad-CAM," J. Korea Institute of Information and Communication Engineering, vol. 22, no. 12, pp. 1596-1603, 2018.

[8] J-J. Kim, J-K. Ryu, D-K. Kwak, S-J. Byun, "A Study on Flame Detection using Faster R-CNN and Image Augmentation Techniques," J. Korean Electrical and Electronics Engineering, vol. 22, no. 4, pp. 1079-1087, 2018.

[9] S-W. Bang, "Implementation of Image based Fire Detection System Using Convolution Neural Network," J. Korea Institute of Electronic Communication Sciences. pp. 331-336, vol. 12, no. 2, 2017.

[10] https://blogs.nvidia.co.kr/2016/08/03/difference_ai_learning_machinelearning/

[11] https://3months.tistory.com/465

[12] https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173, Jonathan Hui's Medium "mAP for Object Detection "

[13] Mark Everingham, S.M. Ali Eslami, Luc Van Gool, Christopher K.I. Williams, John Winn, Andrew Zisserman, The Pascal Visual Object classes Challenge: A Retrospective. 2010, 111, 98-136.

[14] https://en.wikipedia.org/wiki/Power-line_communication

[15] https://www.keri.re.kr/_prog/_board/?mode=V&no=3927&code=child0404&site_ dvs _cd=child&menu_dvs_cd=0406&gubun=

[16] A.Krizhevsky,I.Sutskever,andG.E.Hinton, "Imagenet classification with deep convolutional neural networks," Advances in neural information processing systems,2012,pp.1097-1105.

[17] R. B. Girshick. "Fast R-CNN," in Proceedings of IEEE International Conference on Computer Vision, 2015.

[18] Ren, Shaoqing, et al. "Faster R-CNN: Towards real-time object detection with region proposal networks." Advances in neural information processing systems. 2015.

[19] Liu, Wei, et al. "Ssd: Single shot multibox detector." European conference on computer vision. Springer, Cham, 2016.

[20] https://taeu.github.io/paper/deeplearning-paper-ssd/

[21] http://global-autonews.com/bbs/board.php?bo_table=bd_035&wr_id=392

[22] http://www.engear.net/wp/hyper-paramertesr-/

[23] TensorFlow System, http://www.tensorflow.org/

[24] https://github.com/EdjeElectronics/TensorFlow-Object-Detection-on-the-Raspberry-Pi

[25] https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple- Objects-Windows-10

[26] Jin-Won Lee "3D-Convolutional Neural Network for Efficient Hand Gesture Recognition" Department of Electrical and Computer Engineering The Graduate School Seoul National University 2017. 2 Thesis

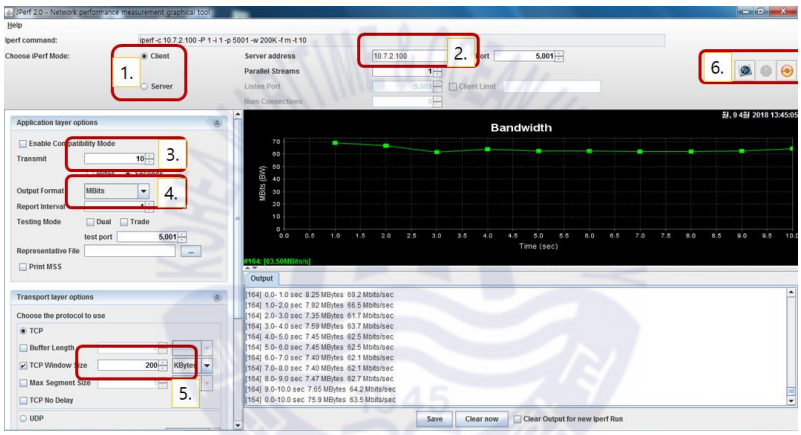
[27]J. Binkofski, "Influence of the properties of magnetic materials on the size and performance of PLC couplers," Int. Symp. Power Line Commun and Its Appl., pp. 281-284, 2005.

[28] Seung-Ho Yang, Jae-Hwan Jeong, Kyung-Rak Sohn, "Inductive Power Line Communication Performance Comparison Using Nano-Crystalline Alloy and Ferrite Coupling Device in Ship" J. KICS, vol. 43, no. 4, pp.740-746, 2018.

[29] Seung-Ho Yang, Jae-Hwan Jeong, Kyung-Rak Sohn, "Implementation of Soft Magnetic Core Type Coupler for Broadband Power Line Communication" J. KICS, vol. 44, no. 4, pp.693-700, 2019.

부록

1. 전력선 통신 대역폭 측정을 위한 Jperf 사용법



1. 모드 설정 - 서버 클라이언트를 설정한다.
2. 주소 기입란 - 클라이언트 모드에서 서버 주소를 기입
2. 통신시간 지정 - 몇 초 동안 통신 속도를 측정할지를 정함
3. 출력 값 치정 - 출력 되는 통신 속도를 Kbits 혹은 Mbits 로 설정
4. TCP Window size - 네트워크에 있을 수 있는 데이터의 양을 제어
5. 실행버튼 - 서버 측에서 실행버튼이 눌려져 서버가 열려 있을 경우 클라이언트 측에서 실행 버튼을 누르면 통신 시작