JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 25 No. 12, pp. 63-71, December 2020
https://doi.org/10.9708/jksci.2020.25.12.063

AI Fire Detection & Notification System

You-min Na*, Dong-hwan Hyun*, Do-hyun Park*, Se-hyun Hwang*, Soo-hong Lee*

*Master's Course, Dept. of Mechanical Engineering, Yonsei University, Seoul, Korea
*Master's Course, Dept. of Mechanical Engineering, Yonsei University, Seoul, Korea
*Master's Course, Dept. of Mechanical Engineering, Yonsei University, Seoul, Korea
*Master's Course, Dept. of Mechanical Engineering, Yonsei University, Seoul, Korea
*Professor, Dept. of Mechanical Engineering, Yonsei University, Seoul, Korea

[Abstract]

In this paper, we propose a fire detection technology using YOLOv3 and EfficientDet, the most reliable artificial intelligence detection algorithm recently, an alert service that simultaneously transmits four kinds of notifications: text, web, app and e-mail, and an AWS system that links fire detection and notification service. There are two types of our highly accurate fire detection algorithms; the fire detection model based on YOLOv3, which operates locally, used more than 2000 fire data and learned through data augmentation, and the EfficientDet, which operates in the cloud, has conducted transfer learning on the pretrained model. Four types of notification services were established using AWS service and FCM service; in the case of the web, app, and mail, notifications were received immediately after notification transmission, and in the case of the text messaging system through the base station, the delay time was fast enough within one second. We proved the accuracy of our fire detection technology through fire detection experiments using the fire video, and we also measured the time of fire detection and notification service to check detecting time and notification time. Our AI fire detection and notification service system in this paper is expected to be more accurate and faster than past fire detection systems, which will greatly help secure golden time in the event of fire accidents.

▸ **Key words**: Fire Detection, YOLOv3, EfficientDet, Notification, Real-time

• First Author: You-min Na, Corresponding Author: Soo-hong Lee
*You-min Na (umuna@yonsei.ac.kr), Dept. of Mechanical Engineering, Yonsei University
*Dong-hwan Hyun (donghwan9411@gmail.com), Dept. of Mechanical Engineering, Yonsei University
*Do-hyun Park (pdh7183@yonsei.ac.kr), Dept. of Mechanical Engineering, Yonsei University
*Se-hyun Hwang (hwanghyun3@gmail.com), Dept. of Mechanical Engineering, Yonsei University
*Soo-hong Lee (shlee@yonsei.ac.kr), Dept. of Mechanical Engineering, Yonsei University
• Received: 2020. 11. 27, Revised: 2020. 12. 15, Accepted: 2020. 12. 15.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

[요 약]

본 논문에서는 최근 가장 신뢰도 높은 인공지능 탐지 알고리즘인 YOLOv3와 EfficientDet을 이용한 화재 탐지 기술과 문자, 웹, 앱, 이메일 등 4종류의 알림을 동시에 전송하는 알림서비스 그리고 화재 탐지와 알림서비스를 연동하는 AWS 시스템을 제안한다. 우리의 정확도 높은 화재 탐지 알고리즘은 두 종류인데, 로컬에서 작동하는 YOLOv3 기반의 화재탐지 모델은 2000개 이상의 화재 데이터를 이용해 데이터 증강을 통해 학습하였고, 클라우드에서 작동하는 EfficientDet은 사전학습모델(Pretrained Model)에서 추가로 학습(Transfer Learning)을 진행하였다. 4종류의 알림서비스는 AWS 서비스와 FCM 서비스를 이용해 구축하였는데, 웹, 앱, 메일의 경우 알림 전송 직후 알림이 수신되며, 기지국을 거치는 문자시스템의 경우 지연시간이 1초 이내로 충분히 빨랐다. 화재 영상의 화재 탐지 실험을 통해 우리의 화재 탐지 기술의 정확성을 입증하였으며, 화재 탐지 시간과 알림서비스 시간을 측정해 화재 발생 후 알림 전송까지의 시간도 확인해보았다. 본 논문의 AI 화재 탐지 및 알림서비스 시스템은 과거의 화재탐지 시스템들보다 더 정확하고 빨라서 화재사고 시 골든타임 확보에 큰 도움을 줄 것이라고 기대된다.

▸ **주제어**: 화재 탐지, YOLOv3, EfficientDet, 알림서비스, 실시간

I. Introduction

화재 사고는 사람의 생명과 재산에 손실을 입히기 때문에 발생하기 전에 탐지하는 것이 매우 중요하다. 오늘날 가장 많이 쓰이는 화재 감지기는 대부분 실내용이며 주로 화재에 의해 생성된 연기와 열을 인식해 반응한다. 하지만 이러한 화재 감지기는 실외 화재 탐지에는 적합하지 않으며, 천장이 높은 공간에서는 화재의 규모가 어느 정도 커지고 나서만 화재로 인식해서 화재 탐지에 많은 시간이 소요될 수 있다. 화재 감지기가 대형 사고로 이어지는 초기 화재를 제대로 인식하지 못한다면 인명을 구하고 물질적 피해를 최소화할 골든타임을 상실할 수도 있다.

CCTV를 활용한 비전 기반의 화재 탐지는 초기 화재도 탐지 가능하며 실외의 환경에서는 광범위한 영역을 관찰할 수 있는 이점을 가진다. 기존에 보안 목적으로 설치된 CCTV를 그대로 사용함으로써 적용 범위를 극대화할 수 있고 하드웨어 장치를 추가로 설치하지 않아 설치비용을 절약할 수 있다.

CCTV를 이용한 AI 화재 탐지 기술은 이미 수많이 존재했다. 그 기술은 크게 분류(Classification) 기술과 탐지(Detection) 기술로 나뉜다. 하지만 단순 분류 기술은 화재를 제대로 판단하지 못했으며, 규모가 작은 초기 화재는 거의 탐지하지 못했다. AI를 이용한 탐지 알고리즘이 적용된 기술도 탐지 모델의 성능 부족으로 거짓 양성률(False Positive Rate)과 거짓 음성률(False Negative Rate)가 상당히 높았다. 정확도가 높은 탐지 알고리즘을 쓰면 지연 시간 없는 실시간 탐지가 불가능했다.

화재를 실시간으로 정확히 탐지하는 건 쉬운 일이 아니다. 지연 시간 없이 정확하게 탐지하기 위해서 정확하고 빠르게 연산하는 합성곱 신경망(Convolutional Neural Network, CNN)을 선택하였다. 그 중에서도 가장 인지도 높은 YOLO(You Only Look Once)v3 기반의 탐지기와 이미지 분류 알고리즘 중 현재 가장 정확하다고 알려진 EfficientNet 기반의 탐지 알고리즘인 EfficientDet을 이용해 실시간 화재 탐지율이 높은 기술을 구현해보았다.

또한, 화재 탐지 후 알림서비스를 다각화해 다양한 방법으로 화재 탐지 결과를 전송하고 공유하는 시스템을 만들었다. 문자서비스와 웹 알림, 앱 알림, 메일 전송이 그 4가지 알림서비스이다. 문자서비스는 Amazon Simple Notification Service를 사용했고, 웹 알림은 Web Push Notification 서비스를, 앱 알림은 Firebase Cloud Messaging을, 메일 전송은 Amazon Simple Email Service를 사용했다. 이 4가지의 알림서비스를 다수의 사람들에게 동시에 보낼 수 있는 게 우리 알림서비스의 가장 큰 장점이다.

화재 탐지 성능을 평가하기 위해 20편 이상의 영화, 드라마, 다큐멘터리에서 가져온, 화재가 있는 영상 클립 100개에 대해서 높은 화재 탐지율을 확인했다. 웹캠과 라이터를 이용해 실시간 화재 탐지 시간 및 문자 알림 서비스의 전송 시간을 확인하였다.

II. Preliminaries

1. Image Classification & Object Detection

1.1 Image Classification

이미지 분류는 머신러닝에서 지도학습의 한 방법이다. 라벨링된 데이터를 이용해 지도학습을 시킨 후 새로운 데이터가 학습된 데이터들 사이에서 어느 클래스에 속하는지를 클래스에 대한 확률값으로 구한다.

이미지 분류는 학습 데이터의 수가 충분하고 데이터의 질이 좋으며 클래스별로 모든 데이터가 비슷한 수로 구성되어 있다면 거의 완벽에 가까운 성능을 보여줄 수 있다.

합성곱 신경망은 빠르긴 하지만 여전히 많은 연산량이 필요하다. CPU를 하나 정도만 가지고 있거나 병렬연산이 가능한 GPU가 하나도 없는 대부분의 장치에서는 일반적인 분류 모델이 제대로 작동할 수 없다. 따라서 이런 장치에 대해서는 연산량이 더 적지만 정확성을 어느 정도 유지하도록 설계된, 가벼운 분류 모델을 적용해야 한다.

최근에도 많이 쓰이는 가벼운 이미지 분류 모델로는 MobileNet, ShuffleNet, SqueezeNet, NasNet-Mobile 그리고 EfficientNet-B0~B3 등이 있다. 먼저 MobileNet은 몇 가지 중요한 설계 결정을 기반으로 만들어진 효율적인 모델이다[1]. 모델의 크기와 지연 시간을 연산 정확도와 절충해 폭 승수와 해상도 승수를 사용해서 만들어진 더 작고 빠른 모델이다. ShuffleNet은 10-150 메가플롭스(MFLOPs)로 컴퓨터 성능이 매우 제한된 모바일 기기의 연산 효율성을 높여주기 위한 모델이다[2]. 이 아키텍처는 Point-wise Convolution (1x1 Convolution)과 Channel Shuffle의 두 가지 작업을 활용해 정확성을 유지하면서 연산량을 크게 감소시킨다.

SqueezeNet은 ImageNet이라는 1400만 개 이상의 이미지들의 파라미터를 50배 감소시켜 압축되지 않은 AlexNet과 같은 수준의 정확도를 보여주는 모델이다[3]. 한편 NasNet은 규모를 변형시킬 수 있는 합성곱 신경망 구조로 강화학습을 사용해 최적화된 기본 구성 블록으로 구성되어 있다[4]. NasNet-Mobile은 NasNet의 모바일 버전으로 530만 개의 파라미터들과 5억 6400만 개의 단일 곱셈-누산기(Multiply Accumulates, MACs)로 구성된다[5, 6].

1.2 EfficientNet

EfficientNet은 여러 가지의 버전이 있는데 그 중 EfficientNet-B0~B3는 파라미터의 수가 적은 가벼운 모델이다[7]. B0 버전에서 B3 버전으로 갈수록 파라미터 수는 조금씩 증가하는 반면 정확성은 평균 약 1.5%씩 증가한다[7].

1.3 Object Detection

물체 탐지(Object Detection)는 이미지 분류에서 한 단계 더 나아간 방법으로 대상의 위치를 바운딩 박스(Bounding Box)와 같은 형태로 예측하는 방법이다[8]. 이미지 분류에 박싱 회귀가 추가된 문제로 보면 된다.

물체 탐지는 이미지 분류와 달리 아직 모든 상황에 대해 완벽하지 않다. 특정 환경에서는 완벽해 보일 수 있어도 다른 배경화면에서는 성능이 저하될 수 있다. 따라서 탐지기와 관련해서 아직도 연구가 활발하게 진행 중이며 끊임없이 성능을 개선하고 있다.

1.4 YOLOv3

YOLO(You Only Look Once)v3은 정확도가 높은 가장 인지도 높은 탐지기들 중 하나이다. 다른 탐지기들과 비교했을 때 mAP(mean Average Precision)이 올라감과 동시에 여전히 낮은 추론 시간(Inference Time)을 유지하여 높은 FPS(Frames Per Second)을 보이는 게 특징이다[8].

이전 버전의 YOLO에 비해 v3의 버전에서 개선된 점들은 크게 4가지가 있다[9]. 첫 번째는 바운딩 박스 예측이다. YOLOv3는 로지스틱 회귀 분석을 사용하여 각각의 바운딩 박스에 대한 객관성 점수를 예측한다. 바운딩 박스의 폭과 높이는 클러스터 중심으로부터 특정한 간격을 띄우면서 지정된다. 시그모이드 함수를 사용해서 필터 적용 위치에 바운딩 박스의 중심 좌표를 예측하는 방식이다.

두 번째는 클래스 예측이다. 각 박스는 다중 라벨 분류를 사용해 바운딩 박스가 포함될 수 있는 클래스를 예측한다. 이때 소프트맥스를 사용하지 않고 독립적인 로지스틱 분류기를 사용한다. 학습 중에 클래스 예측을 위해서 이진 교차 엔트로피 손실법(Binary Cross-Entropy Loss)을 사용한다.

세 번째는 척도를 이용한 예측이다. YOLOv3는 박스들을 3가지의 다른 척도로 예측한다. 기본 특징 추출기에서 몇 개의 컨볼루션 층들을 추가한다. 이 중 마지막에서 3D 텐서 인코딩 바운딩 박스, 객체성, 클래스 예측 등을 예측한다. 3개의 박스를 예측하기 위해서 4개의 바운딩 박스 오프셋, 1개의 객체성 예측 그리고 80개의 클래스 예측에 대해 텐서는 $N \times N \times [3 * (4 + 1 + 80)]$이다.

4번째는 Darknet-53의 특징 추출기이다. 이 특징 추출기를 위해서 새로운 신경망을 사용했다. 이 신경망은 큰 틀에서 기존에 있었던 YOLOv2와 Darknet-19 등의 신경망의 하이브리드 접근 방식이다. 이 신경망은 연속적인 3×3과 1×1의 컨볼루션 층을 사용하지만, 단축 연

결도 가지고 있어서 크기가 크다. 총 53개의 컨볼루션 층을 가지고 있기 때문에 Darknet-53으로 명명했다. 이 신경망은 다른 분류기들과 비슷한 성능을 내지만 부동소수점 연산이 적은 동시에 초당 부동소수점 연산 속도가 빠른 게 특징이다.

2. AI Fire Detection

2.1 YOLOv3 기반의 Fire Detector

YOLOv3 알고리즘을 이용해 2000장 이상의 서로 다른 화재 이미지들을 학습시켰다. 모든 화재 이미지들은 구글과 네이버 포털에서 'Fire Disaster', '화재' 등의 키워드를 검색해 찾아냈다. 실제 화재 이미지가 아닌 데이터들과 화면 대부분이 화재로 가득 차 있는 이미지 데이터들은 검열해 제거했다. 학습 성능을 높이기 위해 모든 이미지에 대해서 2장씩 이미지의 밝기를 랜덤으로 조절하여 데이터 수를 늘리는 데이터 증강(Data Augmentation)을 진행했다. 그 다음 바운딩 박스를 그려서 박스의 좌표를 구할 수 있는 여러 프로그램 중 DarkLabel을 사용해 모든 이미지에 바운딩 박스를 그렸다. 불균형적인 모양의 화재 이미지는 추가로 삭제하거나 화재 부분의 75% 이상이 포함되도록 바운딩 박스를 그렸다.

2.2 EfficientDet 기반의 Fire Detector

EfficientNet 기반의 모델인 EfficientDet에 화재 이미지를 학습시켰다[10]. 500여 장의 라벨링된 데이터를 사용하였으며, 데이터 증강을 통해 전체 데이터 수를 3배로 늘렸다. Training과 Validation, Test를 7:2:1의 비로 분할했다. 학습 가중치(Weight) 모델의 학습은 COCO Dataset [11]에 대해 학습되어 있는 사전학습된 가중치로부터 시작했다. 복합 계수(Compound Coefficient)가 높을수록 필요한 GPU 메모리가 높아졌고, FPS는 감소했으며, 모델 수렴까지의 시간이 오래 걸렸다. YOLOv3 기반의 화재 탐지기를 학습시킬 때와 같은 하드웨어 환경에서 학습을 진행하였다.

3. Notification Service

3.1 Text Notification Service

Amazon Simple Notification Service(아마존 SNS)는 200여 개의 국가와 지역에 메시지를 보낼 수 있도록 지원하는 서비스이다[12]. Fig. 1.은 우리가 구현한 아마존 SNS의 수신 화면이다. 아마존 SNS를 사용해 문자 메시지를 SMS 지원 장치로 보낼 수 있다. 문자 메시지를 보낼 때 전화번호를 입력해 보내거나, 여러 전화번호를 등록하고 한 번에 메시지를 보낼 수 있다.

AWS 계정에 대한 SMS 기본 설정을 설정해서 사용 사례와 예산에 맞게 SMS 전송을 조정할 수 있다. 예를 들어, 메시지가 비용이나 신뢰할 수 있는 배달에 최적화되어 있는지 여부를 선택할 수 있다. 또한 개별 메시지 배달에 대한 지출 할당량과 AWS 계정에 대한 월별 지출 할당량을 지정할 수 있다.

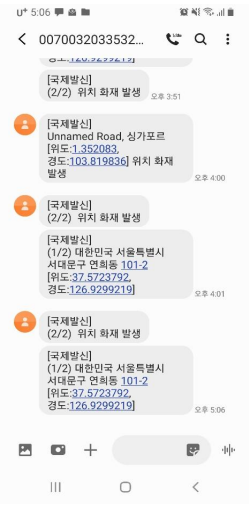


Fig. 1. Amazon SNS

3.2 Web Push Notification

AWS SDK를 사용하면 Amazon Pinpoint API를 통해서 트랜잭션 푸시 알림을 특정 디바이스 식별자로 보낼 수 있다[13]. 여기서 Amazon Pinpoint는 Firebase Cloud Messaging (FCM), Apple Push Notification Service (APNs), Baidu Cloud Push와 Amazon Device Messaging (ADM) 채널을 지원하는데, 이러한 푸시 알림 서비스를 통해서 푸시 알림 전송이 가능하다. Fig. 2.는 우리의 웹 푸시 알림(Web Push Notification)의 입력 화면과 수신 화면이다.

Firebase Cloud Messaging(FCM) 서비스를 통해 푸시 알림을 보낼 때 Amazon Pinpoint API 호출 시 Google Cloud Messaging(GCM)이라는 이름의 서비스를 사용한다. GCM은 구글에서는 2018년 4월 10일에 중단한 서비스이다. 하지만 GCM 서비스를 중단하기 전에 작성된 API 코드와의 호환성 유지를 위해 FCM 서비스를 통해 전송하는 메시지에 GCM 서비스 이름을 여전히 사용한다.

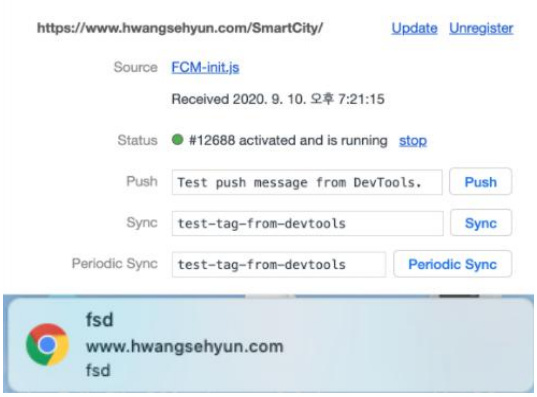


Fig. 2. Web Push Notificaton

3.3 Firebase Cloud Messaging (Android)

FCM은 최대 4KB의 푸시 알림과 작은 메시지를 안드로이드, iOS, 웹 등 다양한 플랫폼에 전송하는 툴이다[14]. Fig. 3.는 우리 시스템의 FCM 안드로이드 알림 수신 화면이다. Firebase는 알림을 작동시키는 가장 간단한 방법 중 하나로 많은 모바일 프로젝트에서 유용하게 사용 가능하다.

FCM 구현에는 송수신을 위한 두 가지 주요 구성요소가 포함된다[13].

1. Firebase용 클라우드 함수들이나 앱 서버와 같이 메시지를 작성, 타겟팅, 전송 가능한 신뢰할 수 있는 환경
2. 해당 플랫폼 별 전송 서비스를 통해 메시지를 수신하는 iOS, 안드로이드 또는 웹 클라이언트 앱

Firebase Admin SDK 또는 FCM 서버 프로토콜을 통해 메시지를 보낼 수 있다. 알림 작성기를 사용하면 기본 제공되는 강력한 타겟팅 및 분석 기능이나 커스텀 세그먼트를 사용하여 마케팅 또는 참여 메시지를 테스트하고 전송할 수 있다.

FCM을 이용한 앱 서버의 구현 방법은 다음과 같다[14].

1. 플랫폼에 맞는 설정 안내에 따라 앱에서 Firebase 및 FCM을 설정한다.
2. 클라이언트 앱에 메시지 처리, 주제 구독 로직 또는 기타 선택적 기능을 추가합니다. 개발 중에는 알림 작성기에서 테스트 메시지를 쉽게 보낼 수 있다.
3. 인증, 보내기 요청 작성, 응답 처리 등을 수행하는 전송 로직을 만들 때 Firebase Admin SDK를 사용할지 아니면 서버 프로토콜 중 하나를 사용할지 결정한다. 그런 다음 신뢰할 수 있는 환경에 로직을 구축한다. 클라이언트 애플리케이션에서 업스트림 메시징을 사용하려면 XMPP(Extensible Messaging and Presence Protocol)를 사용해야 한다[15].

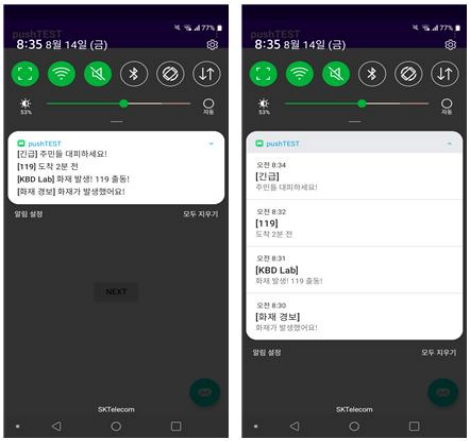


Fig. 3. Firebase Cloud Messaging

3.4 Mail Transfer Service

Amazon Simple Email Service(아마존 SES)는 개발자가 모든 애플리케이션 안에서 이메일을 보낼 수 있는 경제적이고, 유연하며, 확장성 있는 메일 서비스이다[16]. 아마존 SES를 통해서 트랜잭션, 마케팅 또는 대량 이메일 커뮤니케이션을 포함한 다수의 이메일 사용 사례를 지원할 수 있다. 아마존 SES는 발송률을 높이고 발신자 평판을 보호하는 데 도움이 되는 유연한 IP 배포 및 이메일 인증 옵션과 함께 전송 분석을 통해 각 이메일의 영향을 측정하는 기능을 제공한다. 이 아마존 SES를 사용하면 대규모 이메일을 안전하게 전 세계로 보낼 수 있다. Fig. 4.는 우리의 이메일 전송 서비스의 메일 수신 화면이다.

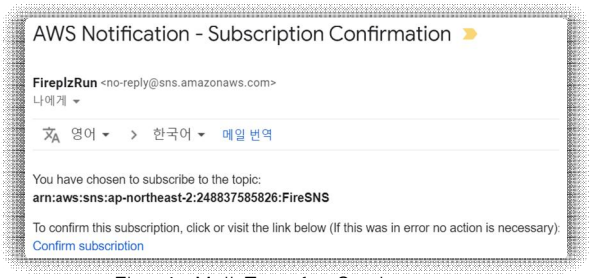


Fig. 4. Mail Transfer Service

III. Integrated System

우리의 화재 탐지 기술은 로컬에서 작동되는 YOLOv3 기반의 탐지 알고리즘과 웹에서 작동하는 EfficientDet 기반의 탐지 알고리즘이다. YOLOv3은 EfficientDet 대비 처리 속도가 더 빨라서 로컬용으로 적합하고, Efficient- Det 기반의 탐지 알고리즘은 로컬보다 더 빠른 클라우드에서 작동하기 때문에 무거운 모델의 약점을

충분히 극복할 수 있다.

우리의 통합 시스템을 개략적으로 나타내면 Fig. 5와 같다. 기본적으로 AWS(아마존 Web Service)의 서비스들을 사용하여 구현하였다. 크게 영상 수신 모듈, 화재 탐지 모듈, 알림과 표출 및 DB 모듈로 구성된다.

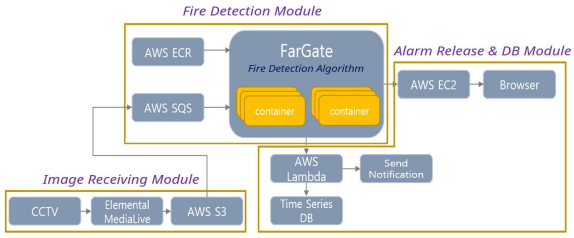


Fig. 5. Integrated System

가장 먼저 영상 수신 모듈에서 CCTV 영상은 영상 처리 모듈로 스트리밍되고 일정한 주기에 따라 프레임 이미지를 캡처한다. 이때 영상 처리 모듈은 AWS Elemental MediaLive를 사용하여 입력 이미지에 대해 전처리를 진행한다[17]. 이렇게 전처리를 마친 이미지는 Cloud Storage에 업로드된다. AWS S3 서비스를 사용할 수 있다[18]. S3에 저장된 이미지를 직접 화재 탐지 코드에서 불러오게 되면 CCTV가 많아질 경우 상대적으로 높은 용량을 가진 이미지 데이터의 대기열 입출력 관리가 쉽지 않다. 이런 경우를 대비해 Amazon SQS(Simple Queue Service)를 사용하여 이미지 자체가 아니라 이미지의 S3 URL 주소만을 대기열에 보낸다[19]. 이렇게 대기열을 지난 URL 주소는 화재 탐지 알고리즘에 입력으로 들어간다.

다음으로 화재 탐지 모듈에서 화재 탐지 알고리즘을 컨테이너화함으로써 높은 확장성을 확보하고 어떤 환경에서도 작동할 수 있게 할 수 있다. 화재 탐지 알고리즘과 작동 환경을 도커(Docker) 이미지로 빌드하여 도커 허브에 저장하여 불러올 수 있게 한다[20]. 클라우드 상에서 해당 컨테이너의 알고리즘을 돌리기 위해 AWS Fargate를 사용한다[21]. 다수의 컨테이너를 생성하여 SQS로부터 받는 URL을 순차적으로 입력하게 하여 작동 속도를 높일 수 있다.

마지막으로 알림과 표출 및 DB 저장 모듈이다. 화재 탐지 알고리즘은 Output으로 탐지 여부와 바운딩 박스를 출력하게 되는데 이 정보는 두 갈래로 보내진다. 첫째는 이벤트 기반 처리 서비스인 AWS 람다(Lambda)[22]이고, 둘째는 표출을 위한 프론트엔드이다. AWS 람다에서는 다시 탐지 데이터 저장을 위한 DB와 클라이언트로의 알림 두 갈래로 데이터를 전송한다.

IV. Experiment

1. Experimental Equipment

Table 1.에 우리가 사용한 장비를 정리하였다. Computer no.1과 no.2는 딥러닝 학습 및 알고리즘 개발을 위해서 쓰였으며, Computer no.3는 주로 AWS 서비스와 FCM 서비스의 구축을 위해서 쓰였고, Computer no.4를 사용해 화재 탐지 및 알림서비스 시스템으로 두 기술을 통합하였다.

Table 1. Equipment Used

	Computer no.1, no.2	Computer no.3	Computer no.4
OS	Linux Ubuntu 16.04	Windows 10	Windows 10
CPU	i9-7980XE	i7-8700	i9-9900K
Mainboard	X299 XPOWER GAMING AC	GIGABYTE B360M DS3H Durable Edition JC Hyun	ASRock B360M PRO4 Aswin
VGA	NVIDIA GeForce GTX Titan Xp * 2	EMTEK GeForce RTX 3070 BLACK EDITION OC D6 8GB	PALIT GeForce RTX 2070 Dual D6 8GB
RAM	total 96GB	Samsung DDR4 16G PC4-21300	Samsung DDR4 16G PC4-21300 8G * 2 Dual Channel
SSD	Samsung 970 Evo M.2 NVMe 1TB	Samsung 970 Evo Plus M.2 2280 500GB	Samsung 860 EVO (1TB)
Count	2	1	1

2. Test using Video Clips

화재 탐지 성능을 평가하기 위해 넷플릭스에 있는 20편 이상의 영화, 드라마, 다큐멘터리에서 가져온, 화재가 있는 평균 26.52초짜리 영상 클립 100개를 테스트 데이터로 가져왔다. Minimum Percentage Probability를 20으로 설정하고 YOLOv3로 학습시킨 알고리즘에 대해 화재 탐지율을 확인했다. 그 결과 100개의 영상 클립들 중 95개는 화재를 제대로 인식했으나, 나머지 5개는 화재를 탐지하지 못했다.(False Negatives)

Fig. 6.와 같이 13번(3:35~4:02) 영상 클립처럼 원거리의 화재는 잘 탐지하지 못했고, Fig. 7.과 같이 24번

(6:55~7:15) 영상 클립처럼 화재가 너무 밝아 그 주변을 모두 밝게 만드는 경우에도 화재를 탐지하지 못했다. 또 Fig. 8.과 같이 34번(11:18~11:42) 영상 클립처럼 적절한 규모의 화재들이 있으나 주변에 소방차 사이렌 등이나 전등, 자동차 전조등 등이 있어 군데군데가 밝은 경우 이 화재들을 제대로 식별하지 못했다. Fig. 9.과 같이 79번(34:56~35:10) 영상 클립처럼 한 화면을 거의 채울 수 있을 만큼 화재의 규모가 큰 경우 화재를 제대로 발견해내지 못했으며, Fig. 10.과 같이 96번(41:23~42:09) 영상 클립처럼 연기로 인해 뿌옇게 보이는 경우에도 연기 속 화재를 감지하지 못했다.

Fig. 6. Image Clip no.13 (Faired)

Fig. 7. Image Clip no.24 (Faired)

Fig. 8. Image Clip no.34 (Faired)

Fig. 9. Image Clip no.79 (Faired)

Fig. 10. Image Clip no.96 (Faired)

이보다 더 큰 에러는 거짓 양성률이다. 화재가 아닌 사건에 대해 화재로 판단하는 경우였는데 특히 낮인데 매우 밝은 상황이나 밤에 비춰진 전등이나 전조등의 경우 화재와 구별하지 못하고 화재로 인식했다. 물론 이 부분은 최신 탐지 알고리즘의 사용이나 추가적인 데이터 증강으로 학습 성능을 증가시켜 충분히 해결 가능할 것으로 생각한다.

3. Real-time Fire Detection Using Webcam

Fig. 11.은 웹캠을 이용한 실시간 화재 탐지의 모습이다. 웹캠과 라이터를 이용해 실시간 화재 탐지 시간 및 문자 알림 서비스의 전송 시간을 확인하였다. 화재 탐지 시간은 라이터를 켠 상태에서 웹캠을 실행시켜 화재를 탐지하기 시작하는 시간까지를 측정하였고, 알림 서비스의 전송 시간은 화재를 탐지하기 시작하는 시간부터 문자 알림이 도착하는 시간을 스톱워치로 측정하였다. 측정은 각각 5번씩 시행되었다.

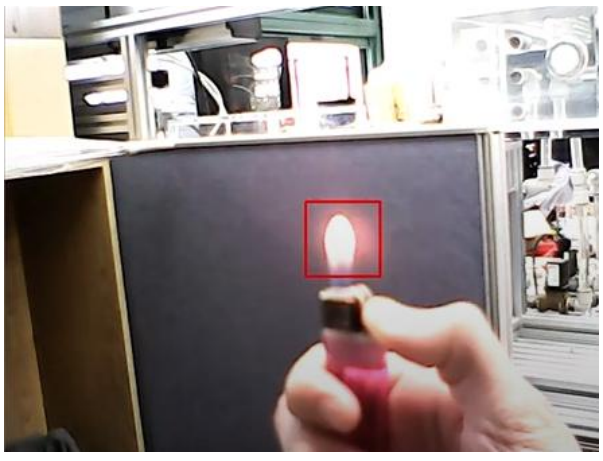


Fig. 11. Real-time Fire Detection Using Webcam

V. Results

YOLOv3 기반의 화재 탐지 기술은 배경의 영향을 많이 받는다. 주변이 너무 밝거나 흐리면 화재 탐지 성능은 급감한다. 또 화재의 규모가 너무 크거나 작을 때에도 화

재를 제대로 감지하지 못했다. 이는 데이터의 화재 부분을 박스 바운딩할 때 주변이 밝거나 흐린 데이터는 전체 학습에 혼란을 줄 수 있다고 생각하고 제거했기 때문으로 생각한다. 화재를 인식할 수 있게 하는 파라미터를 조정해 화재 탐지율을 높이면 거짓 양성률도 높아지고 후처리를 통해 연산을 추가해줘야 한다. 하지만 화재의 경우 비교적 정적으로 확산되고 커지지만, 공기 대류에 의해 그 모습이 계속 바뀌기 때문에 화재에 대한 비(非)탐지율을 비(非)화재에 대한 탐지율보다 높게 설정하는 게 더 낫다고 판단하였다. 예를 들어 20 프레임의 일정한 시간 구간이 있다면 화재 판단은 첫 화재 탐지 프레임이나 화재 탐지의 연속적인 5 프레임에 대해서만 이루어지면 되고 나머지 프레임들은 상관없기 때문에 거짓 음성률을 거짓 양성률보다 약간 높게 설정해야 한다.

실제로 웹캠을 이용한 화재 탐지 및 알림 서비스의 테스트 결과 화재 탐지 프로그램이 모든 화재 프레임에 대해서 화재를 탐지하지는 않았지만, 평균 1.702초의 짧은 시간 안에 화재 탐지가 이루어졌고, 문자 알림 서비스의 경우 평균 0.764초의 지연 시간이 발생했다(Fig. 12.). 스마트폰 자체의 알림 지체 현상과 스탑워치를 이용해 사람이 직접 시간을 측정하였기 때문에 반응 시간의 영향이 포함되어 있다고 볼 수 있다. 따라서 순수 알림 전송 시간은 이보다 더 짧을 것으로 생각한다.

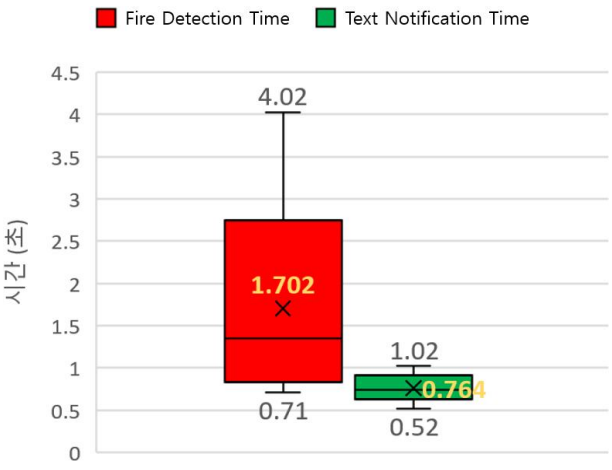


Fig. 12. Fire Detection & Text Notification Time

VI. Conclusions

YOLOv3 알고리즘으로 화재 데이터를 학습시켜 CCTV 영상으로부터 화재를 실시간으로 탐지하였다. 비록 비디오 클립 테스트에서 화재 인식이 안 되는 환경이 발견되었지만, 이 부분은 추후 새로운 탐지 모델을 이용하고 새로운 데이터를 사용해 학습하여 극복 가능할 것으로 보인다. 거짓 음성률 높아서 발생한 우려에 대해서는 화재 탐지 시간이 충분히 짧기 때문에 괜찮다고 판단하였다.

이 연구는 아직 진행 중인 연구이며, 가장 최신 물체 탐지 프레임워크인 YOLOv4를 이용해 거짓 양성률과 거짓 음성률을 줄일 계획이다. 또한 현재는 주간과 야간의 화재만 탐지하지만, 더 어려운 연기 탐지에 대해서도 학습시켜 결국엔 연기 및 화재 탐지 기술을 추구하고자 한다. 실증의 경우 테스트베드를 구축하여 실제 CCTV를 이용해 화재를 탐지하고 화재 발생 후 알림서비스 수신까지의 소요시간을 평가할 예정이다.

ACKNOWLEDGEMENT

This work was supported by Korea Agency for Infrastructure Technology Advancement(KAIA) (20NSPS-B159105-03).

REFERENCES

[1] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," arXiv preprint arXiv:1704.04861, Apr. 2017.

[2] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, Jian Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," CVPR 2018, pp. 6848-6856, June 2018, DOI: https://doi.org/10.1109/cvpr.2018.00716

[3] Md Zahangir Alom, Tarek M. Taha, Christopher Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Brian C Van Esesn, Abdul A S. Awwal, Vijayan K. Asari, "The history began from alexnet: A comprehensive survey on deep learning approaches," arXiv preprint arXiv:1803.01164, Sep. 2018.

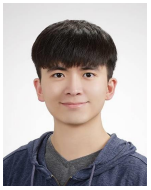
[4] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, Quoc V. Le, "Learning transferable architectures for scalable image recognition," CVPR 2018, pp. 8697-8710, June 2018, DOI: https://doi.org/10.1109/cvpr.2018.00907

[5] Xu Qin, Zhilin Wang, "NASNet: A Neuron Attention Stage-by-Stage Net for Single Image Deraining," arXiv preprint arXiv:1912.03151, May 2019.

[6] Nasnetmobile, https://www.mathworks.com/help/deeplearning/ref/nasnetmobile.html

[7] Mingxing Tan, Quoc V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," arXiv preprint arXiv:1905.11946, Sep. 2019.
[8] Paul Viola, Michael J. Jones, "Robust real-time object detection," International journal of computer vision, 4.34-47: 4, Feb. 2001.
[9] YOLO: Real-Time Object Detection, https://pjreddie.com/darknet/yolo/
[10] Mingxing Tan, Ruoming Pang, Quoc V. Le, "Efficientdet: Scalable and efficient object detection," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10781-10790, 2020.
[11] Common Objects in Context, https://cocodataset.org/#home
[12] Amazon Simple Notification Service, https://aws.amazon.com/ko/sns/?whats-new-cards.sort-by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc
[13] Sending a test push notification, https://docs.aws.amazon.com/pinpoint/latest/userguide/messages-mobile.html
[14] Laurence Moroney, "Firebase Cloud Messaging," The Definitive Guide to Firebase. Apress, Berkeley, CA, pp. 163-188, Nov. 2017, DOI: https://doi.org/10.1007/978-1-4842- 2943-9_9
[15] XMPP, https://xmpp.org/
[16] Amazon Simple Email Service, https://aws.amazon.com/ses/?nc1=h_ls
[17] AWS Elemental MediaLive, https://aws.amazon.com/ko/medialive/
[18] Jason Nadon, "Your Content Solution: An Introduction to AWS S3," Website Hosting and Migration with Amazon Web Services. Apress, Berkeley, CA, pp. 15-24, May 2017, DOI: https://doi.org/10.1007/978-1-4842-2589-9_3
[19] Hobin Yoon, Ada Gavrilovska, Karsten Schwan, Jim Donahue, "Interactive use of cloud services: Amazon sqs and s3," 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012). IEEE, pp. 523-530, May 2012, DOI: https://doi.org/10.1109/CC Grid.2012.85
[20] Price, Miles, "Docker : The Comprehensive Beginner's Guide to Take Control of Docker Programming [paper book]," Createspace Independent Publishing Platform, pp. 1-82, Dec. 2017.
[21] Deepak Vohra, "Amazon Fargate Quick Start Guide: Learn how to use AWS Fargate to run containers with ease," Packt Publishing Ltd, pp. 1-169, July 2018.
[22] Josef Spillner, Serhii Dorodko, "Java code analysis and transformation into AWS lambda functions," arXiv preprint arXiv:1702.05510, Feb. 2017.

Authors



You-min Na received the B.S. degree in Mechanical Engineering from Ajou University, Korea, in 2019. Mr. Na is currently a M.S. student in Mechanical Engineering at Yonsei University.

His research interests include Mechanical Design and Computer Vision.

Dong-hwan Hyun received the B.S. degree in Mechanical Engineering from Yonsei University, Korea, in 2020. Mr. Hyun is currently a M.S. student in Mechanical Engineering at Yonsei University.

His research interests include System Design and Computer Vision.

Do-hyun Park received the B.S. degree in Mechanical Engineering from Yonsei University, Korea, in 2020. Mr. Park is currently a M.S. student in Mechanical Engineering at Yonsei University.

His research interests include IoT and Computer Vision.

Se-hyun Hwang received the B.S. degree in Mechanical Engineering from Korea University, Korea, in 2019. Mr. Hwang is currently a M.S. student in Mechanical Engineering at Yonsei University.

His research interests include Web Technologies and Embedded Software.

Soo-hong Lee received the B.S. degree in Mechanical Engineering from Seoul National University, Seoul, Korea, in 1981. He received the M.S. degree in Mechanical Design from Seoul National University, Seoul, Korea, in 1983. He received the Ph.D. degree in Mechanical Engineering from Stanford University, USA, in 1991. Soo-hong Lee is currently a professor at the department of Mechanical Engineering at Yonsei University. His research interests include System Design, Process Planning Design, Embedded Software and IoT.