통계자료분석대회

CNN 활용
화재 감지 및 신고 시스템

#진강기동대 #진기현 #강동석

KGU

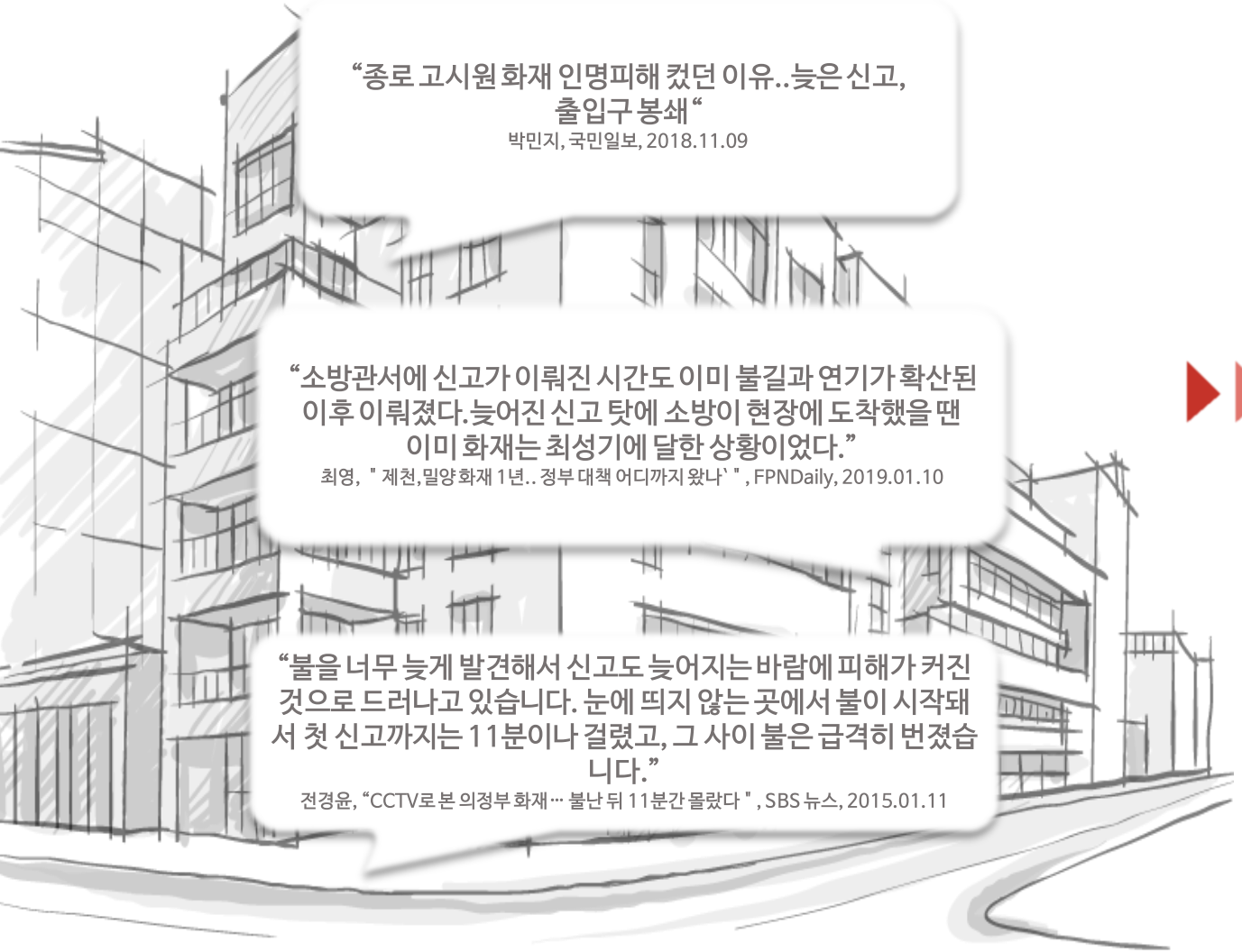
CONTENTS

KGU

분석 배경

1. 분석배경

1) 화재 조기신고



"종로 고시원 화재 인명피해 컸던 이유..늦은 신고, 출입구 봉쇄"
박민지, 국민일보, 2018.11.09

"소방관서에 신고가 이뤄진 시간도 이미 불길과 연기가 확산된 이후 이뤄졌다.늦어진 신고 탓에 소방이 현장에 도착했을 땐 이미 화재는 최성기에 달한 상황이었다."
최영, " 제천,밀양화재 1년.. 정부 대책 어디까지 왔나` " , FPNDaily, 2019.01.10

"불을 너무 늦게 발견해서 신고도 늦어지는 바람에 피해가 커진 것으로 드러나고 있습니다. 눈에 띄지 않는 곳에서 불이 시작돼서 첫 신고까지는 11분이나 걸렸고, 그 사이 불은 급격히 번졌습니다."
전경윤, "CCTV로 본 의정부 화재… 불난 뒤 11분간 몰랐다 " , SBS 뉴스, 2015.01.11

작은 화재가 큰 화재로 번지는 이유는 그 당시의 기후와 건물의 구조, 늦은 화재 진압 등이 있다.

특히나 화재 발생 후 최초 5분은 '골든 타임'으로 불릴 만큼 중요하다. 화재를 초기에 진압하기 위해 이러한 골든 타임을 지키려면 '신속한 화재 감지·신고'와 '빠른 출동·도착' 두 가지 조건을 맞추어야 한다.

본 분석은 화재 골든 타임의 두 가지 조건 중 "신속한 화재 감지·신고"에 초점을 맞추었다.

2) 화재 경보기의 문제점

주변에 환경 따라 달라지는 성능

화재 경보기는 주변 환경에 따라 오작동을 일으키는 경우가 많다.

대표적으로 무더위가 지속되는 여름철에 열을 감지해 빈번하게 화재경보기가 울리는 것을 볼 수 있다.

이러한 반복되는 오작동으로 화재 경보기의 전원을 꺼 놓는 사용자들 또한 생겨나고 있다.

오작동으로 인한 오인출동

국민안전처의 화재 출동 현황에 따르면 지난 5년간
약 500,000건의 출동 중 오인으로 인한 건수가
290,000건 이상으로 50%이상의 오인 출동이 발생하고 있고,
화재 시스템의 경보 오동작으로 인해 발생한 오인 출동의 비율은
매년 증가하고 있는 추세이다. 김영진, 김은경, CNN을 활용한 영상 기반의 화재감지,2016.09

3) 기술의 발달

카메라 보급률 증가

2019년 기준 대한민국은 스마트폰 보급률 95%, 세계 1위 스마트폰 사용 비율이 가장 높은 나라이다. 이러한 보급률에 맞추어 스마트폰의 카메라 기술도 나날이 발전하고 있다.

뿐만 아니라 '카메라'를 사용하는 자동차의 블랙박스 시장과 CCTV 시장 또한 꾸준히 성장세를 유지하고 있다.

영상인식 기술

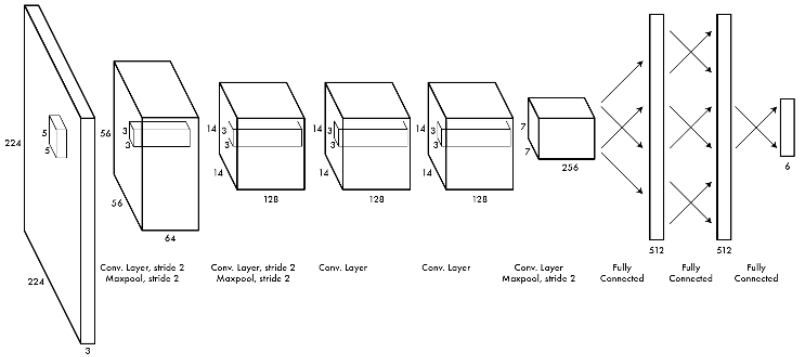
과거 숭례문 방화 사건 이후 숭례문에는 200만 화소의 화재 인식 알고리즘을 가진 CCTV가 설치되었다.

지능형 CCTV의 등장으로 각종 범죄에 대한 재빠른 조치가 가능하다.

분석목적

분석 목적

앞서 설명한 3가지의 배경으로 본 분석은 '카메라를 이용한 즉각적인 **화재 감지 및 신고 시스템**' 생성을 목적으로 한다. 분석에서 만든 모델을 통해, 카메라 기능을 가진 기기에 적용시켜 화재 진압 골든 타임을 지키는데 도움을 줄 수 있다.



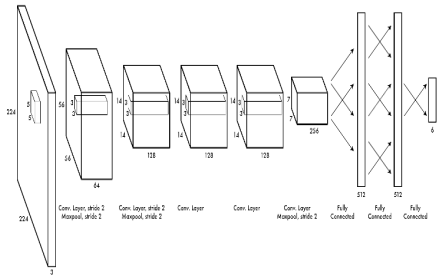
1) CNN: 화재 분류기 생성

2) OpenCV : 화재 영상 변환, 1차 분류기 생성

분석과정

3. 분석과정

1) FLOW CHART



CNN

화재 이미지 분석을 통한
화재 분류기 생성

OpenCV

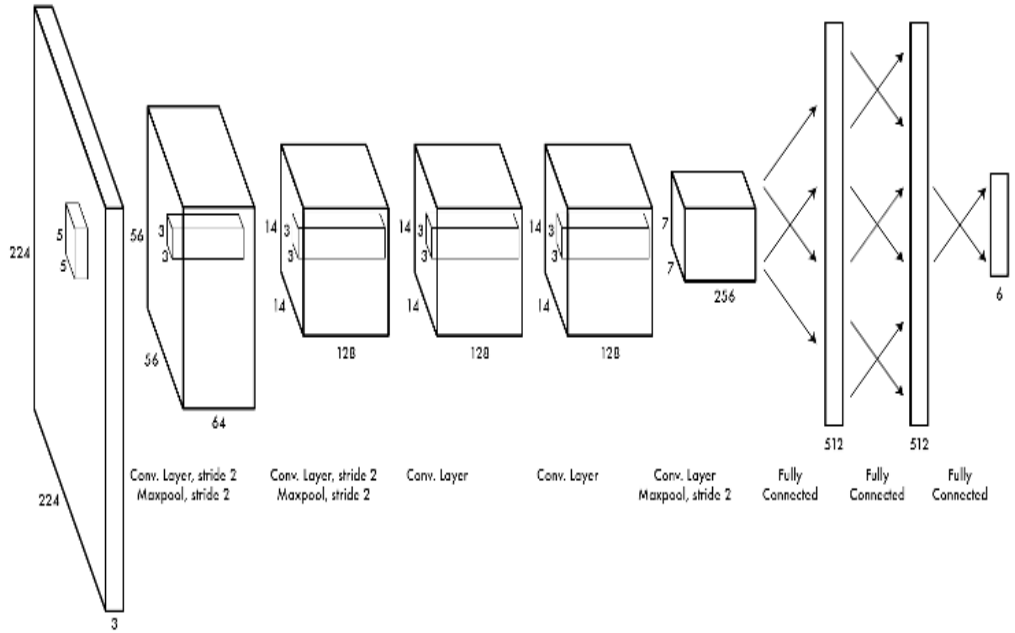
OpenCV를 사용한
1차 화재 인식, 라벨링

CNN 적용 및 신고

OpenCV에서 인식한 화재를
CNN 분류기를 통해 최종 인식

3. 분석과정

2) CNN



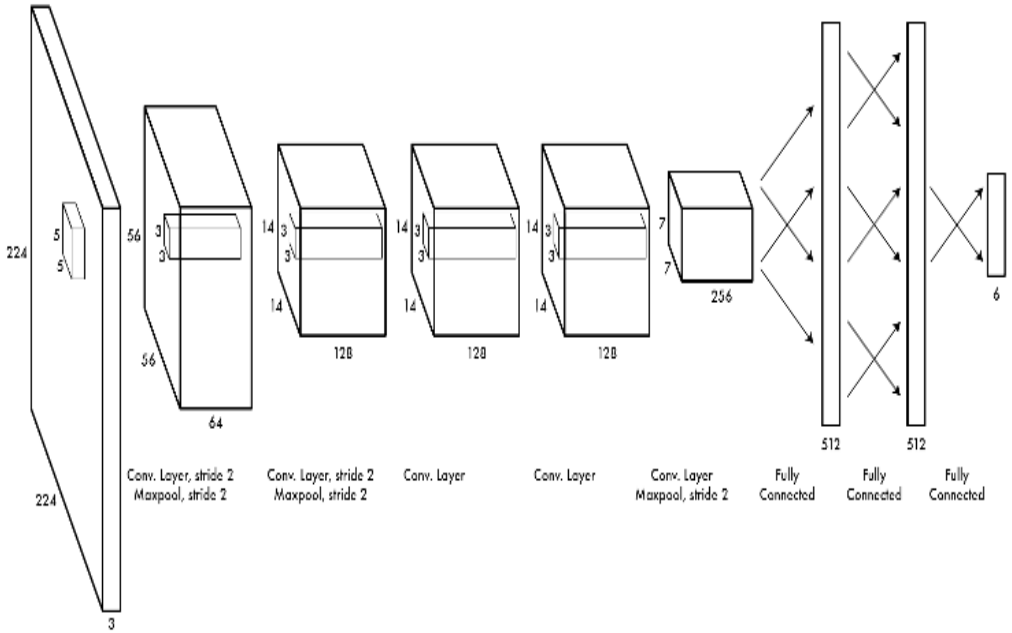
CNN(Convolutional neural network)이란?

딥러닝 알고리즘 중, 이미지 인식에 많이 사용되는 알고리즘이다.

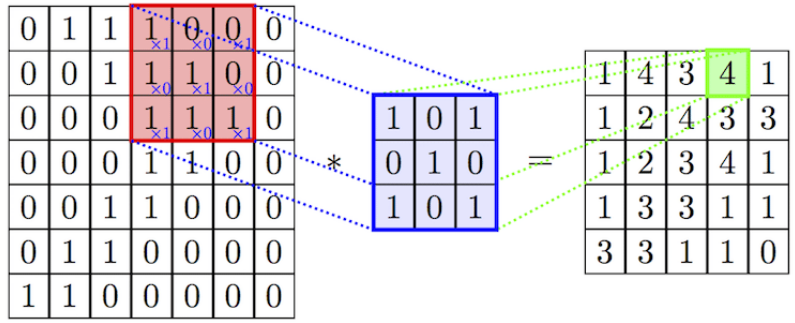
CNN은 Artificial Neural Network의 한 종류이며, 주로 Matrix 데이터나 이미지 데이터에 대하여 추출해내는 데에 쓰인다.

기존 신경망에 필터기술을 병합하여 신경망이 2차원 이미지를 잘 습득할 수 있도록 최적하시킨 알고리즘이다.

2) CNN



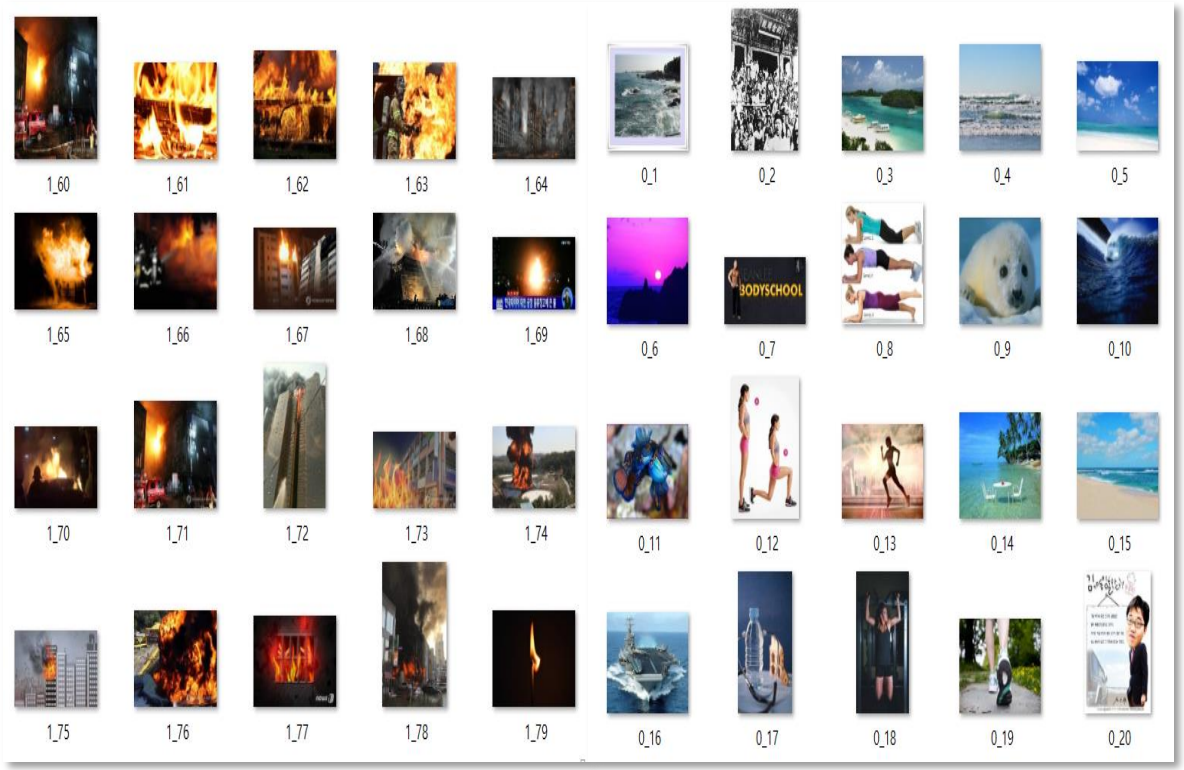
Convolution



28X28X3의 array형식으로 변환된 이미지를
Filter에 통과시켜 하나의 scala값으로
만드는 것을 Convolution 이라고 한다.

3. 분석과정

2) CNN

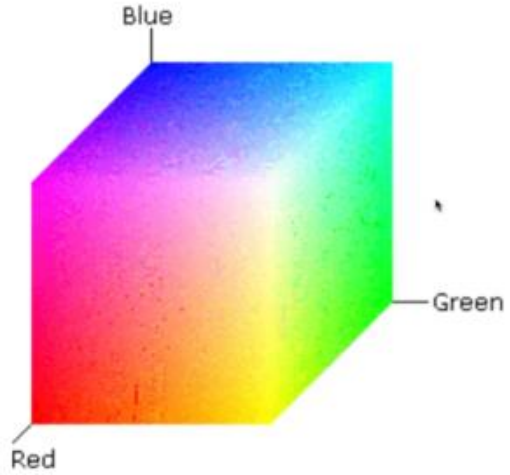


28X28픽셀로 변환

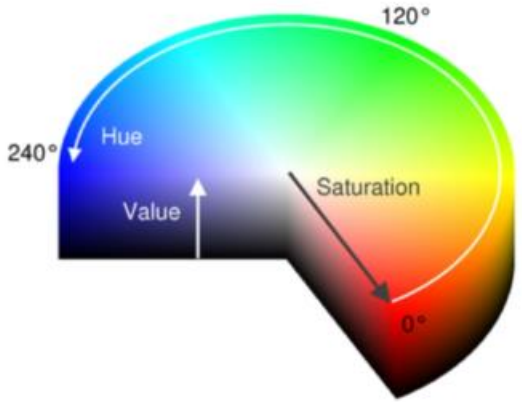
이미지 데이터

748장의 불 사진 중 598장을 train data 150을 test data로 지정하였고
871장의 불이 없는 사진 중 697장을 train data 174장을 test data로 지정하였다.
모든 이미지는 28X28픽셀로 변환하여 학습을 진행 하였다.

3) OpenCV



$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$



1) RGB

이미지를 저장하고 불러올 때 주로 사용한다.
특정한 좌표나 픽셀의 색을 3가지(Red, Green, Blue)
숫자로 표현이 가능하다.

2) Gaussian Blur

중심에 있는 픽셀에 높은 가중치를 부여해
Blurring(흐리게)한다.
이미지의 노이즈를 줄이기 위해 사용한다.

3) HSV

특정한 색상의 영역을 추출하고 싶을 때 사용한다.
명암이나 질감 같은 느낌까지 선택할 수 있다.
Hue는 색상 값. Saturation은 진함의 정도
Value는 밝은 정도를 나타낸다.

3) OpenCV

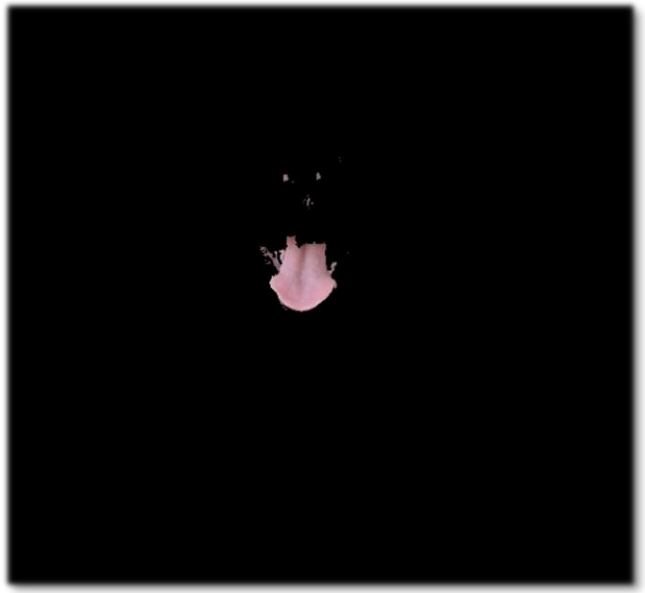
1) RGB

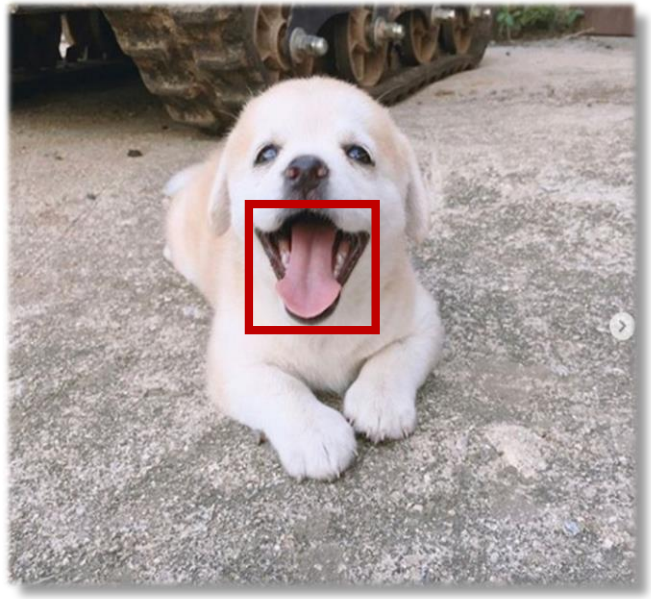
2) Gaussian Blur

3) HSV

*단순 BGR2HSV 사용시

3) OpenCV

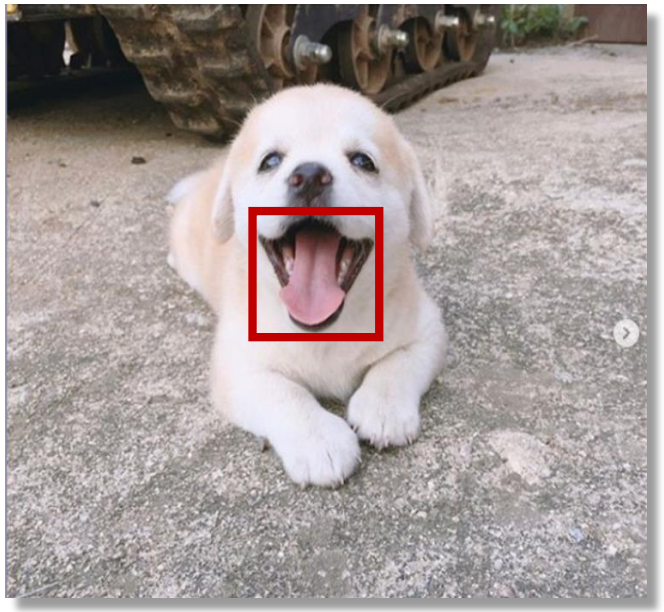




HSV 지정 값을 통한 라벨링

원하는 색의 HSV 값을 통해 인식된 색상의 부분 만을 라벨링한다.
예시의 색상은 H:156 S:34 V:107 상태에서 특정색만을 인식시킨 결과이다.

4) CNN 적용 및 신고



라벨링을통한 이미지 검출

0: Not Tongue

1: Tongue

라벨표시및신고

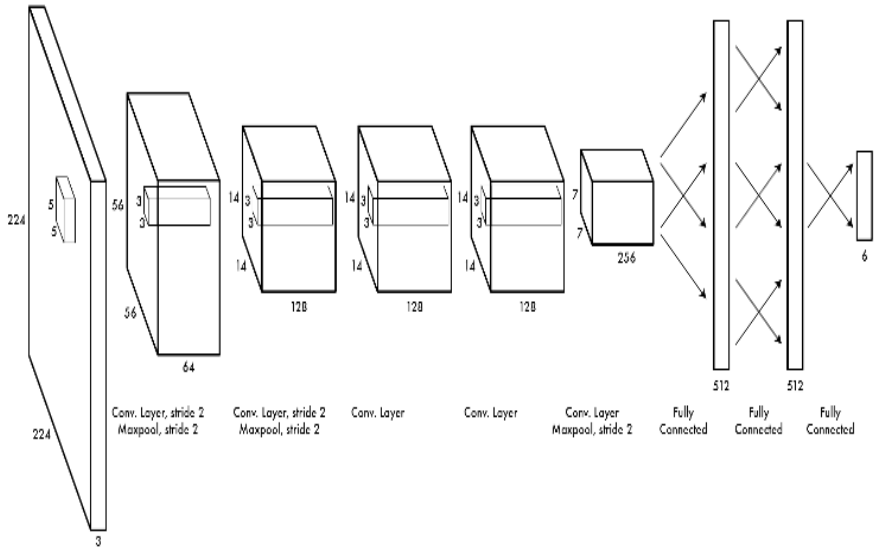
CNN 알고리즘을 통한 분류

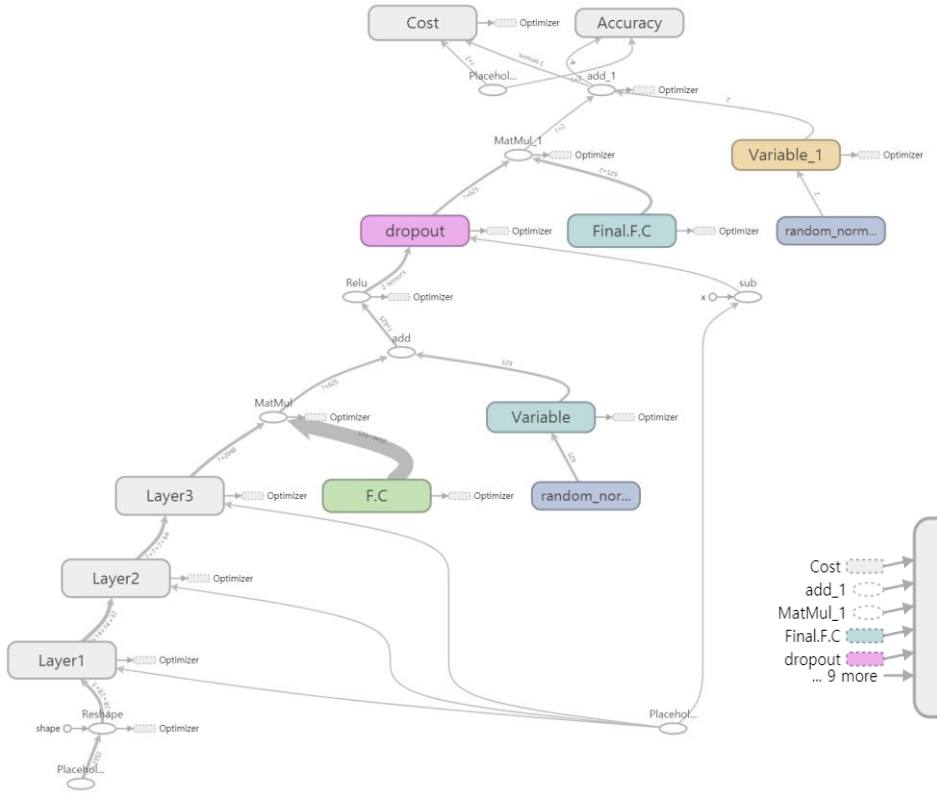
OpenCV를 통해 1차적으로 인식해 라벨링을 한 이미지에
CNN 알고리즘을 적용시켜 이미지를 다시 분류한다.

분석결과

4. 분석결과

1) CNN





TensorFlow

CNN을 Python의 Tensorflow를 이용해 구현 하였다. 우측은 Tensorflow Graph 이다.

1) CNN

$$cost(\hat{Y}, Y) = D(\hat{Y}, Y) = -\sum_i Y_i \ \log(\hat{Y}_i)$$

Cross-entropy cost function

Cross-entropy cost function

Softmax가 예측한 $\hat{Y}$ 와 실제 Y 값의 거리를 계산하는 것이다.
Cost 값이 0에 가까울 수록 예측값과 실제값의 거리가 가깝다는 것을 의미한다.

이는 학습(learning)이 잘되었고 정확도(accuracy)또한 높다는 것을 의미한다.

example

$$-\sum_i Y_i \ \log(\hat{Y}_i) = \sum_i Y_i \ * (-\log(\hat{Y}_i))$$

$$Y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\hat{Y} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \ \begin{bmatrix} 0 \\ 1 \end{bmatrix} * -log \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} * \begin{bmatrix} \infty \\ 0 \end{bmatrix} = 0, \ \ cost = 0$$

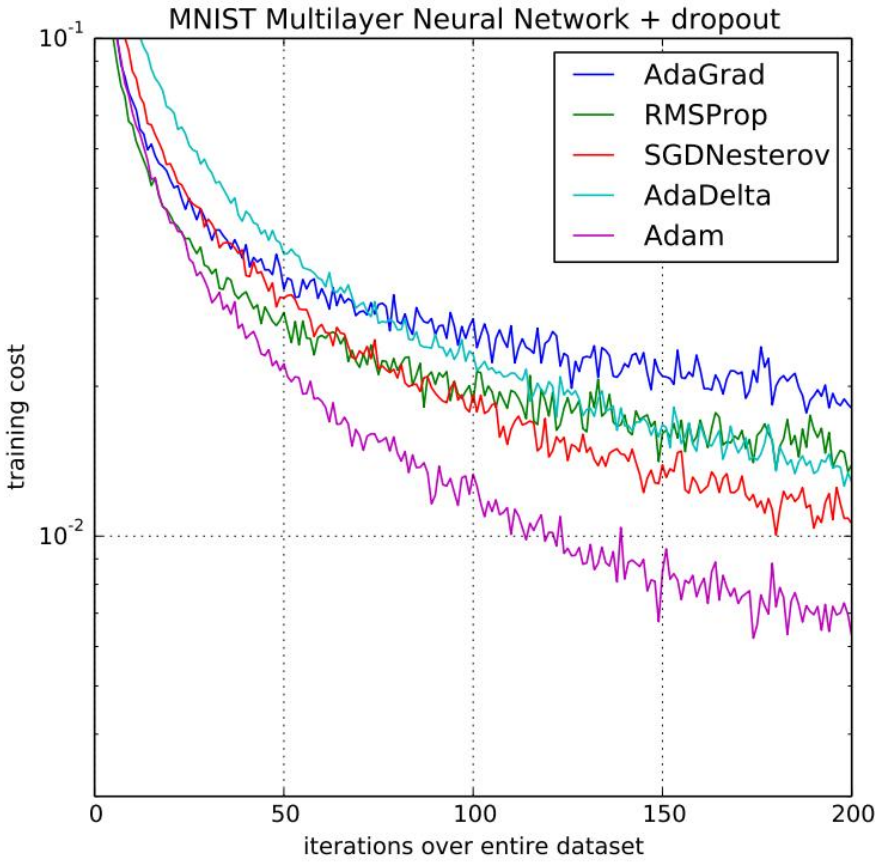
$$\hat{Y} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \ \begin{bmatrix} 0 \\ 1 \end{bmatrix} * -log \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} * \begin{bmatrix} 0 \\ \infty \end{bmatrix} = \infty, \ cost = \infty$$

4. 분석결과

1) CNN

Optimizer

최소가 되는 Cost함수 값을 찾는 것을 optimizer이라 한다.
Python에는 많은 optimizer가 내장 되어있는데 가장 성능이 좋다고 평가되는 Adam optimizer를 사용하였다.



4. 분석결과

1) CNN

Learning result

```
Learning started! Please Waite.
Step:     0     Cost: 0.781     Acc: 49.97%
Step:   100     Cost: 0.284     Acc: 87.92%
Step:   200     Cost: 0.132     Acc: 95.47%
Step:   300     Cost: 0.087     Acc: 96.91%
Step:   400     Cost: 0.052     Acc: 98.08%
Step:   500     Cost: 0.031     Acc: 98.90%
Step:   600     Cost: 0.019     Acc: 99.73%
Step:   700     Cost: 0.014     Acc: 99.73%
Step:   800     Cost: 0.011     Acc: 99.79%
Step:   900     Cost: 0.008     Acc: 99.93%
Learning Finished!
==========Classification==========

Accuracy: 0.9444444
Label:  [0]
Prediction:  [0]

==============================
```

explain

1295장의 이미지를 갖고 1000번의 학습을 시행 하였다.
100번의 시행마다 cost와 accuracy 값을 출력하였다.
학습이 진행될 수록 cost는 0으로 수렴하고 accuracy는 100%에 빠르게 가까워지고 있으므로 학습이 잘 이뤄졌다는 것을 확인 할 수 있다.

324장의 이미지로 학습 모델을 평가하였을 때 accuracy가 0.95이므로 결과적으로 이미지에서 불을 잘 검출 해 내는 알고리즘을 구현 하였다.

4. 분석결과

1) CNN

Learning result

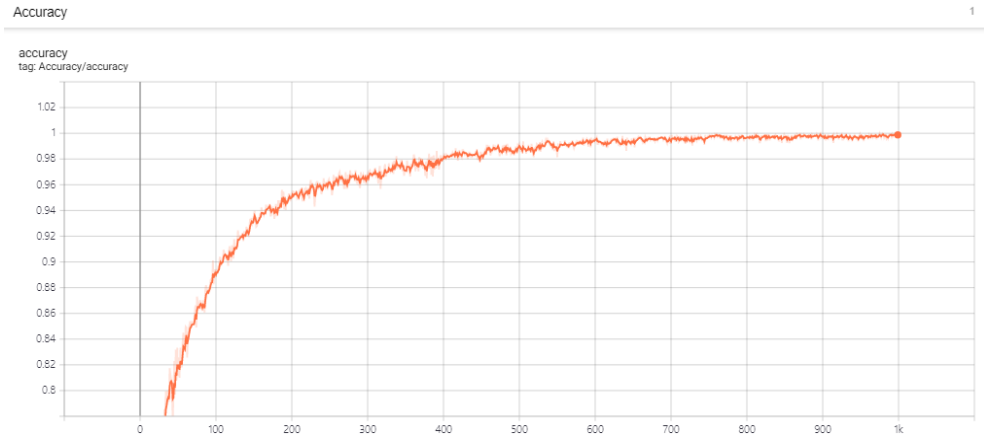
```
Learning started! Please Waite.
Step:     0     Cost: 0.781     Acc: 49.97%
Step:   100     Cost: 0.284     Acc: 87.92%
Step:   200     Cost: 0.132     Acc: 95.47%
Step:   300     Cost: 0.087     Acc: 96.91%
Step:   400     Cost: 0.062     Acc: 98.08%
Step:   500     Cost: 0.031     Acc: 98.90%
Step:   600     Cost: 0.019     Acc: 99.73%
Step:   700     Cost: 0.014     Acc: 99.73%
Step:   800     Cost: 0.011     Acc: 99.79%
Step:   900     Cost: 0.008     Acc: 99.93%
Learning Finished!
==========Classification==========

Accuracy: 0.9444444
Label:  [0]
Prediction:  [0]

==========================================
```

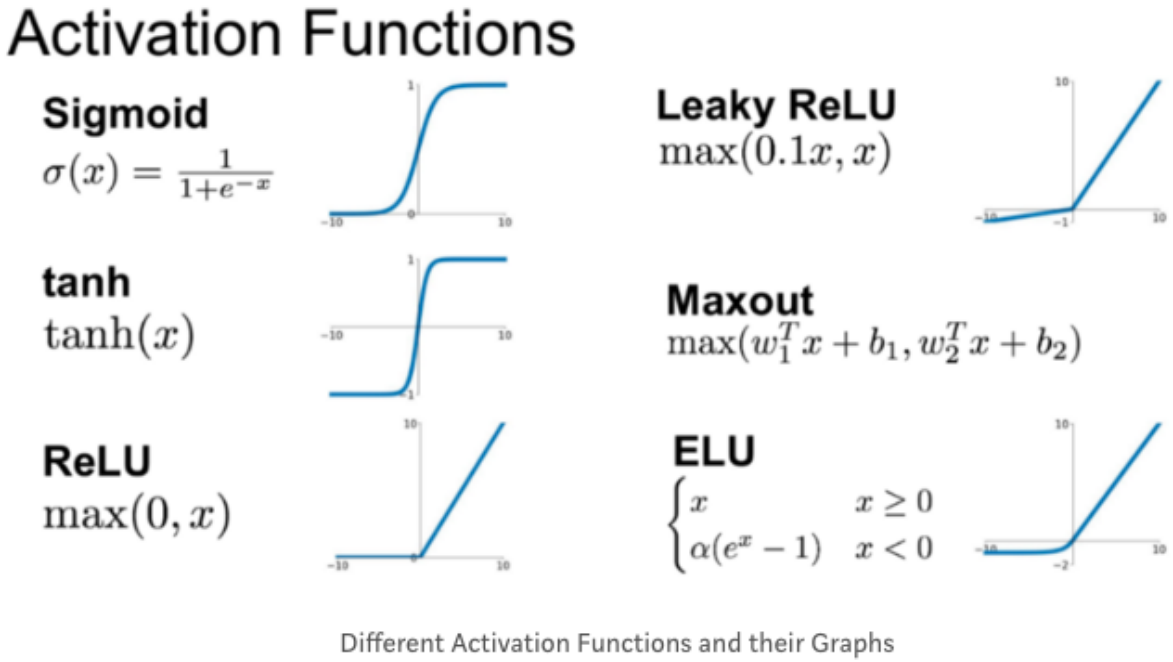
cost

accuracy



4. 분석결과

1) CNN



Different Activation Functions and their Graphs

Sigmoid 함수

Sigmoid 함수는 0<n<1 사이의 값만 다루므로 결국 chain rule을 이용해 계속 값을 곱해 나간다고 했을 때 결과 값이 0에 수렴할 수 밖에 없다는 한계를 가지고 있다. 이 한계를 극복하기 위해 Sigmoid 함수가 1보다 작아지지 않게 하는 위한 대안으로 여러 함수들이 있다.

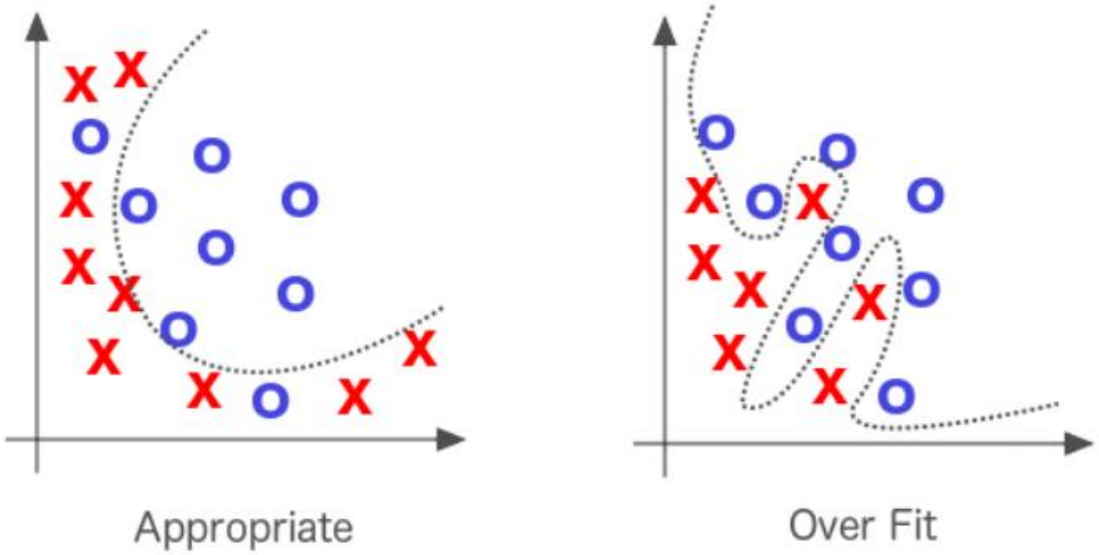
ReLU 함수

ReLU 함수는 0보다 작은 값이 나온 경우 0을 반환하고, 0보다 큰 값이 나온 경우 그 값을 그대로 반환하는 함수이다.

0보다 큰 값일 경우 1을 반환하는 sigmoid와 다르다.
따라서 내부 hidden layer에는 ReLU를 적용하고, 마지막 output layer에서만 sigmoid 함수를 적용하면 이전에 비해 정확도가 훨씬 올라가게 된다.

4. 분석결과

1) CNN

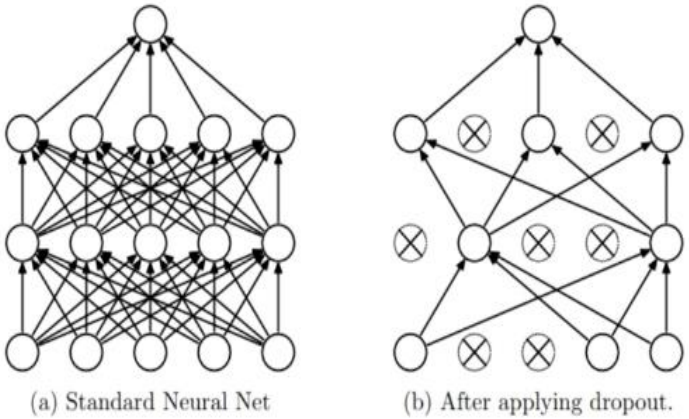


Overfitting

딥러닝의 가장 큰 문제점인 overfiting은 학습데이터를
과하게 잘 학습하는 것을 뜻한다.
달리 말해, 학습 데이터에 대해 과하게 학습하여 실제
데이터에 대한 오차가 증가하는 현상이라고 할 수 있다.

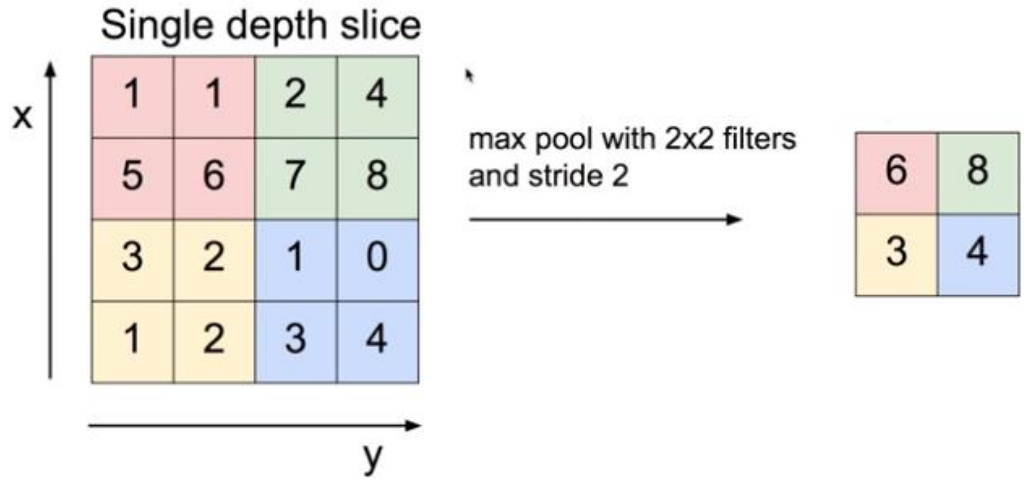
4. 분석결과

1) CNN



Dropout

Layer의 개수가 많을 때 overfitting을 발생시킬 수 있으므로
임의의 확률로 layer들을 학습에서 제외 시켜
overfitting을 방지하는 방법이다.



Max pooling

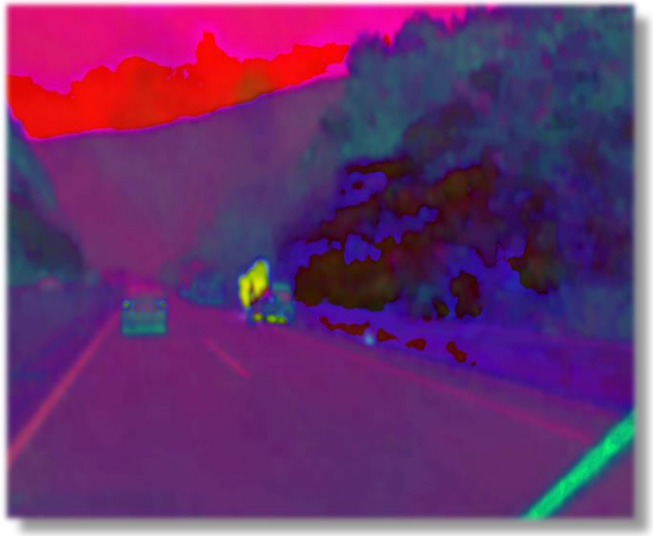
이미지의 pixel값이 크면 layer의 크기도 커져 overfitting을 발생시킬
수 있으므로 최대값을 대표값으로 설정해 layer의 크기를 줄여
overfitting을 방지하는 방법이다.

4. 분석결과

2) OpenCV

1) RGB

2) Gaussian Blur



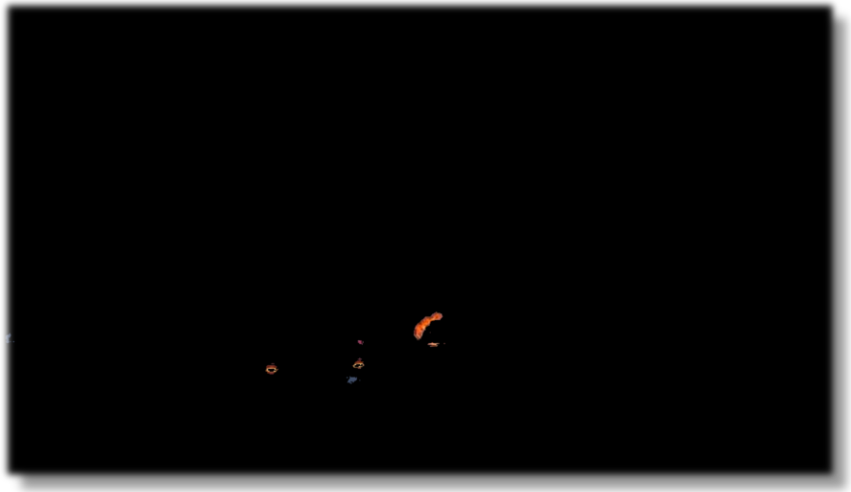
3) HSV

*단순 BGR2HSV 사용시

영상 HSV 변환

실제 영상에서의 화재(불)로 인식할 수 있는 HSV의 범위를
찾기 위해 영상을 HSV로 변환한다.

2) OpenCV



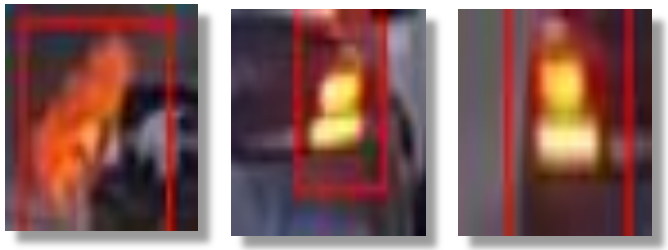
HSV값

실제 화재 영상들에 HSV 트랙바를 적용시켜 HSV 값을 추출했다.
몇 개의 영상에서 추출한 대략적인 HSV값의 범위를 정하였다.

본 분석에서 사용할 HSV값의 범위

	Lower	Upper
Hue	0	13
Saturation	50	190
Value	50	255

2) OpenCV

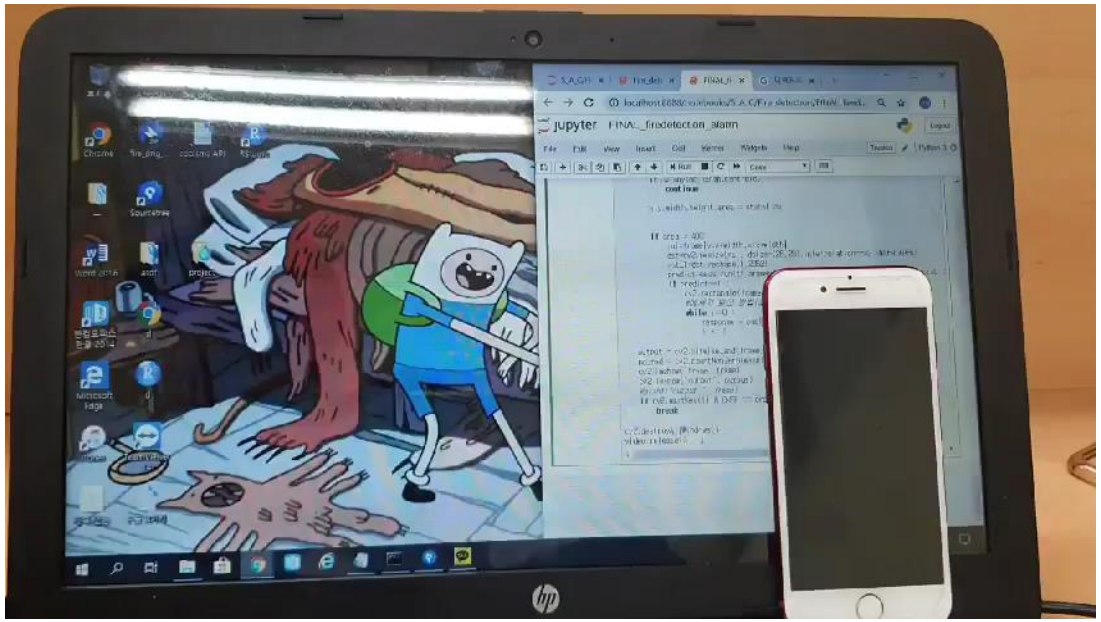


OpenCV가 인식한 불

OpenCV 화재 분류

OpenCV를 통해 1차적으로 분류한 영상의 캡쳐 화면이다.
화재 뿐만 아니라, 자동차의 라이트까지 불로 인식한 것을 알 수 있다.

3) CNN 적용 및 신고



최종 적용 영상

실제 영상에 완성된 알고리즘을 적용시킨 결과이다.
이전 OpenCV에서 인식했던 차량 조명을 인식하지 않고, 화재만을 인식하는 것을 알 수 있다.

완성된 화재 감지 및 신고 시스템이 실제 CCTV, 블랙박스, 스마트폰 등의 다양한 카메라 기기에 적용되어 신속한 화재 조치에 도움을 줄 수 있을 것이라 기대한다.

참고자료 분석도구

5. 참고자료 및 분석도구

분석 도구

python

참고 자료

*김영진,김은경『CNN을 활용한 영상 기반의 화재 감지』, JKIICE(2016.09)

*김영진,김은경『CNN과 Grad-CAM 기반의 실시간 화재 감지』, JKIICE(2018.12)

*뉴옌만동, 노승환 "딥 러닝을 이용한 화재 감지 알고리즘", (2017.07)

깃허브 : https://github.com/JinGiHyun/Statistics_Analysis_Project.git

KGU