캡스톤 디자인

최종 보고서

제출일 : 2021.06.23
과목명 : 캡스톤 디자인
학과명 : 컴퓨터시스템과
교수명 : 이세훈 교수님
이름 : 201745060 이승엽
201645084 노영훈

1. 개요

▪ 프로젝트 개요

현재 화재 감지 시스템은 주로 센서 기반으로 구성되어 있어 주변 환경에 따라 화재의 오감지의 가능성이 크고, 센서들을 일정한 간격으로 여러 대를 설치해야 되는 비용적인 문제점도 존재한다. 반면 영상을 이용한 시각적 화재 감지는 센서 기반에서 오는 문제점들을 극복할 수 있으며 기존 CCTV 장치를 이용해 큰 비용 없이 화재 감지를 적용할 수 있다.

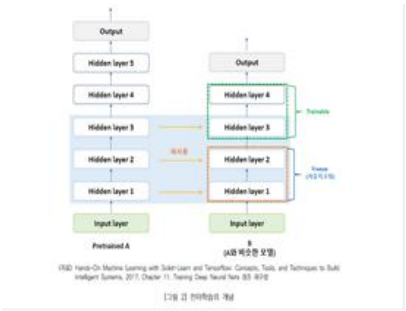
기존 영상을 통한 화염 검출 방법은 색채를 기반으로 검출하는 방법 등이 있는데 화재 발생 시 검출 속도가 빠르지만, 화염과 비슷한 색채의 물체가 있으면 오검출률이 높다는 단점이 있다.

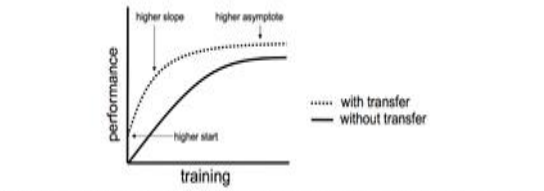
본 프로젝트에서는 Jetson Xavier 환경에서 YOLO를 통한 Object Detection을 수행하고, 돌발 상황에 대한 관심영역(ROI)를 추출하여 여러 종류의 CNN 알고리즘으로 학습한 모델과 비교하여 사고(정차, 화재)검출률의 상승과 사고상황의 실시간 인지와 단말기(Jetson Xavier)와 단말기(Arduino Nano) 사이에서 정보를 전달하는 에지 컴퓨팅을 이용하여 초동 대응이 가능한 시스템을 구축하고자 한다.

2. 관련 연구

▪ TransferLearning

인공지능 전이학습과 응용 분야 동향 - 정보통신기획평가원(2020. 02)





(자료) Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques, Chapter 11. Transfer Learning, p.243.

[그림 4] 전이학습을 활용할 때 장점

관련 연구고찰로는 인공지능 전이학습과 응용분야 동향이 있다. 정보통신기획평가원에서 2020년 2월 달에 전이학습과 응용 분야의 동향에 대해서 발표한 자료이다.

전이학습의 개념을 간단하게 살펴보면, 규모가 매우 큰 딥러닝 모델을 학습시킬 때 처음부터 새로 학습시키는 경우에 학습 속도가 느린 문제가 발생한다.

이러한 경우 기존 학습된 비슷한 모델이 있을 때 이 모델의 하위층(lower layer)을 가져와 재사용

하는 것이 학습 속도를 빠르게 할 수 있을 뿐만 아니라 학습에 필요한 훈련 셋(Training set)도 적다.

왼쪽 그림은 레이어의 재사용에 대한 자료이고 이미 학습된 레이어의 일부분과 가중치를 재사용해 모델 생성시간 단축과 학습시간 단축 및 정확도 향상의 효과를 가져올 수 있게 되는 전이학습의 장점을 설명한다.

오른쪽 그림은 전이학습 사용전과 사용후의 퍼포먼스 차이를 볼 수 있다.

전이학습 적용시 시작단계부터 적용하지 않은 모델과 비교해서 높은 정확도를 가지고 시작하고 보다 높은 정확도에서 모델의 학습이 끝나는 것을 관측할 수 있다.

▪ Dropout

터널 사고에 대한 데이터가 적어 Over fitting 현상에 대한 우려가 있었고, 적은 데이터에서 Over fitting을 억제하는데 좋은 Dropout이라는 방법을 적용하기위해 『Journal of Machine Learning Research』(2014)에 기재된 "Dropout A Simple Way to Prevent Neural Networks from Over fitting" 논문을 참고하였다.

p10. CNN 모델 테스트 에러율

Method	Error %
Binary Features (WDCH) (Netzer et al., 2011)	36.7
HOG (Netzer et al., 2011)	15.0
Stacked Sparse Autoencoders (Netzer et al., 2011)	10.3
KMeans (Netzer et al., 2011)	9.4
Multi-stage Conv Net with average pooling (Sermanet et al., 2012)	9.06
Multi-stage Conv Net + L2 pooling (Sermanet et al., 2012)	5.36
Multi-stage Conv Net + L4 pooling + padding (Sermanet et al., 2012)	4.90
Conv Net + max-pooling	3.95
Conv Net + max pooling + dropout in fully connected layers	3.02
Conv Net + stochastic pooling (Zeiler and Fergus, 2013)	2.80
Conv Net + max pooling + dropout in all layers	2.55
Conv Net + maxout (Goodfellow et al., 2013)	**2.47**
Human Performance	2.0

Table 3: Results on the Street View House Numbers data set.

논문에서 사용한 SVHN(Street View House Numbers data set) 데이터 기준으로

ConvNet + max-pooling은 3.95%의 에러율을 가진다.

ConvNet + max-pooling + dropout in fully connected layers은 3.02%의 에러율을 가진다.

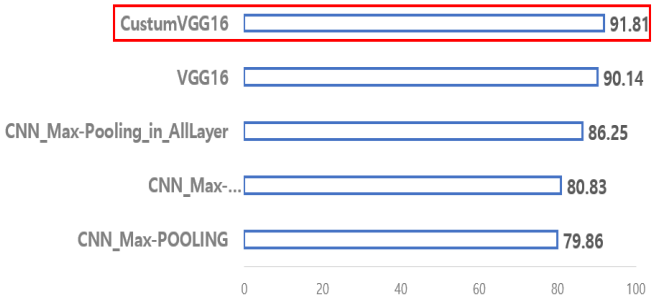
ConvNet + max-pooling + dropout in all layers는 2.55%의 에러율을 가진다.

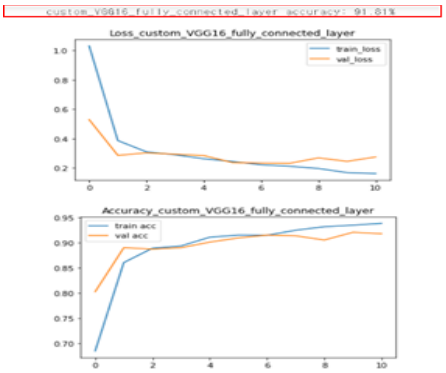
사용한 데이터 세트

Data Set	Domain	Training Set	Test Set
Tunnel_Accident	Vision	3000	750

해당 방법으로 모델을 만드는데 사용한 데이터 세트는 훈련 세트 3000개, 검증 세트 750개로 할당하였으며, 분류할 상황은 사고와 화재, 일반 상황에 대한 분류이다.

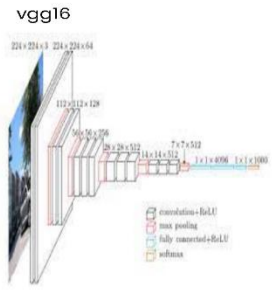
VGG16 + dropout in fully-connected layer

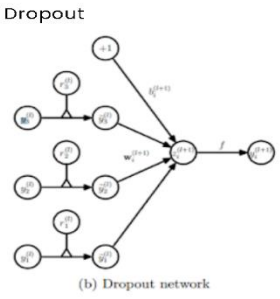


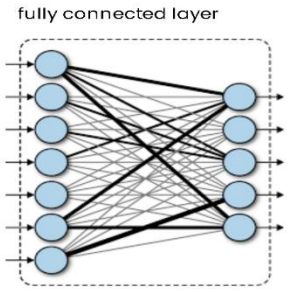


해당 방법들을 적용하여 만든 모델 중 검증 세트에서 가장 높은 정확도인 91.81%의 정확도를 보인 VGG16 + dropout in fully connected layer 방식을 사용한 CustomVGG16 모델로 선택하였다.

VGG16 모델은 기본적으로 ConvNet + max-pooling 구조로 이루어져 있고, 해당 모델은 ImageNet으로 사전 학습된 모델로서 터널 사고 데이터 세트로 전이학습을 수행하였다.







본 프로젝트의 이미지 검출모델의 간략한 구조이다.

▪ 화염 검출

참고한 연구로는 한국정보통신학회논문지에 기재된 "CNN을 활용한 영상 기반의 화재 감지 - 김영진, 김은경"을 참고하였다. 해당 논문에서는 YCbCr 컬러 모델을 사용하여 마스크를 만들어 화염 후보 영역을 추출하고 인접한 8개의 픽셀들을 통해 군집 형태에 따른 라벨링을 진행한다. 그 후 라벨링 된 ROI와 CNN모델을 비교하여 화염을 검출하는 방법을 제안하고 있다.

본 프로젝트에서는 위 논문의 화염 검출 프로세스를 참고하여 전체적인 시스템을 구상하였다

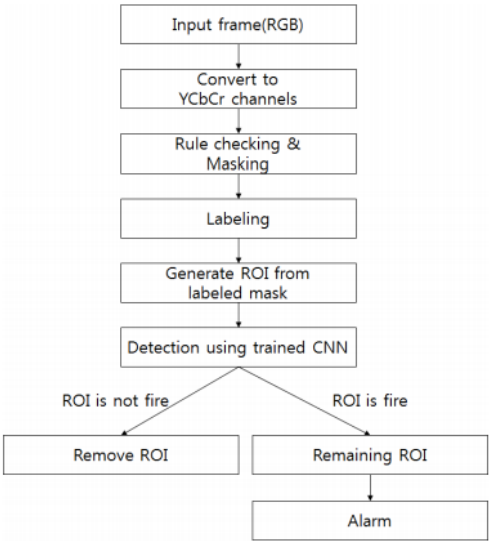


Fig. 7 CNN based Fire Detection Process

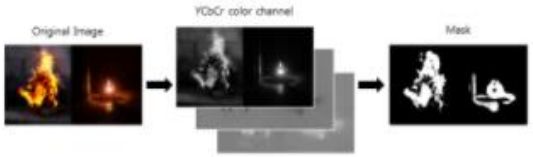


Fig. 8 Detection of Fire Candidate Region using YCbCr Color Model

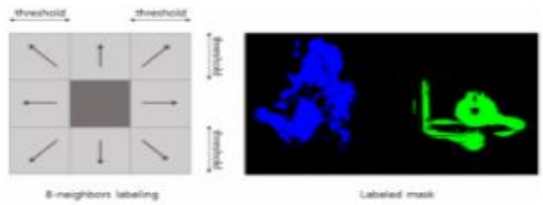
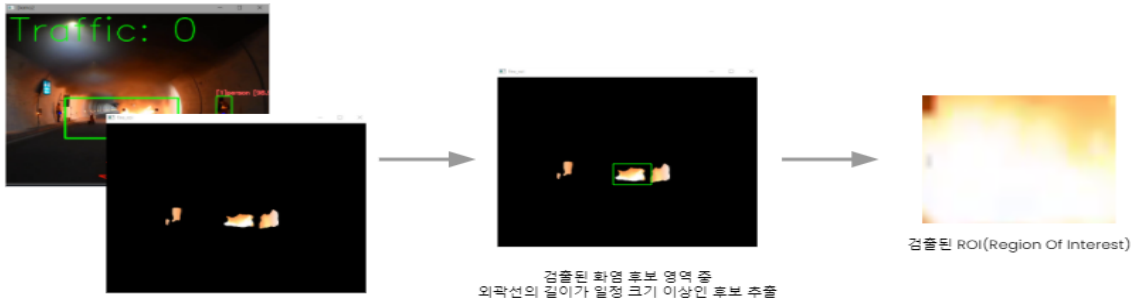


Fig. 9 8-neighbors labeling and the labeling result

8-neighbors를 이용한 픽셀의 군집 형태에 따른 라벨링

본 프로젝트에서는 위 논문에서 사용한 YCbCr 컬러 모델을 사용하여 ROI를 추출하는 대신 HSV 색 공간을 이용하여 마스크를 만들어 화염 후보 영역을 검출하고, findContours()함수를 사용하여 후보 영역들의 외곽선을 찾고 외곽선의 길이가 임계값(500)을 넘으면 경계사각형을 그려 ROI를 추출하였다.



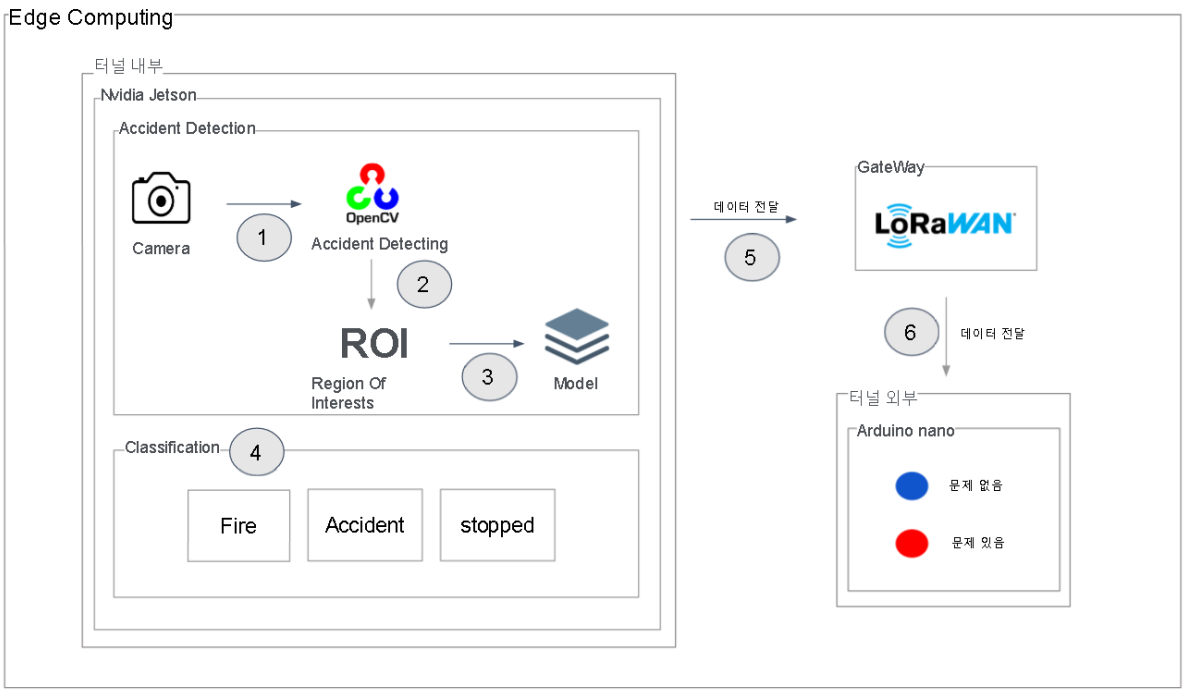
HSV를 통한 화염의 후보 영역 추출

검출된 화염 후보 영역 중
외곽선의 길이가 일정 크기 이상인 후보 추출

검출된 ROI(Region Of Interest)

3. 시스템 설계 및 구현

▪ 시스템 구조

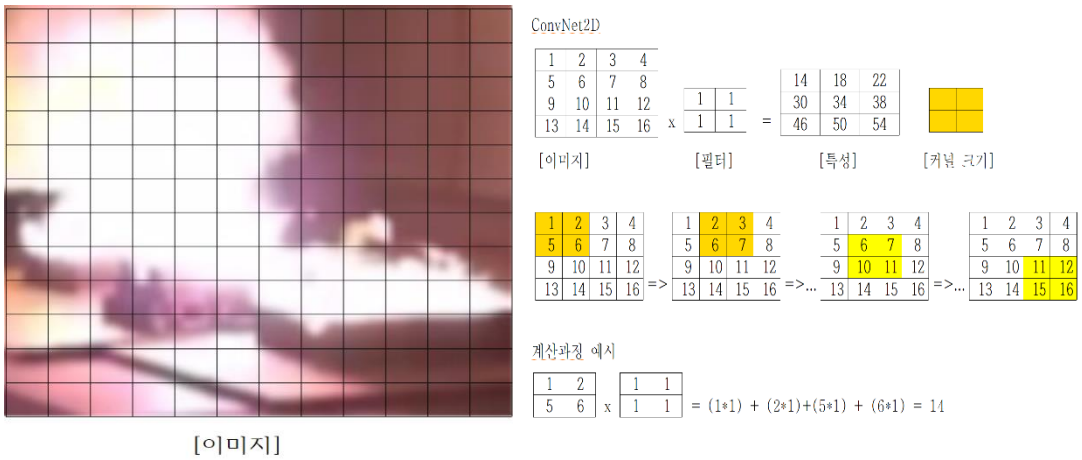


1. 실시간으로 CCTV의 카메라가 터널내 상황을 감지한다.
2. 화재 및 정차상황 발생 시 객체의 ROI를 검출한다.
3. 모델이 정차 상황 시 사고와 정차를 판단한다.
4. 모델이 화재 상황 시 화재와 비화재를 판단한다.
5. 사고가 인지되었을 경우 신호를 외부의 아두이노에 전달한다.
6. 아두이노는 받은 신호를 바탕으로 터널 외부의 운전자들에게 신호를 보낸다.

▪ Conv2D

- 2D Convolution

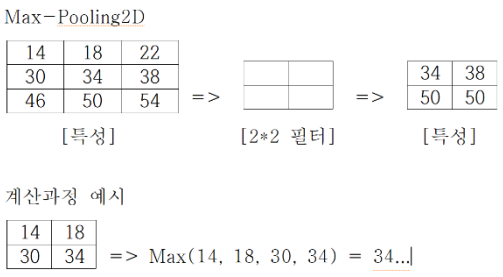
$$(f*g)(c_1,c_2)=\sum_{a_1,a_2} f(a_1,a_2)\cdot g(c_1-a_1,\ c_2-a_2)$$



<그림1> 이미지 합성곱 연산 Conv2D 예시

Conv2D는 이미지 좌상단에 이미지와 필터를 포개 놓고, 대응되는 숫자끼리 곱한 뒤, 모든 숫자를 더해주는 합성 곱연산을 수행함으로써 이미지의 특성 값을 추출해낸다.

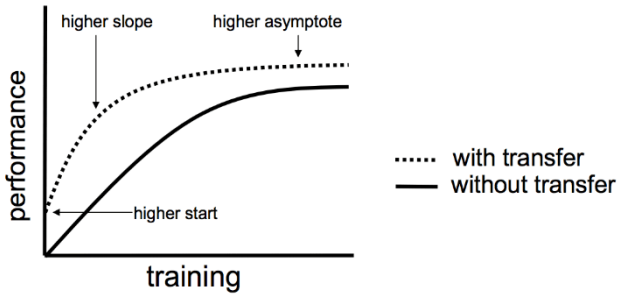
▪ Max-Pooling 2D



<그림2> Max-Pooling2D 예시

컨볼루션 결과로 나온 특성값들을 가지고 온뒤 좌상단의 특성값들에 필터를 포개놓고 그 중에 가장 큰 값을 특성값으로 뽑아낸다.

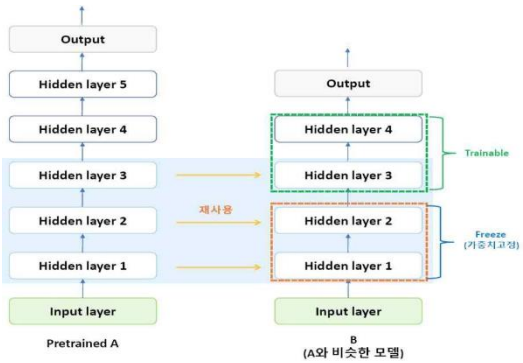
▪ Transfrer Learning



<그림3> 전이학습과 일반 학습의 성능 차이

전이학습은 모델을 생성하는 최적화 기법으로서 시간을 절약하거나 성능을 향상시키는 방법이다.

일반적인 학습에 비해서 더 높은 퍼포먼스와 경사값, 더 높은 점근선을 가진다.



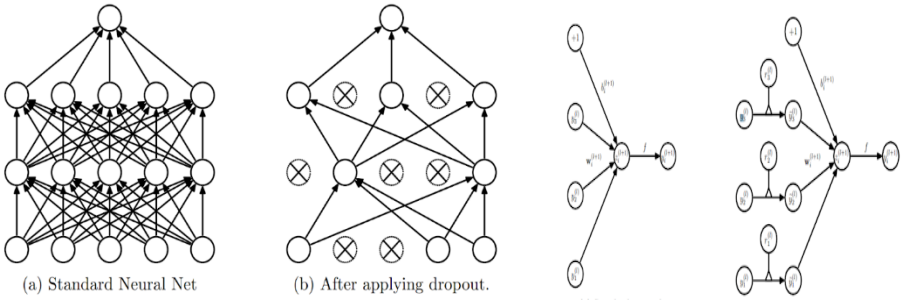
<그림4> 전이학습 예시

일반적으로 전이학습은 모델의 일부 레이어를 재사용해서 학습시간을 절약하고 높은 정확도를 가져올 수 있다.

<그림5> 사용한 VGG16 레이어

본 프로젝트에서는 사전 학습된 VGG16 모델의 일부 레이어를 가져와서 모델의 학습시간을 절약하고 높은 정확도를 가져올 수 있었다.

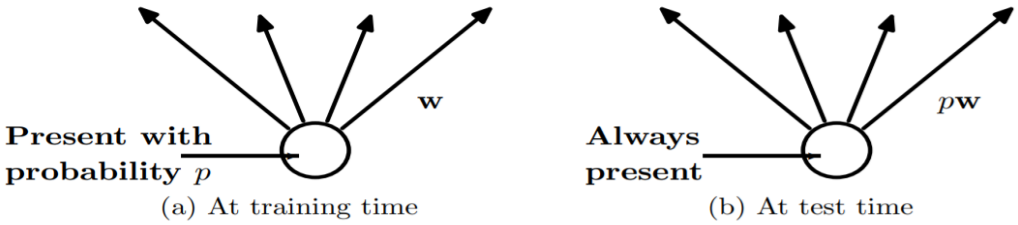
▪ Dropout



<그림6> Dropout Network

DNN의 고질적인 문제점인 Overfitting을 개선하기 위한 방법 중 하나이다.

레이어의 일부 유닛이 동작하지 않게 하여 Overfitting을 개선하는 네트워크의 형태이다.



<그림7> Dropout P Value

Dropout Rate를 p라는 변수라고 가정할 때 p가 1.0이면 모든 노드를 사용하는 상태이다.

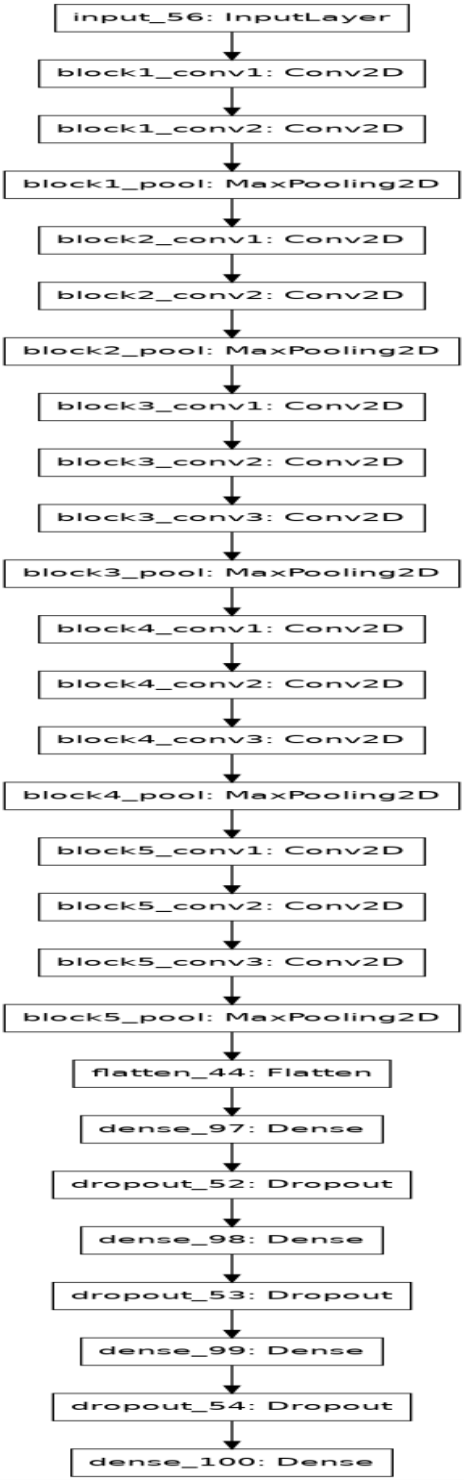
훈련 상황에서는 P Value가 0.6이라고 가정하면 노드의 40%를 무작위로 선택하고 해당 노드의 가중치를 0으로 설정해서 학습을 하지 않도록 만든다.

테스트 상황에서는 P Value값을 모두 가중치에 곱해서 적용한다.

그래서 테스트 시의 출력과 예상 출력은 동일하게 된다.

일반적으로 Dropout Network에서 P Value 값은 0.5~0.8의 값으로 설정하게 된다.

모델 구조 (CustomVGG16)

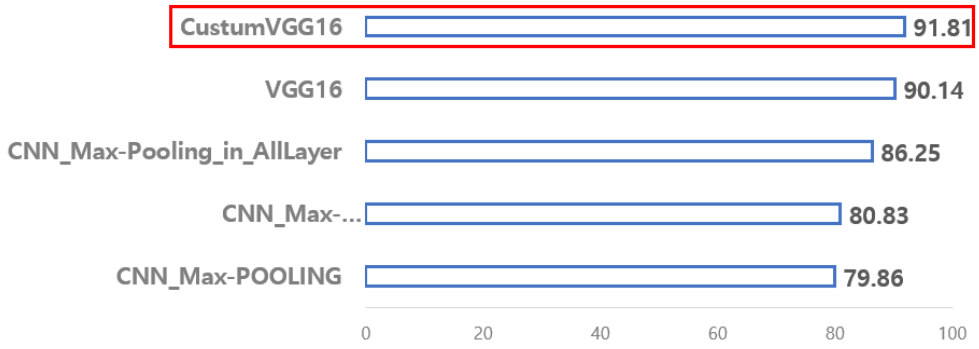


<그림8> 최종 모델 구조

▪ 모델

터널 사고 데이터를 바탕으로 다양한 CNN모델을 만들어 본 결과 VGG16 일부레이어에 드롭아웃레이어를 추가한 CustomVGG16 모델이 가장 검증 정확도가 높은 것을 확인 할 수 있었다.

VGG16 모델은 기본적으로 ConvNet + Max-Pooling 구조이며, ImageNet으로 학습된 모델을 터널 사고 데이터세트로 전이학습을 하였다.



<그림9>모델 검증 정확도 비교 표

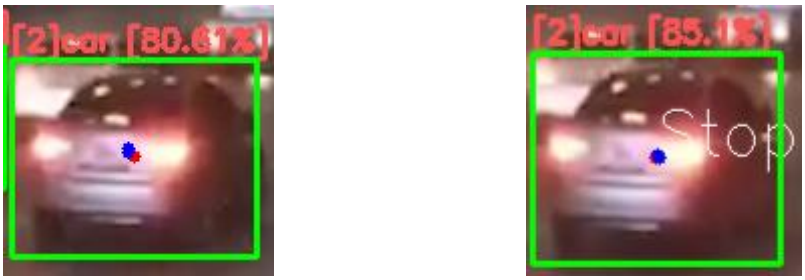
▪ 객체 탐지(Object Detection)

객체 탐지는 Custum dataset을 통해 YOLO_v4 Tiny를 학습시키고 객체 탐지(Object Detection)을 수행하여 사람, 차량, 오토바이, 자전거를 탐지 할 수 있다.

<그림 10>객체 탐지

▪ 정차 감지

정차 감지는 **<그림 11>**과 같이 객체의 파란점(5초 전의 중심점)과 빨간점(현재의 중심점)을 비교하여 빨간점 위치 오차 범위 ± 1 내에 파란점이 있다면 정차로 판단하고 ROI를 추출한다.



정차감지 전 **정차감지**

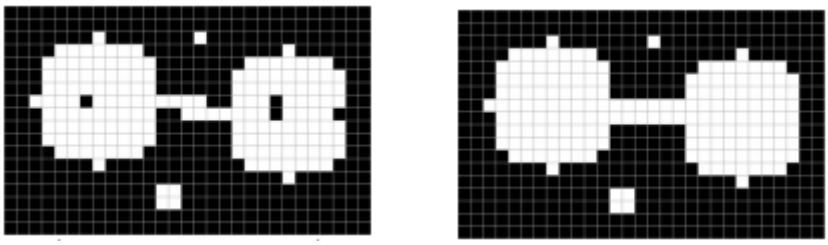
<그림 11>정차감지

▪화염 감지

화염 감지는 화염의 위치와 후보 영역을 검출하기 위해 영상으로부터 들어오는 원본 RGB 프,레임을 HSV 컬러 모델로 변환한다. 이후 화염의 색공간을 담는 최적의 HSV 하이퍼 파라미터 값을 찾아 넘파이 배열로 만든 뒤, HSV 컬러 모델로 변환한 프레임에서 **<그림 12>**과 같이 화염 후보 영역을 검출한다.

<그림 12>화염 검출

<그림12>에서 검출된 화염 후보 영역을 1로 이루어진 15x15 커널과 닫기(closing) 모폴로지(morphology)연산을 수행하여 노이즈를 제거하고, 연결선을 두껍게 만든다.



수행 전 **수행 후**

<그림 13>모폴로지 연산(Closing)

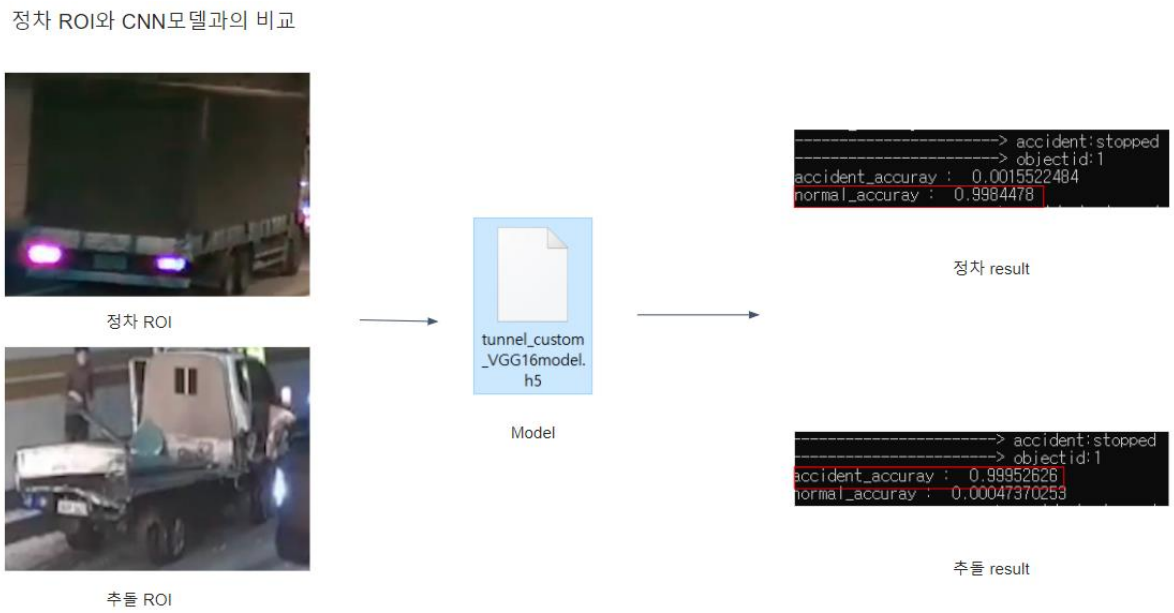
이후, threshold() 함수를 이용하여 0보다 크면 255, 작으면 0으로 만들어 영역을 뚜렷하게 만들고 findContours() 함수를 이용하여 외곽선을 그린다. 그려진 외곽선의 길이를 이용하여 작은 크기의 화염 후보 영역들을 제거하고 남겨진 후보 영역들에 바운딩 박스를 그려 ROI를 추출한다. 추출된 ROI는 CNN모델과 비교하여 fire와 Normal로 분류된다.

<그림 14>검출된 화염 ROI

▪추출된 ROI와 CNN모델 비교

본 프로젝트에서는 사고상황에 대해 OpenCV를 이용해 ROI를 추출하고 CNN모델에 넣어 분류하는 방식으로 정확도를 높인다.

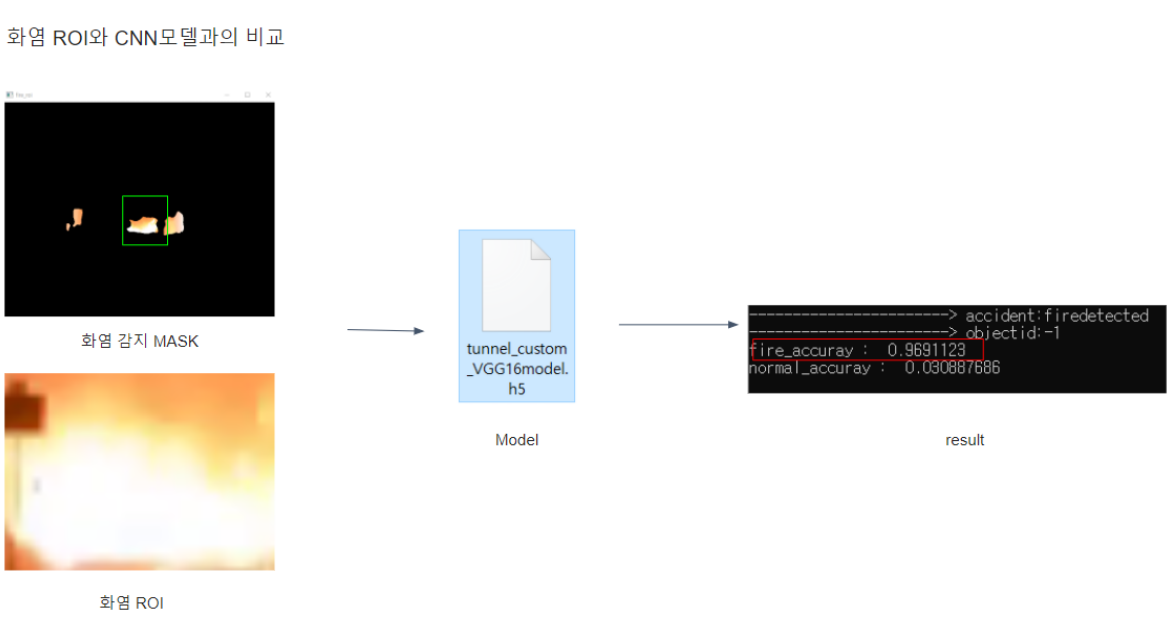
<그림 15>는 정차 ROI를 검증 정확도가 가장 높았던 Custom_VGG16 모델에 넣어 분류한 것이다. 정차 ROI의 경우 객체의 5초전 중심점과 현재 중심점 비교를 통해 정차를 감지하기 때문에 교통체증으로 인한 정차와 사고로 인한 정차가 모두 추출된다.



<그림 15> 정차ROI와 CNN모델

<그림 15>를 보면 교통체증으로 인한 정차는 Normal로 분류되는 것을 확인하였고, 사고로 인한 정차는 Accident로 분류되어 모델이 정확하게 분류하는 것을 확인할 수 있었다.

화염의 경우 HSV 색 기반 마스크를 통해 화염 후보 영역을 추출하고 추출한 후보 영역에서 외곽선의 길이를 통해 화염 ROI를 추출하였다.



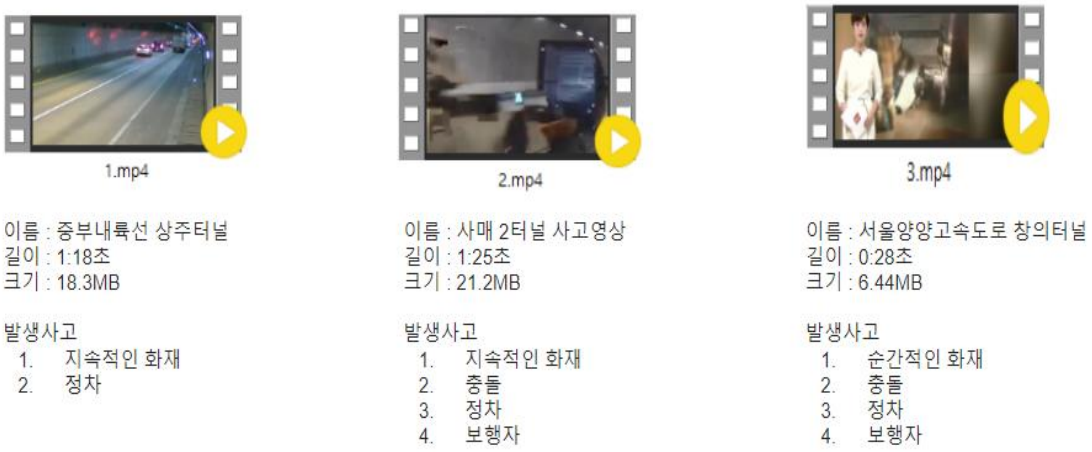
<그림 16> 화염ROI와 CNN모델

<그림 16>을 보면 화염ROI를 CNN 모델이 정확하게 분류하는 것을 볼 수 있다.

4. 실험 및 평가

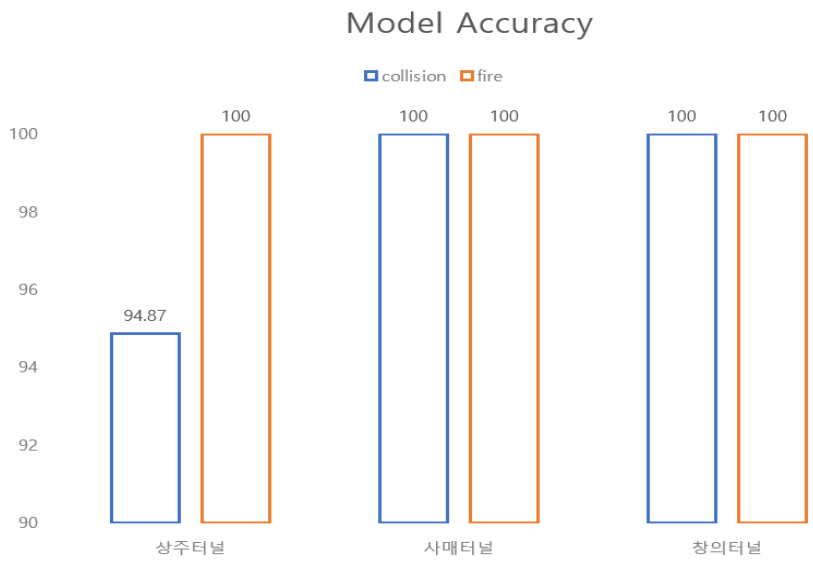
▪ 평가 및 테스트

본 프로젝트에서 제안한 CNN을 활용한 화재 감지 방법의 성능을 객관적으로 평가하기 위해, **<그림 17>**과 같은 3개의 동영상을 사용하여 평가를 진행하였다.



<그림 17> 사용 영상

<그림 17>에서 사용한 영상들은 터널 CCTV와 블랙박스 영상으로 다양한 사고가 발생한다. **<그림 18>**은 상주, 사매, 창의터널에서 검출된 ROI 이미지를 바탕으로 모델을 검증한 결과 값이다.



<그림 18> 모델 검증 정확도 차트

<그림 18>을 보면 사매터널, 창의터널 영상에서는 오검출율 없이 정확도가 100% 검출을 하였고, 상주터널의 영상에서는 충돌 모델 정확도가 94.87%로 다른 영상에 비해 정확도가 조금 낮지만 상주 터널을 제외한 영상에서는 우수한 결과를 보여주었다.

5. 결론

▪ 결론

본 프로젝트에서는 RGB 입력 영상 프레임을 HSV 색 기반 모델로 변환하여 후보 영역을 검출하고 검출된 후보 영역의 외곽선 길이를 통해 화염을 검출하고, 학습된 CNN 모델을 활용해서 화염 여부를 판단하고, 객체의 5초전 중심점과 현재의 중심점을 비교하여 정차를 감지하고 CNN 모델을 통해 사고 여부를 판단하는 시스템을 제안하였다.

VGG16 모델의 일부 레이어를 전이학습에 사용하고, Dropout을 적용함으로써 과적합을 줄이고 정확도가 높은 CNN 모델을 생성할 수 있었다. 사고 차량을 검출할 때, 정차한 차량 객체의 ROI를 검출하고 해당 ROI 이미지를 CNN 모델과 비교함으로써 단순 정차와 사고 상황을 정확하게 분류하는 결과를 얻어낼 수 있었다. 또한 기존의 컬러 모델을 이용한 화재 감지 방법이 유사 색상에 대해 오경보를 발생시키는데 반해, 본 연구에서는 CNN을 이용해 화염 후보 영역에 대해서만 최종적으로 화염 여부를 판 단함으로써 기존의 연구에 비해 오경보 비율을 크게 낮추는 결과를 얻었다. 특히 촛불, 램프, 라이터 등과 같이 화재로 오인될 수 있는 이미지들을 Normal 클래스로 정의하여 학습시킴으로써 불필요한 오경보를 발생시키지 않도록 하였다.

본 프로젝트는 에지 컴퓨팅을 이용하여 사고 정보가 서버를 거치지 않고 터널 내부 단말기와 외부 단말기의 LORA 통신을 통해 2차 피해를 방지하고 나아가 모델이 분류한 ROI 이미지를 웹에 올려 터널 사고 데이터셋을 구축하는 시스템을 구성하였지만, 현재는 사고상황에 대해서 판단할 수 있는 시스템만 구현된 상태이다. 향후 다양한 사고 클래스의 데이터를 확보하고 기술을 습득하여 연기 감지 및 다양한 사고 클래스들에 대해서 분류 및 모니터링이 가능한 시스템을 구현할 수 있게 할 것이다.

5. 참고문헌

[1] N. Srivastava, and G. Hinton, and A. Krizhevsky and I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," Journal of Machine Learning Research, Vol. 15, No.1 pp. 1930-1938, June 2014.

[2] https://www.itfind.or.kr/WZIN/jugidong/1934/file1812303722317570254-1934(2020.02.19)-06.pdf

[3] https://image-net.org/challenges/LSVRC/

[4] T. Celik and H. Demirel, "Fire Detection in Video Sequences Using a Generic color Model," Fire Safety Journal, vol. 44, no. 2, pp. 147-158, Feb. 2009

[5] Kim Youngjin, and Kim Eunkyung. "Image based Fire Detection using Convolutional Neural Network" 20.9 (2016): 1649-1656

[6] SIMONYAN, Karen; ZISSERMAN, Andrew. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, September 2014. - VGG

[7] WEISS, Karl; KHOSHGOFTAAR, Taghi M.; WANG, DingDing. A survey of transfer learning. Journal of Big data, Vol. 3, No.1 pp. 1-40, March 2016. -Transfer Learning

[8]
https://teddylee777.github.io/linux/docker%EB%A5%BC-%ED%99%9C%EC%9A%A9%ED%95%98%EC%97%AC-%EB%94%A5%EB%9F%AC%EB%8B%9D-%ED%99%98%EA%B2%BD%EA%B5%AC%EC%84%B1.md

[9]
https://www.youtube.com/watch?v=5pYh1rFnNZs&list=PL_Nji0JOuXg2E6QVMwCrLOzzTmr36fFcH&index=4

[10]
https://www.youtube.com/watch?v=5pYh1rFnNZs&list=PL_Nji0JOuXg2E6QVMwCrLOzzTmr36fFcH&index=5

6. 부록

▪ 모델 생성을 위한 서버 환경 구성

환경설정

https://teddylee777.github.io/linux/docker%EB%A5%BC-%ED%99%9C%EC%9A%A9%ED%95%98%EC%97%AC-%EB%94%A5%EB%9F%AC%EB%8B%9D-%ED%99%98%EA%B2%BD%EA%B5%AC%EC%84%B1.md

딥러닝 모델의 경우 학교에 있는 서버의 포트를 열어 tensorflow 2.4.1버전과 cuda 11.0버전, 주피터 노트북을 도커 이미지로 만들어 서버에 설치하였다.

▪ 소스 코드

ImageNet으로 훈련한 VGG16 모델을 전이학습시켜서 4가지 상황에 대한 분류 실시

(accident, dense_traffic, fire, sparse_traffic)

환경 : Ubuntu 18.04 , Jupyter notebook

사용한 파일 : https://github.com/OlafenwaMoses/Traffic-Net/releases/tag/1.0

라이브러리

```
## layer
from keras.layers import Input, Lambda, Dense, Flatten

from keras.models import Model

from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import preprocess_input
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.models import model_from_json
import numpy as np
from glob import glob
import matplotlib.pyplot as plt
```

 # 이미지 사이즈 지정

```
IMAGE_SIZE =[224,224]
train_path = 'trafficnet_dataset_v1/train'
valid_path = 'trafficnet_dataset_v1/test'
```

전이학습 모델 생성

```
# imagenet으로 훈련된 VGG16모델 생성.IMAGE_SIZE +[3]에서 [3]값은 RGB
input_shape = IMAGE_SIZE + [3] => (224, 224,3)

vgg = VGG16(input_shape=IMAGE_SIZE + [3], weights = 'imagenet',include_top=False)


# 기존 가중치를 훈련하지 않음.

for layer in vgg.layers:

layer.trainable = False
```

이미지 파일 경로 설정

```
# train폴더의 클래스들 사용

folders = glob('trafficnet_dataset_v1/train/*')


# 폴더확인

folders

# 설정된 폴더

[ 'trafficnet_dataset_v1/train/accident' ,

 'trafficnet_dataset_v1/train/sparse_traffic' ,

 'trafficnet_dataset_v1/train/dense_traffic' ,

 'trafficnet_dataset_v1/train/fire' ]


len(folders)
```

클래스 예측

```
# 각 folder 클래스 예측.ex) 사고, 원할한 차 통행, 원할하지 않은 통행, 화재

prediction = Dense(len(folders), activation='softmax')(x)
```

모델 구성 확인

```
# 모델 객체 생성

model = Model(inputs=vgg.input, outputs = prediction)

vgg16_model_obejct = model


# vgg16 모델 설명

model.summary()
```

모델 시각화

```
import keras.utils

keras.utils.plot_model(model)
```

모델 컴파일

```
model.compile(

loss = 'categorical_crossentropy',

optimizer = 'adam',

metrics=['accuracy']

)
```

데이터 전처리

```
from keras.preprocessing.image import ImageDataGenerator


train_datagen = ImageDataGenerator(rescale = 1./255,

shear_range = 0.2,

zoom_range = 0.2,

horizontal_flip = True)

# rgb 변형 (1~255) => (0~1)

test_datagen = ImageDataGenerator(rescale = 1./255)


# 훈련 세트 설정

training_set = train_datagen.flow_from_directory('trafficnet_dataset_v1/train',

target_size = (224,224),

batch_size = 32,

class_mode = 'categorical')


# 테스트세트 설정

test_set = test_datagen.flow_from_directory('trafficnet_dataset_v1/test',

target_size = (224,224),

batch_size = 32,

class_mode = 'categorical')
```

early stopping

```
from keras.callbacks import EarlyStopping, ModelCheckpoint


# early_stopping

epochs_to_wait_improve=10 # validation_loss가 10번 이내에 일정 수치 이상 향상이 없을 경우 학습을 종료

model_name = 'model/trafficfeatures_vgg16_model.h5' # model checkpoint(saved when validation_loss decrease)

early_stopping_callback = EarlyStopping(monitor = 'val_loss', patience=epochs_to_wait_improve)

checkpoint_callback = ModelCheckpoint(model_name, monitor='val_loss',verbose=1, save_best_only=True,mode='min')
```

학습

```
r = model.fit_generator(

training_set, # traing_set

validation_data = test_set,

epochs=50,

steps_per_epoch=len(training_set),

validation_steps=len(test_set),

callbacks=[early_stopping_callback, checkpoint_callback] # earlyStopping 상태에서 checkpoint에 저장됨.

)
```

시각화

```
# 손실율 시각화
```

```
plt.title('LossVal_loss_vgg16')
plt.plot(r.history['loss'],label = 'train_loss')
plt.plot(r.history['val_loss'],label = 'val_loss')
plt.legend()
plt.savefig('LossVal_loss_vgg16')
```

```
# 정확도 시각화
plt.title('AccVal_acc_vgg16')
plt.plot(r.history['accuracy'], label='train acc')
plt.plot(r.history['val_accuracy'], label='val acc')
plt.legend()
plt.savefig('AccVal_acc_vgg16')
```

모델 저장

```
자동으로 저장되지만 수동으로 저장하고 싶을때 사용가능한 방법
import tensorflow as tf
model.save('model/trafficfeatures_vgg16_model.h5')
========== 기준 선 ==========
```

 # 기준선 위의 코드는 학습과 저장을 위한 코드이고 아래 코드를 사용해서 기존에 학습한 모델을 불러와 예측하는데 사용 가능하다.

모델 불러오기

```
import tensorflow as tf
# model_load
model = tf.keras.models.load_model('model/trafficfeatures_vgg16_model.h5', compile=False)
```

예측 함수 생성

```
import tensorflow as tf
import cv2
from tensorflow.python.keras.preprocessing.image import img_to_array
from keras.applications.inception_v3 import preprocess_input
from IPython.display import Image # image input
# 예측 함수
def run_predict(file_path):
image = cv2.imread(file_path)
image = cv2.resize(image,dsize=(224,224))
image = img_to_array(image)
image = image.reshape((1, image.shape[0],image.shape[1],image.shape[2]))
image = preprocess_input(image)
yhat = model.predict(image)
print("accident_accuray : ",yhat[0][0])
print("dense_traffic_accuray : ",yhat[0][1])
print("fire_accuray : ",yhat[0][2])
print("sparse_traffic_accuray : ",yhat[0][3])
```

예측 시행

images/test1.JPG 이미지 예측

file_path = 'images/4.jpg'

run_predict(file_path)

Image("images/4.jpg")

```
In [20]: file_path = 'images/4.jpg'
         run_predict(file_path)
         Image("images/4.jpg")

         accident_accuray :  4.680762e-06
         dense_traffic_accuray :  2.2644215e-07
         fire_accuray :  0.99995923
         sparse_traffic_accuray :  3.587607e-05

Out[20]:
```

본인이미지 파일을 사용하고 싶으면 이러한 파일 구조로 생성하면 된다.

파일구조

```
| ㅡㅡtrafficnet
|  ㅡ
|  | --- train
|  ㅡ
|  |  ---class1
|  |  ---class2 ...
|  l--- test
|  ㅡ
|  | --- class1
|  |  ---class2 ...
```

▪ **YOLO v4 Tiny 사용을 위한 darknet 환경 구성**

YOLO v4 Tiny를 사용하기 위해 유튜브를 보고 darknet 환경을 구성하였다.

설치방법1 : https://www.youtube.com/watch?v=5pYh1rFnNZs&list=PL_Nji0JOuXg2E6QVMwCrLOzzTmr36fFcH&index=4

설치방법2 : https://www.youtube.com/watch?v=sUxAVpzZ8hU&list=PL_Nji0JOuXg2E6QVMwCrLOzzTmr36fFcH&index=5

▪ **소스 코드**

환경 : Window10, Python 3.7.7, OpenCV 4.1.0, AIexyab의 darknet

라이브러리

```
from ctypes import *
import math
import random
import os
import cv2
import numpy as np
import time
import threading
import darknet
from itertools import combinations
from time import sleep
import tensorflow as tf
from tensorflow.python.keras.preprocessing.image import img_to_array
```

사고감지

```
def cvDrawBoxes(detections, img):
if len(detections) > 0:
        centroid_dict = dict()
        centroid_dict2 = dict()
        roi_dict = dict()
        global centroid_list
        global before_centroid_list
        global sec
        global tm
        global model1
        car_roi=img.copy()

        objectId = 0

#사람감지
        for label, confidence, bbox in detections:
            name_tag =label
            if name_tag=='person':
                x, y, w, h = (bbox[0],
```

```
                    bbox[1],
                    bbox[2],
                    bbox[3])
                xmin, ymin, xmax, ymax =
convertBack(float(x), float(y), float(w), float(h))
                centroid_dict2[objectId] =
(int(x), int(y), xmin, ymin, xmax, ymax)
                objectId += 1
                pt1 = (xmin, ymin)
                pt2 = (xmax, ymax)

# 감지된 객체 판단 및 정확도 표시

cv2.putText(img,"["+str(objectId)+"]"+
                                label+
                                " [" +
str(confidence) + "%]",
                                (pt1[0], pt1[1]
- 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
                                (246,86,86), 2)


                for idx, box in
centroid_dict2.items():
                    cv2.rectangle(img,
(box[2], box[3]), (box[4], box[5]), (255, 0, 0), 2)
                    location = (150,55)


#차량 감지
            if name_tag == 'car' or
name_tag =='truck' or name_tag=='bicycle'
or name_tag=='motorbike':
                x, y, w, h = (bbox[0],
                    bbox[1],
                    bbox[2],
                    bbox[3])
                xmin, ymin, xmax, ymax =
convertBack(float(x), float(y), float(w), float(h))

# 5초마다 중심점 복사
                if sec%5==0:

before_centroid_list=centroid_list.copy()

# 경계 사각형에 중심점 추가.
                centroid_dict[objectId] =
(int(x), int(y), xmin, ymin, xmax, ymax)
                centroid_list[objectId] =
((int(x),int(y)))
                roi_dict[objectId]=(int(y),
int(y+h), int(x), int(x+w)) # y1 , y2 , x1, x2
(190, 394, 746, 1007)


                objectId += 1
                pt1 = (xmin, ymin)
                pt2 = (xmax, ymax)

# 감지된 객체 판단 및 정확도 표시

cv2.putText(img,"["+str(objectId)+"]"+
                                label+
                                " [" +
str(confidence) + "%]",
                                (pt1[0], pt1[1]
- 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
                                (246,86,86), 2)

# 중심점을 지속적으로 찍음
                cv2.line(img,
```

```
tuple(((int(x),int(y)))), tuple(((int(x),int(y)))),
(255, 0, 0), 5)


# 5초 전 중심점을 찍고, 임계값 사이에 있으면 스톱 텍스트 표시

                    for i in
range(len(before_centroid_list)-1):

                        cv2.line(img,
before_centroid_list[i], before_centroid_list[i],
(0, 0, 255), 5)

                        if tuple(((int(x-1),int(y-
1)))) <= before_centroid_list[i] and
tuple(((int(x+1),int(y+1)))) >=
before_centroid_list[i]:


cv2.putText(img,"Stop!",
before_centroid_list[i] ,
cv2.FONT_HERSHEY_SIMPLEX, 1,
(255,255,255), 1)


today=str(tm.tm_year)+"."+str(tm.tm_mon)+".
"+str(tm.tm_mday)+"-"+str(tm.tm_hour-3)+"
"+str(tm.tm_min)+" "+str(tm.tm_sec)


#차량 ROI

                        for box in
roi_dict.items():


car_roi2=car_roi.copy()


car_roi2=cv2.cvtColor(car_roi2,
cv2.COLOR_BGR2RGB)

                        # (box[2],
box[3]), (box[4], box[5]), (0, 255, 0), 2


car_roi2=car_roi2[int(y-h/2):int(y+h/2), int(x-
w/2):int(x+w/2)] # y1:y2 , x1:x2


cv2.imwrite("./accident/collision/collision%s.p
ng" % today, car_roi2)


file_path="./accident/collision/collision"+str(t
oday)+".png"


run_predict1(file_path,model1,"stopped",objec
tId,today)


red_zone_list = [] # 거리 조건에 맞지 않은 개체 ID를 포함하는 리스트

        red_line_list = []

        for (id1, p1), (id2, p2) in
combinations(centroid_dict.items(), 2):

            dx, dy = p1[0] - p2[0], p1[1] -
p2[1]

            distance = is_close(dx, dy)


            if distance>10 and distance
<100:

                if id1 not in red_zone_list:


red_zone_list.append(id1)


red_line_list.append(p1[0:2])

                if id2 not in red_zone_list:


red_zone_list.append(id2)


red_line_list.append(p2[0:2])


        for idx, box in centroid_dict.items():
# dict (1(key):red(value), 2 blue)   idx - key
box - value

            if idx in red_zone_list:   # id가
레드존 목록에 있는 경우
```

```
                cv2.rectangle(img, (box[2], box[3]), (box[4], box[5]), (255, 0, 0), 2) # 빨간색 경계사각형
            else:
                cv2.rectangle(img, (box[2], box[3]), (box[4], box[5]), (0, 255, 0), 2) # 초록색 경계사각형

# 3.3 위험분석 및 위험지표 표시 <> 가까움 정도 표시
        text = "Traffic: %s" % str(len(red_zone_list))
        location = (10,25)
        if len(red_zone_list)<3:
            cv2.putText(img, text, location, cv2.FONT_HERSHEY_SIMPLEX, 1, (0,255,0), 2, cv2.LINE_AA)
        else:
            cv2.putText(img, text, location, cv2.FONT_HERSHEY_SIMPLEX, 1, (255,0,0), 2, cv2.LINE_AA)

        for check in range(0, len(red_line_list)-1):
            start_point = red_line_list[check]
            end_point = red_line_list[check+1]
            check_line_x = abs(end_point[0] - start_point[0])
            check_line_y = abs(end_point[1] - start_point[1])
            if (check_line_x < 100) and (check_line_y < 100):
                cv2.line(img, start_point, end_point, (255, 0, 0), 2
cv2.putText(img,str(check_line_y), (start_point), cv2.FONT_HERSHEY_SIMPLEX, 0.3, (246,86,86), 1)
    return img

## 화재감지

_, frame = cap.read()
        frame = cv2.resize(frame, (800, 600))
        frame = cv2.GaussianBlur(frame, (3, 3), 1)
        hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

# 마스크 생성
        l_m = np.array([20, 10, 255])
        u_m = np.array([180, 30, 255])
        mask = cv2.inRange(hsv, l_m, u_m)

# 모폴로지 연산 수행
        kernel1 = np.ones((15, 15), np.uint8)
        mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel1)
        cv2.imshow("img2", mask)

        res = cv2.bitwise_and(frame, frame, mask=mask)
```

```
            cv2.imshow("aa",res)


            img = frame.copy()
            ret, thresh = cv2.threshold(mask, 0,
255, cv2.THRESH_BINARY)
            contours, hierarchy =
cv2.findContours(
                    thresh,
                    cv2.RETR_EXTERNAL,
                    cv2.CHAIN_APPROX_NONE
            )
            for cnt in contours:
                l = cv2.arcLength(cnt, True)
                if l > 500:
                    x, y, w, h =
cv2.boundingRect(cnt)


# cnn모델과 비교
                    img_test = frame[y:y + h,
x:x + w]
                    #img_test =
cv2.resize(img_test, (224, 224))


today=str(tm.tm_year)+"."+str(tm.tm_mon)+".
"+str(tm.tm_mday)+"-"+str(tm.tm_hour)+"
"+str(tm.tm_min)+" "+str(tm.tm_sec)
cv2.imwrite("./accident/fire/fire%s.png" %
today, img_test)


file_path="./accident/fire/fire"+str(today)+".p
ng"


run_predict2(file_path,model2,"firedetected",-
1,today)
                    label =
make_prediction(img_test, model2,
class_dictionary)
                    if label == 'fire':
                        img = cv2.rectangle(
                            img,
                            (x, y),
                            (x + w, y + h),
                            (0, 0, 255), 2)
                        cv2.putText(
                            img,
                            "Fire", (x, y),

cv2.FONT_HERSHEY_SIMPLEX,
                            0.7, (0, 0, 255), 2)
                        cv2.imshow("img33",
img)


            cv2.imshow("img", img)
```

```
        #a=cv2.resize(image,(1200,800))
        cv2.imshow('Demo2', image)
        cv2.waitKey(3)
```

모델비교

```
# 클래스 인덱스 별 라벨
class_dictionary = {}
class_dictionary[0] = 'not a fire'
class_dictionary[1] = 'not a fire'
class_dictionary[2] = 'fire'

#모델 화염 감지 함수
def make_prediction(image, model, class_dictionary):
    image = cv2.resize(image, (224, 224))
    img = image / 255.

    # convert to 4D tensor
    image = np.expand_dims(img, axis=0)

    # fire, not fire 구분
    class_predicted = model.predict(image)
    inID = np.argmax(class_predicted[0])
    label = class_dictionary[inID]
    return label

#모델 이미지 분류 함수
def run_predict1(file_path,model,fom,objectId,today):
#전처리
    image = cv2.imread(file_path)
    image = cv2.resize(image,dsize=(224,224))
    fimage=image
    img = image / 255.
    image = np.expand_dims(img, axis=0)

    yhat = model.predict(image)
    print("----------------------> "+"accident:"+fom)
    print("----------------------> objectid:"+str(objectId))
    print("accident_accuray : ",yhat[0][0])
    print("normal_accuray : ",yhat[0][1])

#분류된 항목별로 이미지 저장
    inID = np.argmax(yhat[0])
    if(inID==0):

file_path2="./accident_discrimination/collision/"+str(today)+" collision"+"-"+str(round(yhat[0][0],2))+".jpg"
        cv2.imwrite(file_path2, fimage)
    elif (inID==1):

file_path2="./accident_discrimination/normal/"+str(today)+" normal"+"-"+str(round(yhat[0][1],2))+".jpg"
        cv2.imwrite(file_path2, fimage)
```