이름 : 노영훈 날짜 : 2021.06.02(수)
학번 : 201645084
학과 : 컴퓨터시스템과
학년 : 3학년

주제(제목) : 에지 컴퓨팅(Edge Computing) 기반의 터널 내 사고방지 시스템

보고서 목차 : 개요, 관련 연구, 시스템구조, 개발 일정 및 참고 문헌, 별첨

□ 개요

1. 개발 목적 : 딥러닝과 에지 컴퓨팅을 활용한 시스템을 통해 터널 내부의 사고를 빠르게 파악하여 피해를 최소화 하고자 하는 것에 목적을 두는 프로젝트.

2. 개발의 필요성 : 폐쇄된 터널 내부에서 사고가 일어났을 때 외부에서는 터널 내 사고상황을 알 수가 없음. 작은 사고가 발생했더라도 후속 2차 사고 발생 가능성이 큼, CCTV가 터널에 있다곤 하지만 사람은 24시간 동안 터널을 확인할 수 없음.

3. 참여 인원 : 노영훈, 이승엽

□ 관련 연구

1. Dropout: A Simple Way to Prevent Neural Networks from Overfitting(논문)
요약 : DNN의 문제 중 overfitting을 다른 정규화 방법에 비해 크게 개선함.
해당 논문을 참고하게 된 계기는 사용할 데이터가 적기 때문에 Overfitting 현상에 대한 우려가 있었고, 적은 데이터에서 Overfitting을 억제하는데 좋은 Dropout이라는 방법을 알게 되어 사용하기 위해 해당 논문을 참고함.

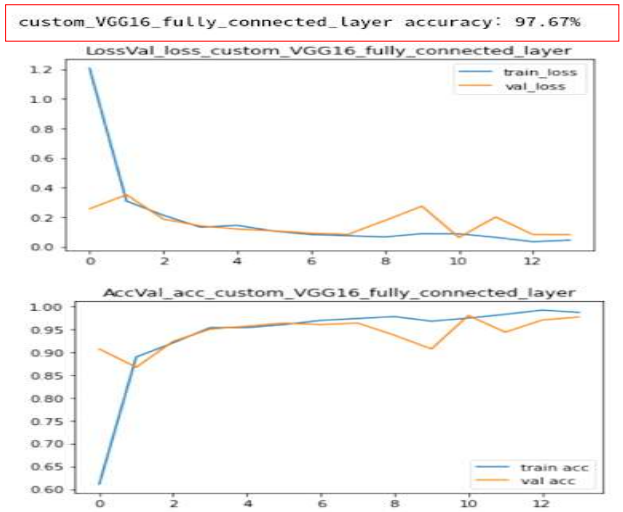
p10. CNN 모델 테스트 에러율

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 가지고, ConvNet + max-pooling + dropout in fully connected layers은 3.02%의 에러율을 가지고, ConvNet + max-pooling + dropout in all layers는 2.55%의 에러율을 가짐.
해당 논문에서 ConvNet + max-pooling, ConvNet + max-pooling + dropout in fully connected layers, ConvNet + max-pooling + dropout in all layers 방식을 참고, 해당 방법으로 모델을 만드는데 사용한 데이터 세트는 훈련 세트 1200개, 검증 세트 300개로 할당하였으며, 분류할 상황은 사고와 화재, 일반 상황에 대한 분류임.

VGG16 + dropout in fully_connected layer

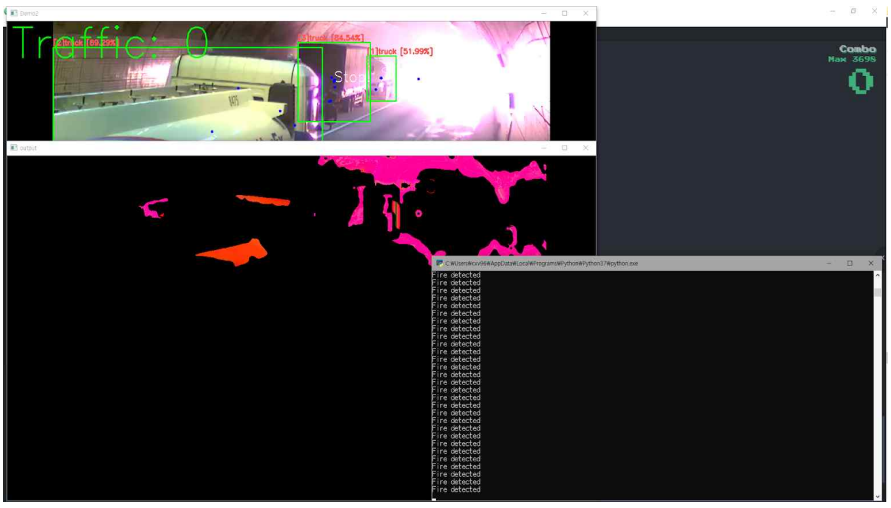


해당 방법들을 적용하여 만든 모델 중 검증 세트에서 가장 높은 정확도인 97.67%의 정확도를 보인
VGG16 + dropout in fully connected layer 방식을 사용할 모델로 선정.
VGG16 모델은 기본적으로 ConvNet + max-pooling 구조로 이루어져 있음, 해당 모델은 ImageNet으로 학습된 모델을 터널 사고 데이터 세트로 전이학습 하였음.

2. CNN을 활용한 영상 기반의 화재 감지(논문)

영상 프레임을 YCbCr 채널로 변환하여 화염 후보 픽셀을 만들고 라벨링을 수행한 CNN을 활용해서 영상 기반의 화재 감지 방법을 제안함.
기존의 영상 기반 화재 감지 방법과의 차별점은 CNN 구성으로 볼 수 있음. Caffe에서 제공하는 CNN 유형 가운데 하나인 BVLC의 구조를 일부 변경하여 화염 감지에 최적화된 CNN을 구성함.
- 원본 RGB 프레임을 YCbCr 컬러 모델로 변환
- 화염 후보 영역을 검출하는 식을 이용하여 화염 후보 영역을 검출하고 mask를 생성.
- 검출된 픽셀의 군집 형태에 따라 라벨링.
- 라벨링 된 마스크를 이용하여 ROI를 추출
- 추출된 부분 이미지는 미리 학습된 CNN을 통해 화염 여부를 확인.

결과는 기존 연구보다 오경보 비율을 크게 낮추지만, 화재가 아닌 연기와 화재의 연기를 구분하기 위해서는 많은 데이터가 필요하다는 결과가 도출됨.
화염 검출 조건을 만드는 데에 해당 논문을 참고하였음.
우선 HSV를 이용하여 화염 검출조건 설정하고 마스크를 만들어 원본 이미지와 합친 후 영역을 탐지하여 설정한 임계 값(영역)보다 크다면 화염으로 감지하고 메시지를 보내도록 설정.



□ 유사 시스템

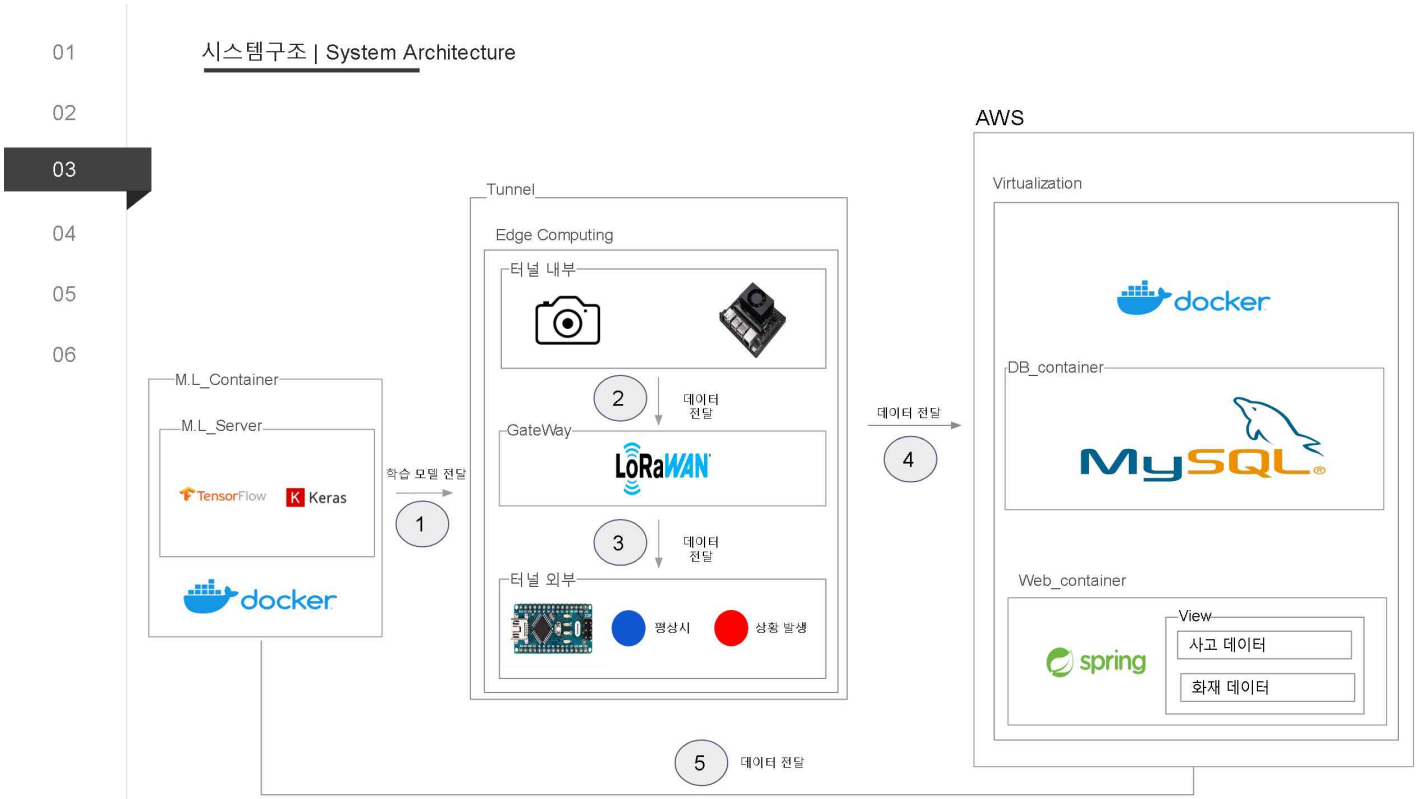
TADs(Tunnel Accident Detection System) – DBNtech

딥러닝 기반의 터널 유고감지 시스템으로 기능을 크게 세 가지로 나누어보면 유형별 사고감지, 영상추론 시스템, 관제 보조 시스템으로 나눌 수 있음.

사고감지 유형으로는 정차 감지, 역주행 감지, 화재 감지, 보행자 감지 등이 있으며, 영상추론 시스템은 딥러닝 기반 고정밀 실시간 유고감지 알고리즘을 사용하고 있으며, 열악한 환경에서도 높은 정확도의 사고 검출이 가능하다.

관제 보조시스템으로는 위치 정보 기반 관제 보조 소프트웨어를 지원하고 있으며, 마찬가지로 위치 정보 기반 CCTV 영상을 실시간으로 확인 가능, 유고 상황이 발생한 CCTV 영상 실시간 확인 가능

□ 시스템 구조



1. 서버 컴퓨터에서 학습모델을 JetsonNX에 전달.
2. 전달한 모델을 통해 JetsonNX가 들어간 CCTV가 터널 내부에서 사고상황 감지 후 결과 송신
3. 터널 외부의 아두이노가 송신 받은 정보를 통해 터널 외부에 터널 내부 사고를 알림.
4. 해당 사고 이미지를 DB에 저장
5. 다시 데이터를 서버에 전달하여 학습.

□ 개발 일정 및 참고 문헌

1. 개발 일정

내용	3월		4월				5월				6월	
	4주	5주	6주	7주	8주	9주	10주	11주	12주	13주	14주	15주
시장조사												
주제선정 및 자료조사												
시스템 설계												
데이터 수집 및 전처리												
기능 구현												
학습												
테스팅												
수정기간												

2. 참고 문헌

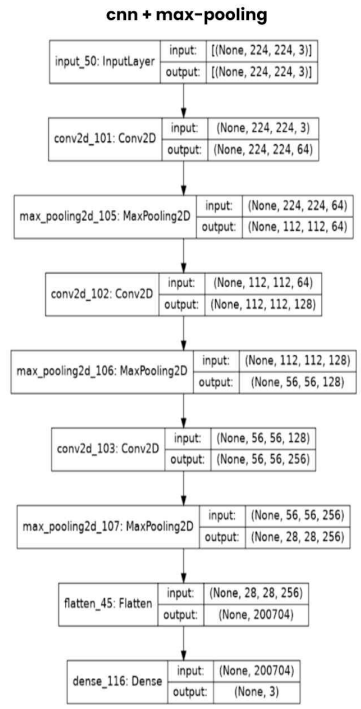
Dropout: A Simple Way to Prevent Neural Networks fromOverfitting
https://jmlr.org/papers/volume15/srivastava14a.old/srivastava14a.pdf

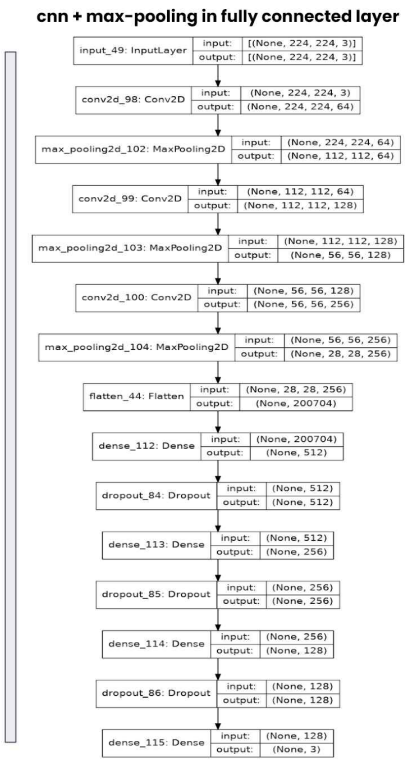
CNN을 활용한 영상 기반의 화재 감지
https://scienceon.kisti.re.kr/commons/util/originalView.do?cn=JAKO201630762630909&oCn=JAKO201630762630909&dbt=JAKO&journal=NJOU00431883

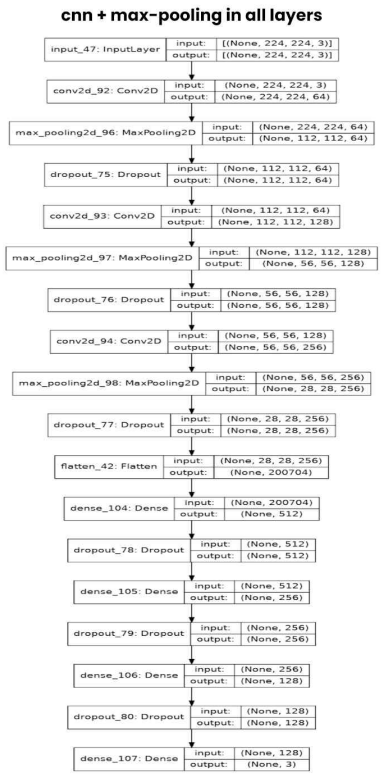
□ 별첨

01
02
03
04
05
06

별첨 - 모델 구조

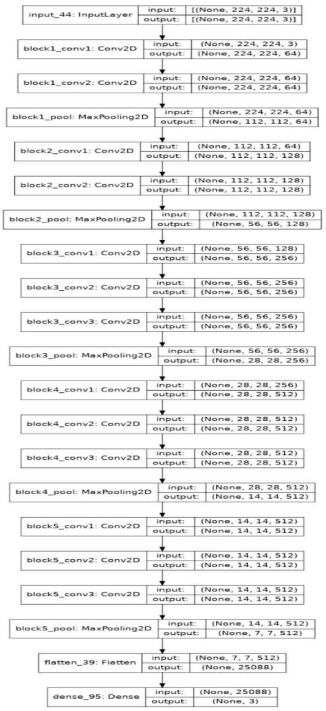






별첨 - 모델 구조

VGG16



VGG16 + in fully connected layer

VGG16 model .output (MaxPooling2D)	input :	(None, 14,14,512)
	output :	(None, 7,7,512)

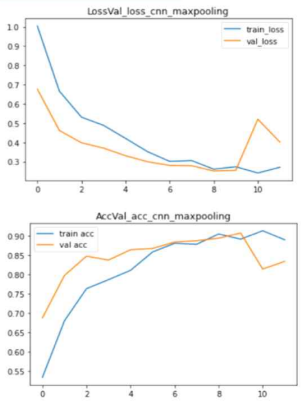
flatten_46: Flatten	input:	(None, 7, 7, 512)
	output:	(None, 25088)
dense_117: Dense	input:	(None, 25088)
	output:	(None, 512)
dropout_87: Dropout	input:	(None, 512)
	output:	(None, 512)
dense_118: Dense	input:	(None, 512)
	output:	(None, 256)
dropout_88: Dropout	input:	(None, 256)
	output:	(None, 256)
dense_119: Dense	input:	(None, 256)
	output:	(None, 128)
dropout_89: Dropout	input:	(None, 128)
	output:	(None, 128)
dense_120: Dense	input:	(None, 128)
	output:	(None, 3)

01
02
03
04
05
06

별첨 - 모델 정확도

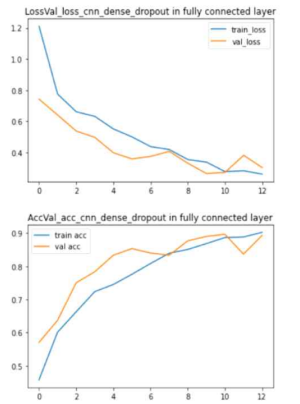
cnn + max-pooling

cnn_maxpooling accuracy: 83.33%



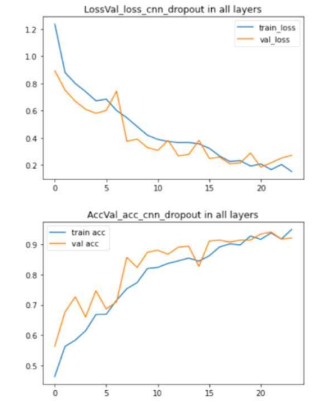
cnn + max-pooling + dropout in fully-connected layer

cnn_dense_dropout in fully connected layer accuracy: 89.33%



cnn + max-pooling + dropout in all layers

cnn_dropout in all layers accuracy: 92.00%



테스트 데이터로 검증 결과
cnn + max-pooling모델은 83.33%의 검증 정확도를

cnn + max-pooling + in fully-connected layer는
89.33%의 검증 정확도를

cnn + max-pooling + dropout in all layers는
92.00% 의 검증 정확도를 보였습니다.

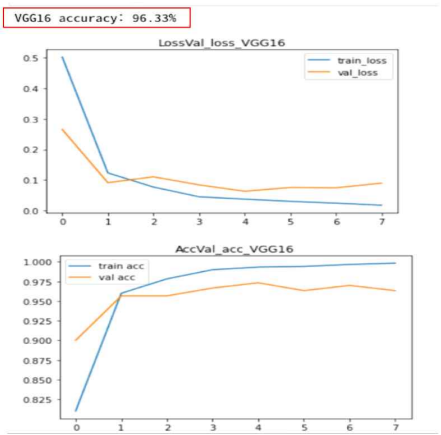
검증 코드

```
In [93]: def model_evaluate(model, model_name):
             scores = model.evaluate_generator(test_set,steps=len(test_set))
             print(model_name,"%s: %.2f%%" %(model.metrics_names[1], scores[1]*100))
```

별첨 - 모델 정확도

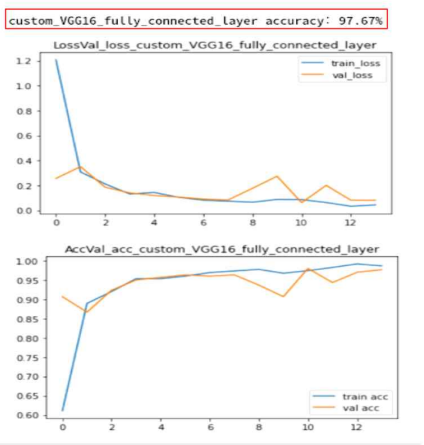
VGG16(Pretrained)

```
VGG16 accuracy: 96.33%
```



VGG16(Pretrained) + dropout in fully connected layer

```
custom_VGG16_fully_connected_layer accuracy: 97.67%
```



테스트 데이터로 검증 결과 VGG 16은 96.33%의 검증 정확도를

VGG 16 + in fully-connected layer는 97.67% 의 검증 정확도를 보였습니다.

검증 코드

```
In [93]: def model_evaluate(model, model_name):
             scores = model.evaluate_generator(test_set,steps=len(test_set))
             print(model_name,"%s: %.2f%%" %(model.metrics_names[1], scores[1]*100))
```