2021 년 1 학기 인공지능특론 과제 제안서

영상을 통한 인공지능의 교통사고 위치 파악 및 사고 영상 검출

주제 설명

1. 문제상황

보유한 차량의 대수에 비례하여 매일 다수의 사고가 발생하고 있으며, 사고 영상을 찾기 위해 많은 노력이 필요하다. 영상을 수집 후 영상내 사고의 발생 지점을 찾아야 하는 노력이 필요하기 때문이다.

2. 영상을 수집하는 과정

a. 고객이 신고한 경우 고객이 신고한 사고 시각을 기준으로 인근 CCTV 일정 구간의 영상을 수집한다.

b. 고객이 신고하지 않은 경우 어느 시점의 영상을 수집해야 하는지 모르기 때문에 의심되는 구간 전체를 수집한다. 또한 네트워크를 통해 업로드하기 어려운 경우 SD 카드를 회수하여 영상 확보한다.

c. 블랙박스의 자료들로 사고 발생을 진단한다.

3. 해결 과제

영상을 확보시 다음과 같이 영상내 사고시점을 검출할 수 있다.

a. 사고 발생시 움직이는 블랙박스 좌표를 이용해서 많은 영상중에 사고의 영상을 찾아낼수 있다.

b. 사고 발생시 목격자의 신고가 없을 경우 사고 발생지점의 인근 CCTV 혹은 사고 주위의 차량의 블랙박스를 통해 사고 영상을 도출할 수 있다.

4. 해결 전략

주어진 데이터를 이용해 영상내 사고가 있을 확률을 도출하는 프로그램 개발

a. 주행중인 차량내 블랙박스의 영상내 사고 시점 도출

1) 차량과 차량이 충돌시 블랙박스가 크게 흔들린다. 블랙박스의 진동폭이 큰 데이터를 통하여 사고 시점을 검출한다.

2) 영상내 사고의 좌표를 이용해서 시간에 따른 좌표 변화의 BPTT 시계열 자료를 이용해서 영상내 사고가 발생했을 확률을 판단한다.

관련된 학습 연구 및 방법

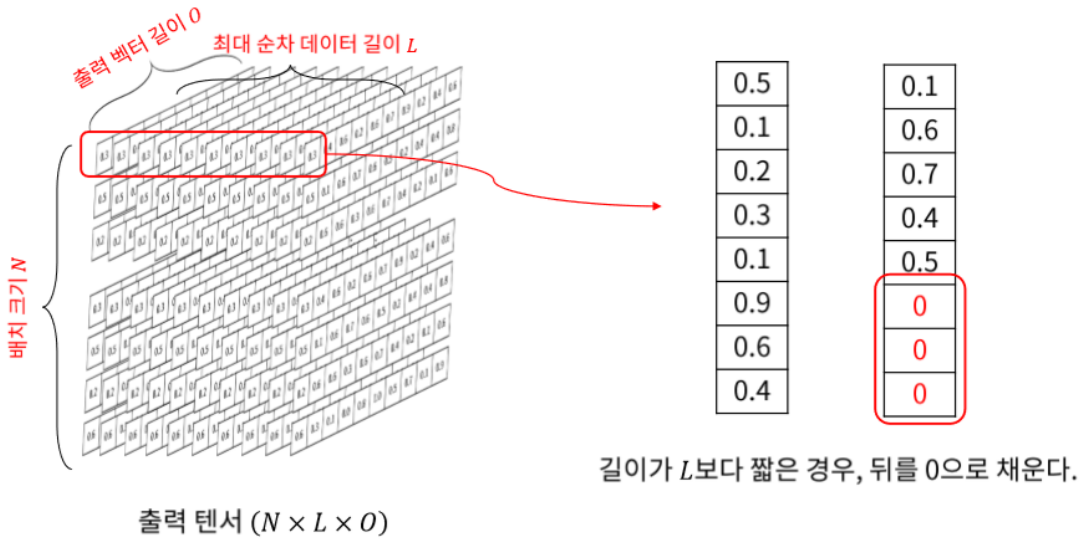
1. 연구 방법의 설명

a. 블랙 주어진 좌표를 읽고 시간에 따라 변화된 움직임을 저장한다.

b. 저장된 좌표들의 값들에 +10 을 하여 평균중심점을 (10,10,10)으로 만든다.

(1) LSTM 시계열 레이어의 embed 분류를 하기 위해 음수값을 양수값으로 바꾼다.

RNN의 출력 텐서



(2)

c. 사고의 발생한 영상파일의 이름들을 저장하는 리스트를 만들고 좌표와 함께 라벨한 파일을 x_train_acc, y_train_acc 으로 저장한다.

d. 사고가 나지 않은 파일의 영상들의 이름을 저장하는 리스트를 만들고 그 좌표변화들의 데이터를 x_train_non_acc, y_train_non_acc 으로 저장한다.

e. Test 셋과 training 셋으로 20: 80 으로 나눈후 인공지능 모델에 넣는다.

f. 모델의 구조는 다음과 같다.

1) self.emb = tf.keras.layers.Embedding
   (i) 임베디드 레이어를 사용하여 0~10000 까지의 변화구조를 선정한다.
2) self.rnn = tf.keras.layers.LSTM(32)
   (i) LSTM 레이어를 사용하여 시계열 데이터를 넣는다.
3) self.dense = tf.keras.layers.Dense(3, activation='softmax')
   (i) 소프트맥스 활성함수를 사용하고 fully connected layer 를 지나가게 한다.

g. Optimizer 는 adamoptimizer 를 사용하며,
loss 함수는 `CategoricalCrossentropy()`를 사용한다.

1) loss_object = tf.keras.losses.CategoricalCrossentropy()
2) optimizer = tf.keras.optimizers.Adam()

(2)

h. Iterator 를 1000 으로 설정하여 모델 학습을 시작하며 모델의 성능을 출력한다.

```
poch 28, Loss: 0.6161779761314392, Accuracy: 69.31103515625, Test Loss: 0.6180636286735535,
poch 29, Loss: 0.6162238121032715, Accuracy: 69.31103515625, Test Loss: 0.6178680529594421,
poch 30, Loss: 0.6162056922912598, Accuracy: 69.31103515625, Test Loss: 0.6178016066551208,
poch 31, Loss: 0.6160716414451599, Accuracy: 69.31103515625, Test Loss: 0.6177147030830383,
poch 32, Loss: 0.6160773038864136, Accuracy: 69.33200073242188, Test Loss: 0.6177210211753845
poch 33, Loss: 0.6157849431037903, Accuracy: 69.33200073242188, Test Loss: 0.6179473400115967
poch 34, Loss: 0.6159039735794067, Accuracy: 69.33200073242188, Test Loss: 0.6177881956100464
poch 35, Loss: 0.6159586906433105, Accuracy: 69.33200073242188, Test Loss: 0.6178967356681824
poch 36, Loss: 0.6158143877983093, Accuracy: 69.34249114990234, Test Loss: 0.6177053451538086
```

i.

2. 예상되는 적용

a. 사고 영상을 찾는 인적 자원을 절약 할 수 있으며, 사고 시작 지점을 찾는 시간을 줄일 수 있다.

b. 사고 책임 판단에 적용하여 영상을 통해 사고 책임을 빠르게 판단 할 수 있다.

c. 새로운 영상 및 블랙박스 자료들을 인공지능 모델에 넣었을 시 영상내 사고가 발생한 시점이 있을 확률을 알 수 있다.

3. 관련 연구들이나 논문들 자세한 요약

You Only Look Once: Unified, Real-Time Object Detection Joseph Redmon∗ , Santosh Divvala∗†, Ross Girshick¶ , Ali Farhadi∗† University of Washington∗ , Allen Institute for AI† , Facebook AI Research¶

CVPR. 2016 발표된 논문

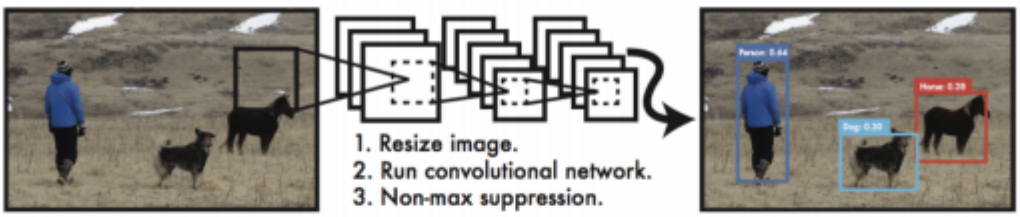


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

a.

VOC 2012 test	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
MR_CNN_MORE_DATA [11]	**73.9**	**85.5**	**82.9**	**76.6**	**57.8**	**62.7**	**79.4**	77.2	86.6	**55.0**	**79.1**	**62.2**	87.0	**83.4**	**84.7**	78.9	45.3	73.4	65.8	80.3	74.0
HyperNet_VGG	71.4	84.2	78.5	73.6	55.6	53.7	78.7	**79.8**	87.7	49.6	74.9	52.1	86.0	81.7	83.3	**81.8**	**48.6**	**73.5**	59.4	79.9	65.7
HyperNet_SP	71.3	84.1	78.3	73.3	55.5	53.6	78.6	79.6	87.5	49.5	74.9	52.1	85.6	81.6	83.2	81.6	48.4	73.2	59.3	79.7	65.6
Fast R-CNN + YOLO	70.7	83.4	78.5	73.5	55.8	43.4	79.1	73.1	**89.4**	49.4	75.5	57.0	**87.5**	80.9	81.0	74.7	41.8	71.5	68.5	**82.1**	67.2
MR_CNN_S_CNN [11]	70.7	85.0	79.6	71.5	55.3	57.7	76.0	73.9	84.6	50.5	74.3	61.7	85.5	79.9	81.7	76.4	41.0	69.0	61.2	77.7	72.1
Faster R-CNN [27]	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
DEEP_ENS_COCO	70.1	84.0	79.4	71.6	51.9	51.1	74.1	72.1	88.6	48.3	73.4	57.8	86.1	80.0	80.7	70.4	46.6	69.6	**68.8**	75.9	71.4
NoC [28]	68.8	82.8	79.0	71.6	52.3	53.7	74.1	69.0	84.9	46.9	74.3	53.1	85.0	81.3	79.5	72.2	38.9	72.4	59.5	76.7	68.1
Fast R-CNN [14]	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	**87.5**	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
UMICH_FGS_STRUCT	66.4	82.9	76.1	64.1	44.6	49.4	70.3	71.2	84.6	42.7	68.6	55.8	82.7	77.1	79.9	68.7	41.4	69.0	60.0	72.0	66.2
NUS_NIN_C2000 [7]	63.8	80.2	73.8	61.9	43.7	43.0	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	38.6	68.3	58.0	68.7	63.3
BabyLearning [7]	63.2	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6
NUS_NIN	62.4	77.9	73.1	62.6	39.5	43.3	69.1	66.4	78.9	39.1	68.1	50.0	77.2	71.3	76.1	64.7	38.4	66.9	56.2	66.9	62.7
R-CNN VGG BB [13]	62.4	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3
R-CNN VGG [13]	59.2	76.8	70.9	56.6	37.5	36.9	62.9	63.6	81.1	35.7	64.3	43.9	80.4	71.6	74.0	60.0	30.8	63.4	52.0	63.5	58.7
YOLO	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
Feature Edit [32]	56.3	74.6	69.1	54.4	39.1	33.1	65.2	62.7	69.7	30.8	56.0	44.6	70.0	64.4	71.1	60.2	33.3	61.3	46.4	61.7	57.8
R-CNN BB [13]	53.3	71.8	65.8	52.0	34.1	32.6	59.6	60.0	69.8	27.6	52.0	41.7	69.6	61.3	68.3	57.8	29.6	57.8	40.9	59.3	54.1
SDS [16]	50.7	69.7	58.4	48.5	28.3	28.8	61.3	57.5	70.8	24.1	50.7	35.9	64.9	59.1	65.8	57.1	26.0	58.8	38.6	58.9	50.7
R-CNN [13]	49.6	68.1	63.8	46.1	29.4	27.9	56.6	57.0	65.9	26.5	48.7	39.5	66.2	57.3	65.4	53.2	26.2	54.5	38.1	50.6	51.6

Yolo 모델의 다른모델과 비교해보았을때의 평균 정확도

b. Fast RNN 보다 정확성은 떨어지지만 FPS 가 높을때 Real time 으로 사물을 인지할때 성능이 뛰어나다.

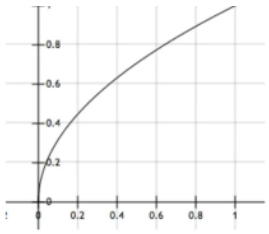
Real-Time Detectors	Train	mAP	FPS
100Hz DPM [30]	2007	16.0	100
30Hz DPM [30]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [37]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[27]	2007+2012	73.2	7
Faster R-CNN ZF [27]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

Table 1: Real-Time Systems on PASCAL VOC 2007. Comparing the performance and speed of fast detectors. Fast YOLO is the fastest detector on record for PASCAL VOC detection and is still twice as accurate as any other real-time detector. YOLO is 10 mAP more accurate than the fast version while still well above real-time in speed.

c. Yolo 의 loss function

loss function: Grid cell 보다, BB 를 더 크게

위치보다, 크기에 대해서 큰것보다 작은박스 에러가 작을수 있으니 큰물체 작은에러



$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

No obj

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2 \quad (3)$$

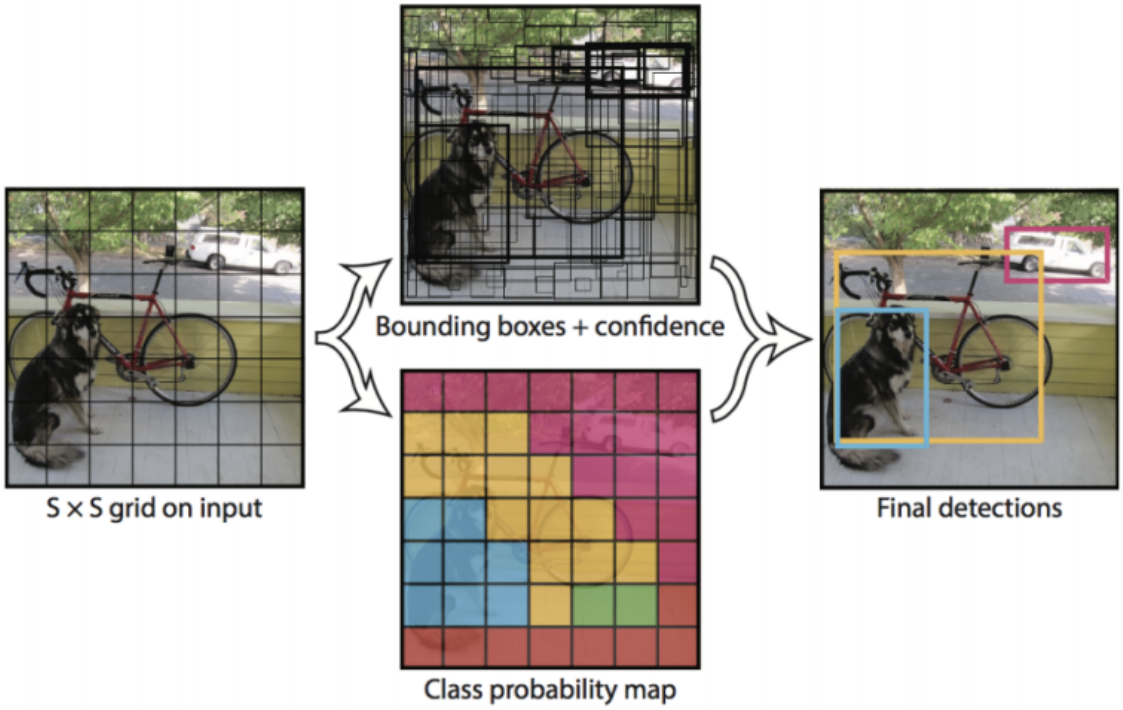
$\lambda_{\text{coord}}=5$, $\lambda_{\text{noobj}}=0.5$

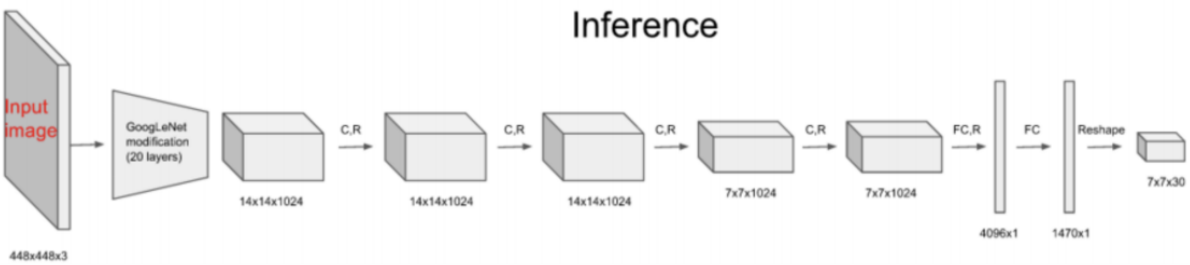
$\mathbb{1}_{ij}^{obj}$ The *j*th bbox predictor in *cell i* is "responsible" for that prediction

$\mathbb{1}_{ij}^{noobj}$

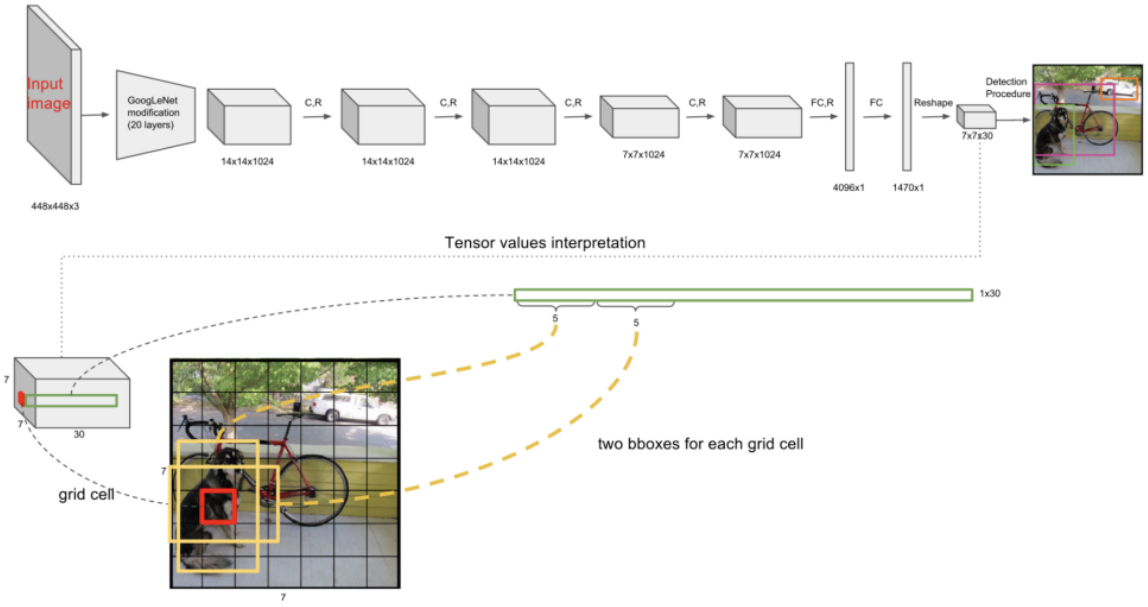
$\mathbb{1}_{i}^{obj}$ If object appears in *cell i*

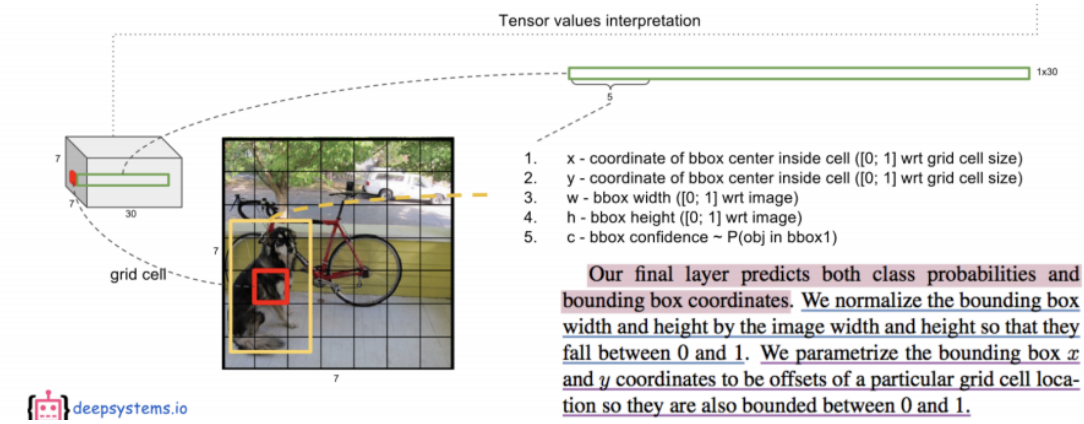
d. Yolo model





GoogleNet 변형 featrue extraction,
컨볼루션 레이어 4회,
풀 커넥션 레이어 2번
사이즈를 7x7x30



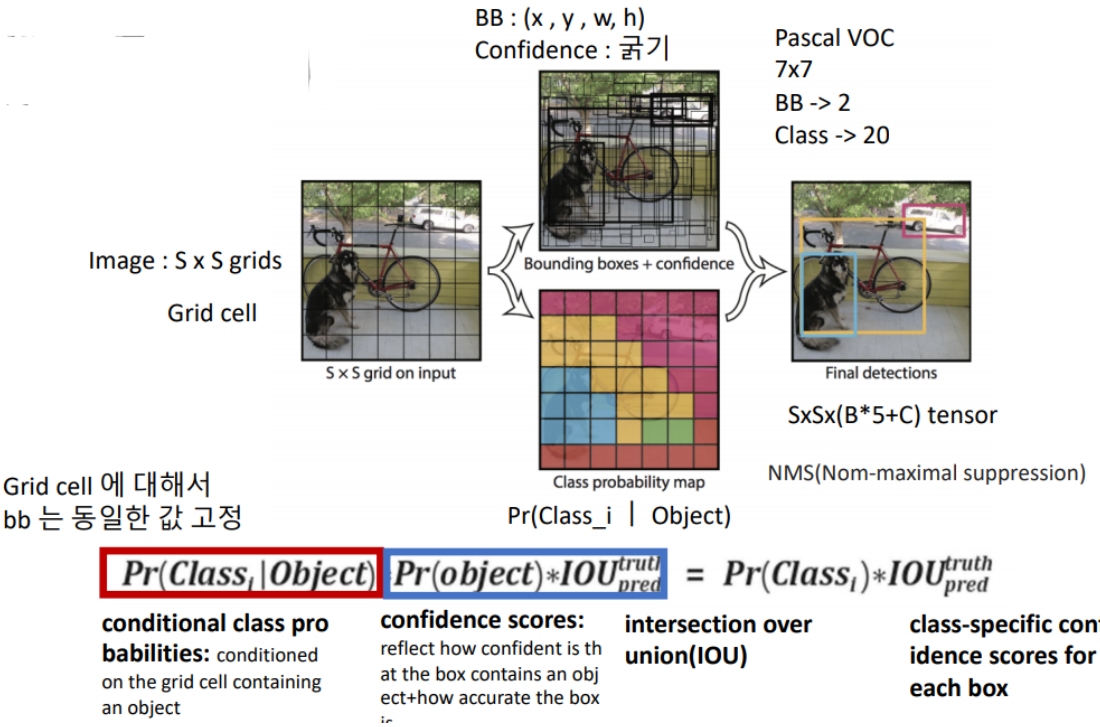


1. x - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
2. y - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
3. w - bbox width ([0; 1] wrt image)
4. h - bbox height ([0; 1] wrt image)
5. c - bbox confidence ~ P(obj in bbox1)

Our final layer predicts both class probabilities and bounding box coordinates. We normalize the bounding box width and height by the image width and height so that they fall between 0 and 1. We parametrize the bounding box x and y coordinates to be offsets of a particular grid cell location so they are also bounded between 0 and 1.

deepsystems.io

20



e. 정확도를 높이기 위한 해결방안

1) Fast R-CNN 과 Yolo 의 결합

Combining Fast R-CNN and YOLO

YOLO makes far fewer background mistakes than Fast R-CNN. By using YOLO to eliminate background detections from Fast R-CNN we get a significant boost in performance. For every bounding box that R-CNN predicts we check to see if YOLO predicts a similar box. If it does, we give that prediction a boost based on the probability predicted by YOLO and the overlap between the two boxes.

The best Fast R-CNN model achieves a mAP of 71.8% on the VOC 2007 test set. When combined with YOLO, its

	mAP	Combined	Gain
Fast R-CNN	71.8	-	-
Fast R-CNN (2007 data)	**66.9**	72.4	.6
Fast R-CNN (VGG-M)	59.2	72.4	.6
Fast R-CNN (CaffeNet)	57.1	72.1	.3
YOLO	63.4	**75.0**	**3.2**

Table 2: Model combination experiments on VOC 2007. We examine the effect of combining various models with the best version of Fast R-CNN. Other versions of Fast R-CNN provide only a small benefit while YOLO provides a significant performance boost.

4. LSTM(Long Short-Term Memory)

Learning to Forget: Continual Prediction with LSTM

Felix A. Gers

J ¨urgen Schmidhuber

Fred Cummins

IDSIA, 6900 Lugano, Switzerland

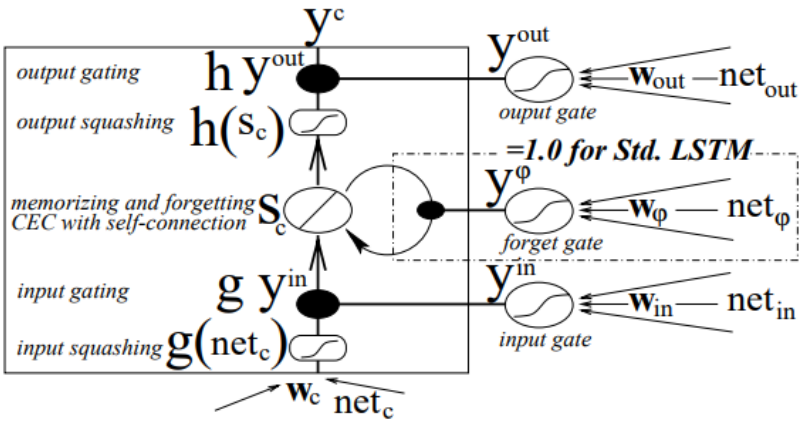
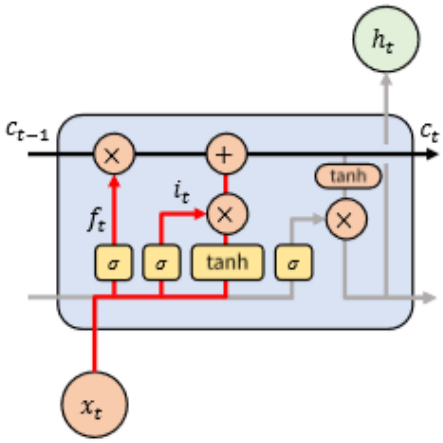


Figure 1: The standard LSTM cell has a linear unit with a recurrent self-connection with weight 1.0 (CEC). Input and output gates regulate read and write access to the cell whose state is denoted s_c. The function g squashes the cell's input; h squashes the cell's output. See the text for details.

output gates, respectively. When gates are closed (activation around zero), irrelevant inputs and noise do not enter the cell, and the cell state does not perturb the remainder of the network. Figure 1 shows a memory block with a single cell. The cell state, s_c, is updated based on its current state and three sources of input: net_c is input to the cell itself, while net_{in} and net_{out} are inputs to the input and output gates.

We consider discrete time steps $t = 1, 2, \ldots$. A single step involves the update of all units (forward pass) and the computation of error signals for all weights (backward pass). Input gate activation y^{in} and output gate activation y^{out} are computed as follows:

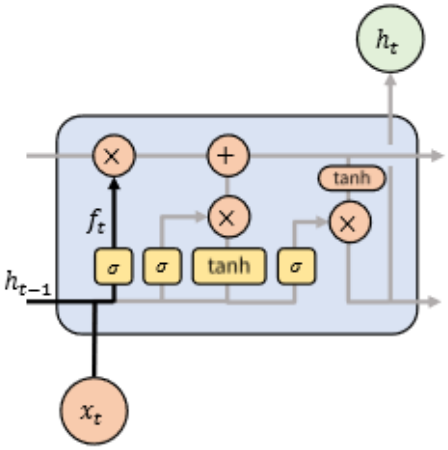
Cell state



$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$
$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$
$$g_t = \tanh(W_{xg}x_t + W_{hg}h_{t-1} + b_g)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$
$$h_t = o_t \odot \tanh(c_t)$$

Cell state는 '기억'을 총괄하는 메모리 역할을 한다. 여러 차원으로 되어있어, 각 차원은 특정 정보를 기억한다. Hadamard 연산자의 특성으로 인해, 특징 별로 기억하고, 잊고, 새로이 정보를 받을 수 있다.

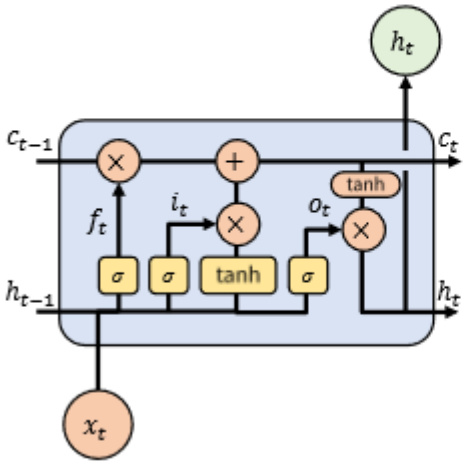
Forget gate



$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$
$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$
$$g_t = \tanh(W_{xg}x_t + W_{hg}h_{t-1} + b_g)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$
$$h_t = o_t \odot \tanh(c_t)$$

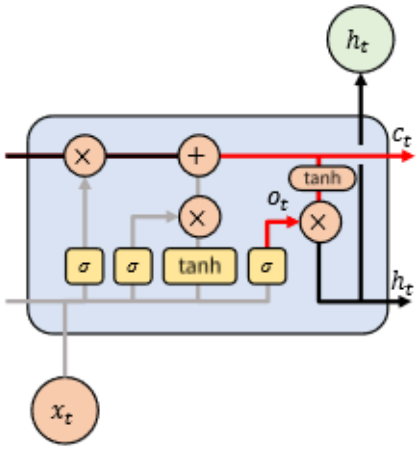
Forget gate는 기억을 '잊고자 하는 정도'를 나타낸다. Sigmoid activation이므로 값의 범위는 0~1이다. 특징은 여러 차원으로 되어 있으므로, 특징별로 기억할지 말지를 결정할 수 있다.

LSTM 수식



$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$
$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$
$$g_t = \tanh(W_{xg}x_t + W_{hg}h_{t-1} + b_g)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$
$$h_t = o_t \odot \tanh(c_t)$$

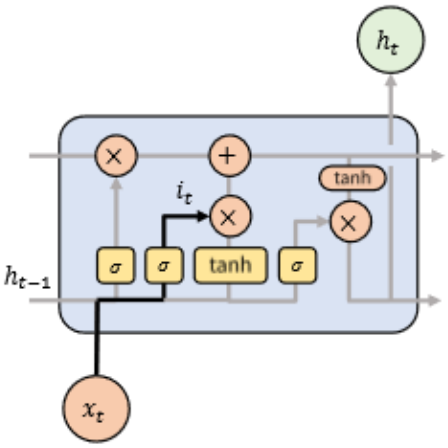
Hidden state



$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$
$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$
$$g_t = \tanh(W_{xg}x_t + W_{hg}h_{t-1} + b_g)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$
$$h_t = o_t \odot \tanh(c_t)$$

Hidden state는 Cell state에 tanh activation을 적용한 후, Output gate로 선별하여 출력한다.
tanh을 사용하는 이유는 출력 값의 범위가 -1~1로 bound되게 하기 위함이다.

Input gate



$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$
$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$
$$g_t = \tanh(W_{xg}x_t + W_{hg}h_{t-1} + b_g)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$
$$h_t = o_t \odot \tanh(c_t)$$

Input gate는 새로운 입력을 받고자 하는 정도를 나타낸다. Sigmoid activation이므로 값의 범위는 0~1이다.
특징은 여러 차원으로 되어 있으므로, 특징별로 받아들일지 말지를 결정할 수 있다.

조 TEAM 설명

조 이름 (KMUAI_YOLO)

*반드시 KMUAI_필수, XXX 는 자유롭게 선택, 해당 ID 로 최종 결과 확인 예정

조원 1 의 M2020527 김용환 50%

조원 2 의 M2020415 이옥걸 50%

각 조원의 업무 분장

김용환 : 알고리즘 기획 및 문제 해결 전략 제시 팀장

이옥걸 : 알고리즘 설계 및 해결 전략 제시

<코드> 데이터 만드는 코드

make_input.py

```
from matplotlib import pyplot as plt
from mpl_toolkits import mplot3d
import numpy as np
import pandas as pd
import os
import numpy as np
import csv
import glob
import pandas as pd
import os
import os


def open_file(path,encode):
    csv = pd.read_csv(path, encoding=encode)
    csv = csv.values
    return csv


def data_set_class(data):
    Accident_set = []
    Non_Accident_set = []
    for i in data:
        if (i[1] == 1):
            name = i[0]
            # print(name)
            filename = name.strip('.mp4')
            file_data = open_file('task1/%s/%s-mp4-acc.csv' %
(filename, filename), 'utf-8')
            for j in file_data:
                time = str(j[0])
                x = np.float32(j[1])+10
                y = np.float32(j[2])+10
                z = np.float32(j[3])+10
                Accident_set.append([x,y,z])

        elif (i[1] == 0):
            name = i[0]
            # print(name)
            filename = name.strip('.mp4')
```

```
            file_data = open_file('task1/%s/%s-mp4-acc.csv' %
(filename, filename), 'utf-8')
            for j in file_data:
                for k in j:
                    time = str(j[0])
                    x = np.float32(j[1])+10
                    y = np.float32(j[2])+10
                    z = np.float32(j[3])+10
                    Non_Accident_set.append([x, y, z])



    return Accident_set,Non_Accident_set
    # print(Accident_set)

def label_set(set,label):

    N = len(set)
    print(N)
    ano_data = []

    for i in range(N):
        ano_data.append(label)

    yy = np.array(ano_data)
    return yy

def save_file(acc_set,non_acc_set):

    for i, label in enumerate(["acc", "non_acc"]):
        # 3가지에 이것에 대해 하겠다.(3개의 문자열 원소를 가진 리스트)
        # data = enumerate([1, 2, 3])
        # for i, value in data:
        # print(i, ":", value)
        # print()

        # 파일 패스 csv 파일 읽어오고 3가지에 대해서 (먼저 웍부터)

        # 어노테이션 csv 파일을 하나 더 생성함.

        # 아웃풋 파일이름: xx_ +윈도우 사이즈_ +threshold_ +
label .csv
        outputfilename1 = "./input_files/xx_"  + label +
".csv"
        outputfilename2 = "./input_files/yy_"  + label +
".csv"

        # x,y = dataimport (파일 패스1,2 에서) 임포트 한다. 다시
이게 진짜인듯
```

```
        # x,y = dataimport(filename,path1)  # 여기서 x와 y를
만든다.
        if(label == 'acc'):

            with open(outputfilename1, "w") as f:
                writer = csv.writer(f, lineterminator="\n")  #
쓰고 다음줄로 넘어가나보다.
                writer.writerows(acc_set)  # x를 쓴다.
            with open(outputfilename2, "w") as f:
                writer = csv.writer(f, lineterminator="\n")
                labeling = [1,0,0]
                labeling = np.float32(labeling)
                y = label_set(acc_set,labeling)
                writer.writerows(y)  # y를 쓴다.
            print(label + "finish!")
        elif label == "non_acc":

            with open(outputfilename1, "w") as f:
                writer = csv.writer(f, lineterminator="\n")  #
쓰고 다음줄로 넘어가나보다.
                writer.writerows(non_acc_set)  # x를 쓴다.
            with open(outputfilename2, "w") as f:
                writer = csv.writer(f, lineterminator="\n")
                labeling = [0,1,0]
                labeling = np.float32(labeling)
                y = label_set(non_acc_set, labeling)
                writer.writerows(y)  # y를 쓴다.
            print(label + "finish!")

# main
if __name__ == '__main__':
    data_set_label =
open_file('task1/data_set_01_labeling_result.csv', 'EUC_KR')
    accident_set, Non_accident_set =
data_set_class(data_set_label)
    # print(accident_set)
    # accident distance
    acc_dis = []
    # Non_accident distance
    Nonacc_distance = []

    # draw_3D(accident_set,Non_accident_set)

    save_file(accident_set,Non_accident_set)

    fo = open_file('input_files/xx_acc.csv', 'utf-8')

    for i in fo:
        print(type(i[0]))
```

<모델 트레이닝>

```
import tensorflow as tf
import pandas as pd
import numpy as np
from itertools import chain
EPOCHS = 1000
NUM_WORDS = -10000
from sklearn.model_selection import train_test_split
import tensorflow.keras
from tensorflow.keras.utils import to_categorical
import torch


def open_file(path, encode):
    csv = pd.read_csv(path, encoding = encode)
    csv = csv.values
    res = []

    for i in csv:
        res.append(i)

    return res

class MyModel(tf.keras.Model):

    def __init__(self):
        super(MyModel, self).__init__()

        self.emb = tf.keras.layers.Embedding(-NUM_WORDS, 16)  # 길이가
10000 인 16 으로 바꿔준다. float 으로 바꾸기 위해.
        self.rnn = tf.keras.layers.LSTM(32)
        # self.tmp = tf.keras.layers.Bidirectional(
            # self.rnn, merge_mode='concat', weights=None,
backward_layer=None
            # )
        self.dense = tf.keras.layers.Dense(3, activation='softmax')  #
감정분석, 길이가 2 인 softmax 함수

    def call(self, x, training=None, mask=None):
```

```
        x = self.emb(x)
        x = self.rnn(x)
        # x = self.tmp(x)
        return self.dense(x)

# Implement training loop
@tf.function
def train_step(model, inputs, labels, loss_object, optimizer,
train_loss, train_accuracy):
    with tf.GradientTape() as tape:
        predictions = model(inputs, training=True)
        loss = loss_object(labels, predictions)
    gradients = tape.gradient(loss, model.trainable_variables)

    optimizer.apply_gradients(zip(gradients,
model.trainable_variables))
    train_loss(loss)
    train_accuracy(labels, predictions)

# Implement algorithm test
@tf.function
def test_step(model, images, labels, loss_object, test_loss,
test_accuracy):
    predictions = model(images, training=False)

    t_loss = loss_object(labels, predictions)
    test_loss(t_loss)
    test_accuracy(labels, predictions)


##main

# imdb = tf.keras.datasets.imdb#imdb 에서 가져온다.

# (x_train, y_train), (x_test, y_test) =
imdb.load_data(num_words=NUM_WORDS) #범위를 몇으로 할것인가. 정해준다.
x_train_csv = open_file("./input_files/xx_acc.csv","UTF-8") +
open_file("./input_files/xx_non_acc.csv","UTF-8")
y_train_csv = open_file("./input_files/yy_acc.csv","UTF-8") +
open_file("./input_files/yy_non_acc.csv","UTF-8")
x_train_csv = np.array(x_train_csv)
y_train_csv = np.array(y_train_csv)

x_train, x_test, y_train, y_test  =
train_test_split(x_train_csv,y_train_csv,test_size=0.2, random_state=
666)

print(x_train.shape)
```

```
print(y_train.shape)
    #x_train 길이가 x_test 와 다를수 있다 그래서 특정길이로 잘라줘야한다.
그길이에 미치지 못하는것들은 패딩도해줘야한다.
#앞으로 패딩을 해줄것이다.
x_train = tf.keras.preprocessing.sequence.pad_sequences(x_train,
                                                        dtype=float,
                                                       value=0,
                                                       padding='pre',
                                                       maxlen=120)
x_test = tf.keras.preprocessing.sequence.pad_sequences(x_test,
                                                       dtype=float,
                                                      value=0,
                                                      padding='pre',
                                                      maxlen=120)

print(x_train.shape)
print(y_train.shape)
#tensor_slices(x_train, y_train)을 가져올수 있따. 그리고 shuffle 하고
배치크기 32
x_train_ds = tf.data.Dataset.from_tensor_slices((x_train,
y_train)).shuffle(1000).batch(32)
x_test_ds = tf.data.Dataset.from_tensor_slices((x_test,
y_test)).batch(32) #test 도 그대로 한다.

# Create model
model = MyModel()
loss_object = tf.keras.losses.CategoricalCrossentropy()

optimizer = tf.keras.optimizers.Adam()

# Define performance metrics
train_loss = tf.keras.metrics.Mean(name='train_loss')
train_accuracy = tf.keras.metrics.CategoricalAccuracy()

test_loss = tf.keras.metrics.Mean(name='test_loss')
test_accuracy = tf.keras.metrics.CategoricalAccuracy()
# print(x_train)
for epoch in range(EPOCHS):
    for seqs, labels in x_train_ds:
        train_step(model, seqs, labels, loss_object, optimizer,
train_loss, train_accuracy)
        # print(train_step)

    for test_seqs, test_labels in x_test_ds:
        test_step(model, test_seqs, test_labels, loss_object,
test_loss, test_accuracy)
```

```
    template = 'Epoch {}, Loss: {}, Accuracy: {}, Test Loss: {}, Test
Accuracy: {}'
    print(template.format(epoch + 1,
                          train_loss.result(),
                          train_accuracy.result() * 100,
                          test_loss.result(),
                          test_accuracy.result() * 100))
    train_loss.reset_states()
    train_accuracy.reset_states()
    test_loss.reset_states()
    test_accuracy.reset_states()
```

결론 및 느낀점

인공지능 모델을 학습하면서 데이터 전처리 과정이 매우 재미가 있었으며 트레이닝과정중 차원을 축소하고 맞추는것에 의의를 두고 공부를 하였습니다. 완성된 결과물을 제출할수 있어서 보람찬 과제 였습니다.

(외부인) 보안 서약서

소속 : 국민대학교 성명 : 이옥걸

※ 보안 서약 및 출입 등의 관리를 위한 필수정보(소속,성명)를 수집 후 5 년간 보존함을 동의합니다.
(□동의 / □미동의)

본인은 주식회사 쏘카(이하 '회사'라 한다)의 업무를 수행함에 있어 보안에 관한 다음의 사항을 준수할 것을 엄숙히 서약합니다.

가. 정보보호 및 영업비밀보호 관련 준수사항을 숙지하였습니다. **(□예 / □아니오)**

1) 업무상 취득한 회사의 자산은 업무 목적 외 취급을 금하며, 불법으로 유출, 변조하거나 훼손하지 않겠습니다.

2) 업무상 취득한 회사 또는 제 3 자 소유의 정보를 회사의 승인 없이 타인에게 제공 등 누설하지 않겠습니다.

3) 업무상 회사 내부에 상주 또는 방문 시,
   (1) 회사의 보안구역 및 통제구역에 무단으로 출입하지 않겠습니다.
   (2) 회사의 통신망을 이용하여 외부인 접근이 금지된 타 회사의 통신망 또는 시스템 등의 매체에 임의로 접속을 시도하지 않겠습니다.
   (3) 회사의 정보통신(사내·외 이메일, 메신저 및 협업툴 등 일체의 대내·외적 정보통신 수단) 사용과 관련하여 정보보호를 위한 회사의 통제(사내·외 정보통신에 대한 열람, 내용의 기록, 보관 등 일체의 행위)에 대하여 이의 없이 동의하겠습니다.
   (4) 본인이 사용하는 PC, 노트북 등 정보처리기기에 최신 바이러스 백신 S/W 를 설치하여 반입하며, 필요시 회사에서 규정한 보안솔루션을 설치하고 이를 우회하는 시도를 하지 않겠습니다.
4) 계약 완료 시 또는 계약을 중도에 그만두게 된 경우, 본인이 보유하고 있는 모든 회사 업무 관련 자료(노트북 및 보조기업매체 저장파일 포함)를 반납 및 완전 삭제하며, 별도 사본 등을 보유하지 않겠습니다.
5) 기타 회사의 정보보호 및 영업비밀 보호 관련 규정을 준수하겠습니다.

나. 개인정보보호 관련 준수사항을 숙지하였습니다. **(☑예 / □아니오)**
1) 업무상 알게 된 개인정보를 허가없이 제 3 자에게 제공하거나 수집 목적 외로 이용하지 않겠습니다.
2) 명백히 허가 받지 않은 개인정보나 개인정보 취급 구역 및 시설에 접근하지 않겠습니다.
3) 업무와 관련한 개인정보의 수집, 생성, 기록, 저장, 보유, 가공, 편집, 검색, 출력, 정정, 복구, 이용, 제공, 공개, 파기 및 그 밖에 이와 유사한 일체의 행위에 대하여 기관의 규정과 통제절차를 준수하겠습니다.
4) 본인에게 할당된 출입증, 개인정보처리시스템 ID·패스워드를 타인과 공동 사용하거나 누출하지 않겠습니다.
5) 업무상 취급하는 개인정보자산(서류, 사진, 영상, 전자파일, 저장매체 등) 무단 변조, 복사, 훼손, 분실 등으로부터 안전하게 관리하겠으며, 승인을 받지 않은 프로그램 및 매체 등을 이용하여 취급하지 않겠습니다.
6) 퇴직 또는 유관업무 종료 시 업무상 제공받은 모든 정보자산을 반드시 반납할 것이며, 퇴직 후에도 퇴직전의 모든 개인정보에 대하여는 일체 누설하지 않겠습니다.
7) 기타 회사의 개인정보보호 관련 규정을 준수하겠습니다.

본인은 상기 사항을 숙지하여 이를 성실히 준수하겠으며, 위 사항을 이행치 않았을 경우 제 법규에 따른 책임은 물론 민·형사상 그리고 회사 인사규정에 따른 어떠한 처벌도 감수하고, 회사에 끼친 손해에 대해 지체 없이 변상·복구시킬 것을 서약합니다.

2021 년 4 월 20 일

서약자 성명 : 이윤호 (인)

주식회사 쏘카 대표이사 귀중