

개별연구 최종보고서

학생별 작성용

학생 현황			
수행 학기	□2020년 2학기		
프로젝트명	YOLOv3 알고리즘을 이용한 실시간 낙상 검출		
	학과	학번	성명
참여학생	컴퓨터공학전공	2014112027	이현동
	컴퓨터공학전공	2015112135	윤종연
	컴퓨터공학전공	2018112011	최수정
지도교수	교과목명	개별연구	
	소속	컴퓨터공학과	
	성명	신연순 교수님	

보고서					
작품명 (프로젝트명)	Zinnia (YOLOv3 알고리즘을 이용한 실시간 낙상 검출 프로그램)				
# Key Words	# YOLOv3	# 딥러닝	# 객체 인식	# 실시간성	# Zinnia
1.개발동기/ 목적/필요성및 개발 목표	<p>* 개발동기 및 목적(필요성) : 인적이 드문 장소에서 낙상 사고가 발생하는 경우 신고가 어려워 후속 조치가 잘 되지 않는 문제를 해결하기 위해 프로그램을 작성하게 되었습니다.</p> <p>* 개발목표 : YOLOv3 알고리즘을 활용하여 CCTV로 실시간 낙상 사고가 검출되면 알림을 발생하여 후속 조치를 취할 수 있는 프로그램을 작성하는 것이 개발목표입니다.</p>				
2.최종 결과물 소개	해당 문서에서는 낙상 이미지 데이터에 대한 학습을 진행하면서 어려움을 느낀 부분과 이에 대한 분석을 진행하였고, 최종 결과물은 같이 첨부한 논문에 기재하였습니다.				

3.프로젝트 추진 내용

- 저희는 당초 실시간 낙상 검출 프로그램을 만들기 위해 YOLOv3 모델을 낙상 이미지들로 학습시킨 후 생성된 weight 파일과 .dll 파일로 변환한 YOLOv3를 C++ 상으로 import하여 후속조치를 취하는 코드를 작성할 계획이었습니다.

- YOLOv3를 .dll파일로 변환시키는 과정은 성공적이었으나 YOLOv3 모델을 커스텀하여 학습시키는 과정에는 많은 문제가 발생하였습니다.

- 이에 저희가 YOLOv3를 커스텀하여 학습을 진행하기 위해 진행한 내용들을 정리하여 이와 같은 문제가 발생한 원인을 분석하였습니다.

- 아래 표는 데이터 학습과 실행을 한 PC의 환경입니다.

이현동 데스크톱	
프로세서	Intel(R) Pentium(R) CPU G4560 @ 3.50GHz 3.50GHz
메모리 (RAM)	8.00GB
운영체제	Windows 10 Home
GPU	NVIDIA GeForce GTX 1050, core : 4

윤종연 데스크톱	
프로세서	Intel(R) Core(TM) CPU i7-4790K @ 4.00GHz
메모리 (RAM)	16.00GB
운영체제	Windows 10 Pro
GPU	NVIDIA GeForce GTX 1060, 6GB

윤종연 노트북	
프로세서	Intel(R) Core(TM) CPU i5-9300H @ 2.40GHz
메모리 (RAM)	8.00GB
운영체제	Windows 10 Pro
GPU	NVIDIA GeForce GTX 1650

최수정 노트북	
프로세서	Intel(R) Core(TM) CPU i5-8250U @ 1.60GHz 1.80GHz
메모리 (RAM)	8.00GB
운영체제	Windows 10 Home
GPU	NVIDIA GeForce GTX 1060, 6GB

- 아래 내용은 저희가 데이터 학습을 위해 진행한 과정들을 운영체제 환경과 사용한 모델들을 통해 분류하여 정리한 것입니다.

1. window 기반 ubuntu + darknet + cpu

1.1) 환경 설정

- window 기반 ubuntu 앱 다운로드



- openCV 및 각종 라이브러리 설치

* 참고 사이트 : <https://webnautes.tistory.com/1186>

* 명령어

```
sudo apt-get install build-essential cmake
sudo apt-get install pkg-config
sudo apt-get install libjpeg-dev libtiff5-dev libpng-dev
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libxvidcore-dev
libx264-dev libxine2-dev
sudo apt-get install libv4l-dev v4l-utils
sudo apt-get install libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev
sudo apt-get install libgtk2.0-dev
sudo apt-get install mesa-utils libgl1-mesa-dri libgltk2.0-dev libgltkglext1-dev
sudo apt-get install libatlas-base-dev gfortran libeigen3-dev
sudo apt-get install python2.7-dev python3-dev python-numpy python3-numpy
mkdir opencv
cd opencv
wget -O opencv.zip https://github.com/opencv/opencv/archive/4.2.0.zip
unzip opencv.zip
wget -O opencv_contrib.zip https://github.com/opencv/opencv_contrib/archive/4.2.0.zip
unzip opencv_contrib.zip
- usr\local\lib\pkgconfig 에서 opencv4.pc 이름을 opencv.pc로 변경

cmake -D CMAKE_BUILD_TYPE=RELEASE
-D CMAKE_INSTALL_PREFIX=/usr/local
-D WITH_TBB=OFF
-D WITH_IPP=OFF
-D WITH_1394=OFF
-D BUILD_WITH_DEBUG_INFO=OFF
-D BUILD_DOCS=OFF
-D INSTALL_C_EXAMPLES=ON
-D INSTALL_PYTHON_EXAMPLES=ON
-D BUILD_EXAMPLES=OFF -D BUILD_TESTS=OFF
-D BUILD_PERF_TESTS=OFF
-D WITH_QT=OFF
-D WITH_GTK=ON
-D WITH_OPENGL=ON
-D OPENCV_EXTRA_MODULES_PATH=../../opencv_contrib-4.2.0/modules
-D WITH_V4L=ON
-D WITH_FFMPEG=ON
```

```
-D WITH_XINE=ON
-D BUILD_NEW_PYTHON_SUPPORT=ON
-D OPENCV_GENERATE_PKGCONFIG=ON ./
코어 수 확인 : cat /proc/cpuinfo | grep processor | wc -l
컴파일 : time make -j4
설치 : sudo make install
설정 파일 확인 : cat /etc/ld.so.conf.d/*
opencv 라이브러리 사용 설정 : sudo ldconfig
```

- darknet 다운로드 및 설정

* 참고 사이트 : <https://j-remind.tistory.com/60>

* 명령어

```
다운로드 : git clone https://github.com/pjreddie/darknet
cpu 사용 설정 : cd darknet → sudo vim Makefile → OPENCV = 1
make
darknet test : wget https://pjreddie.com/media/files/yolov3.weights
→ ./darknet detect cfg/yolov3.cfg yolov3.weights data/dog.jpg → 완료
```

```
GPU=0
CUDNN=0
OPENCV=1
OPENMP=0
DEBUG=0
```

* cfg파일 수정

```
[train]
batch = 64 , subdivision = 16
max_batches = 4(class 개수) * 2000 = 8000
steps = 8000*0.8 , 8000*0.9
network = 608.608
[yolo layer]
class = 4
[convolution layer]
filters = (4+5)*3 = 27
```

```
batch=64
subdivisions=16 max_batches = 8000
width=608 policy=steps
height=608 steps=6400,7200
```

```
[yolo]
mask = 3,4,5
anchors = 10,1
classes=4
```

```
[convolutional]
size=1
stride=1
pad=1
filters=27
```

* obj.names 파일 생성

```
falling_person
fallen_person
sitting_pseron
picking_person
```

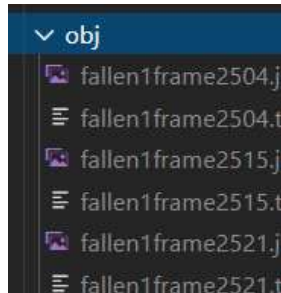
```
data > ≡ obj.names
1 falling_person
2 fallen_person
3 picking_person
4 sitting_person
```

* obj.data 파일 생성

```
classes = 4
train = data/train.txt
names = data/obj.names
backup = backup/
```

```
data > ≡ obj.data
1 classes= 4
2 train = data/train.txt
3 valid = data/test.txt
4 names = data/obj.names
5 backup = backup/
```

- * 이미지 파일 및 labeling 파일 data/obj에 삽입



- * train.txt파일 생성

clone https://github.com/AlexeyAB/Yolo_mark
 linux_mark.sh파일에 이미지 경로 수정
 train.txt파일 생성 : bash linux_mark.sh
 darknet 폴더로 이동

```
data > train.txt
1 data/obj/fallen10frame170.jpg
2 data/obj/fallen10frame21.jpg
3 data/obj/fallen10frame31.jpg
4 data/obj/fallen10frame53.jpg
5 data/obj/fallen11frame219.jpg
6 data/obj/fallen11frame239.jpg
7 data/obj/fallen11frame283.jpg
8 data/obj/fallen11frame301.jpg
9 data/obj/fallen11frame31.jpg
10 data/obj/fallen11frame349.jpg
```

- * pretrained 모델 다운로드

darknet53.conv.74 다운로드

1.2) 학습

- 명령어 : ./darknet detector train data/obj.data cfg/yolov3.cfg darknet53.conv.74
- 학습 시간 : 약 100시간
- weight파일 생성 : 100번째 weight 생성

yolov3.backup
 yolov3_100.weights

1.3) 결과

- 결과 화면



1.4) 학습 환경

- darknet 학습은 최수정 팀원의 노트북, yolov3 실행은 윤종연 팀원의 노트북으로 진행하였습니다.

2. window + anaconda + keras

2.1) 환경 설정

* 참고 사이트 1 : <https://readystory.tistory.com/115>

* 참고 사이트 2 : <https://nero.devstory.co.kr/post/pj-too-real-03/#Image-Generating>

- anaconda window 64bit 다운로드

- 가상 환경 구축 및 keras 설치

```
conda create -n keras2 python =3.6 anaconda
conda install -n keras2 numpy matplotlib pandas pydotplus h5py scikit-learn
conda install -n keras2 scipy theano mkl-service libpython m2w64-toolchain
conda install -n keras2 git graphviz
pip install --ignore-installed --upgrade tensorflow
pip install keras
```

- darknet model을 keras모델로 변환

* 명령어 : `python convert.py -w yolov3.cfg yolov3.weights model_data/yolo_weights.h5`

- annotation파일(train_all) 생성 :

* xml 파일에서 이미지 파일 경로와 xmin, xmax, ymin, ymax 만 추출하여 train_all.txt에 저장

```
C:\Users> chdir C:\Users\chlt\3_2> cctv_monitoring_yolo > preconditioning_process > resource
1 <annotation>
2   <folder>dotable</folder>
3   <filename>fallen1frame2504.jpg</filename>
4   <path>C:\Users\chlt\3_2\cctv_monitoring_yolo\precon
5   <source>
6     <database>Unknown</database>
7   </source>
8   <size>
9     <width>1920</width>
10    <height>1080</height>
11    <depth>3</depth>
12  </size>
13  <segmented>0</segmented>
14  <object>
15    <name>fallen_person</name>
16    <pose>Unspecified</pose>
17    <truncated>0</truncated>
18    <difficult>0</difficult>
19    <bndbox>
20      <xmin>1113</xmin>
21      <ymin>813</ymin>
22      <xmax>1298</xmax>
23      <ymax>904</ymax>
24    </bndbox>
25  </object>
26 </annotation>
```

[이미지 별 xml 파일]

```
# 2. generate new annotation file(train_all.txt) , classes.txt
# train.py 실행 resource폴더 안에 넣기
import xml.etree.ElementTree as ET
from os import getcwd
import glob

classes = ["person", "falling_person", "fallen_person", "sitting_person", "picking_person", "tie_person"]
image_id = "a"

def convert_annotation(annotation_voc, train_all_file):
    tree = ET.parse(annotation_voc)
    root = tree.getroot()

    for obj in root.iter('object'):
        difficult = obj.find('difficult').text
        cls = obj.find('name').text
        if cls not in classes or int(difficult) == 1:
            continue
        cls_id = classes.index(cls)
        bbox = obj.find('bndbox')
        b = (int(bbox.find('xmin').text), int(bbox.find('ymin').text), int(bbox.find('xmax').text), int(bbox.find('ymax').text))
        train_all_file.write(" " + " ".join([str(a) for a in b]) + " " + str(cls_id))

train_all_file = open('resources/train/train_all.txt', 'w')
# 1. image labeling한 xml파일 폴더를 만들어주세요. (예> falling_person 이면 resource/train/xml/falling_person/ 폴더)
# 2. 모든 클래스, 이미지에 대한 xml 파일이 (resources/train/train_all.txt 파일에 들어가 있게 되실겁니다)
for className in classes:
    annotations_voc = glob.glob('resources/train/xml/{className}/*.xml')
    for annotation_voc in annotations_voc:
        image_id = annotation_voc.split('/')[-1].split('.')[0].split('.')[1]
        print(annotation_voc.split('/')[-1].split('.')[0].split('.')[1].split('.')[2])
        train_all_file.write('resources/train/frames/{className}/{image_id}')
        convert_annotation(annotation_voc, train_all_file)
        train_all_file.write('\n')
train_all_file.close()
```

[annotation 파일 생성 코드]

```
train_all - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
./resources/train/frames/falling_person/fallen10frame21.JPG 1483,372,1551,497,0
./resources/train/frames/falling_person/fallen10frame31.JPG 1473,393,1560,504,0
./resources/train/frames/falling_person/fallen10frame53.JPG 1473,412,1555,505,0
./resources/train/frames/falling_person/fallen11frame31.JPG 1382,616,1469,771,0
./resources/train/frames/falling_person/fallen11frame39.JPG 1382,627,1471,783,0
./resources/train/frames/falling_person/fallen11frame57.JPG 1387,661,1466,774,0
./resources/train/frames/falling_person/fallen12frame17.JPG 1388,620,1458,772,0
./resources/train/frames/falling_person/fallen12frame3.JPG 1388,606,1466,757,0
./resources/train/frames/falling_person/fallen12frame32.JPG 1376,663,1465,760,0
./resources/train/frames/falling_person/fallen12frame43.JPG 1383,680,1455,791,0
./resources/train/frames/falling_person/fallen14frame32.JPG 1508,420,1558,535,0
./resources/train/frames/falling_person/fallen14frame40.JPG 1498,441,1562,545,0
./resources/train/frames/falling_person/fallen14frame49.JPG 1505,456,1565,535,0
```

[annotation file (train_all.txt)]

- keras의 train.py 파일 수정


* keras 파일 다운로드 : <https://github.com/qqwweee/keras-yolo3>

* train.py 에서 main부분의 파일 경로 부분 수정


```
def _main():
    # new annotation file ( 앞에서 생성 ) -----
    annotation_path = 'resources/train/train_all.txt'
    # file to store log
    log_dir = 'logs/000/'
    # new classes file
    classes_path = 'resources/train/classes.txt'
    # anchors path
    anchors_path = 'model_data/yolo_tiny_anchors.txt'
    transfer_learning_path = 'model_data/yolo_tiny.h5'
    class_names = get_classes(classes_path)
    num_classes = len(class_names)
    anchors = get_anchors(anchors_path)
```

2.2) 학습

- h5 파일 생성
- * 명령어 : python train.py
- * 결과 파일 : trained_weights_final.h5

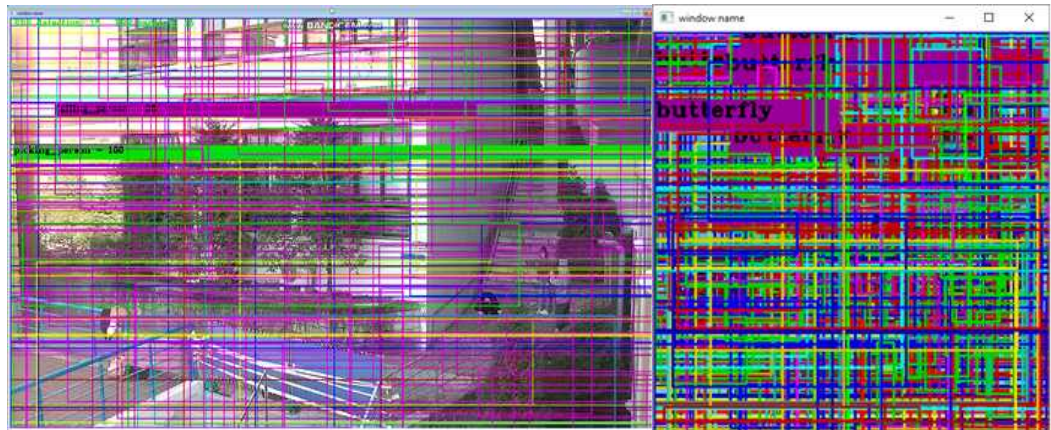
 trained_weights_final.h5

 trained_weights_stage_1.h5

- h5 to weight
- * 참고 사이트 : <https://github.com/HimariO/Keras-2-Darknet>
- * 명령어 : python k2day.py yolo.cfg trained_weights_final.h5 yolov3.weights

 yolov3.weights

2.3) 결과



2.4) 학습 환경

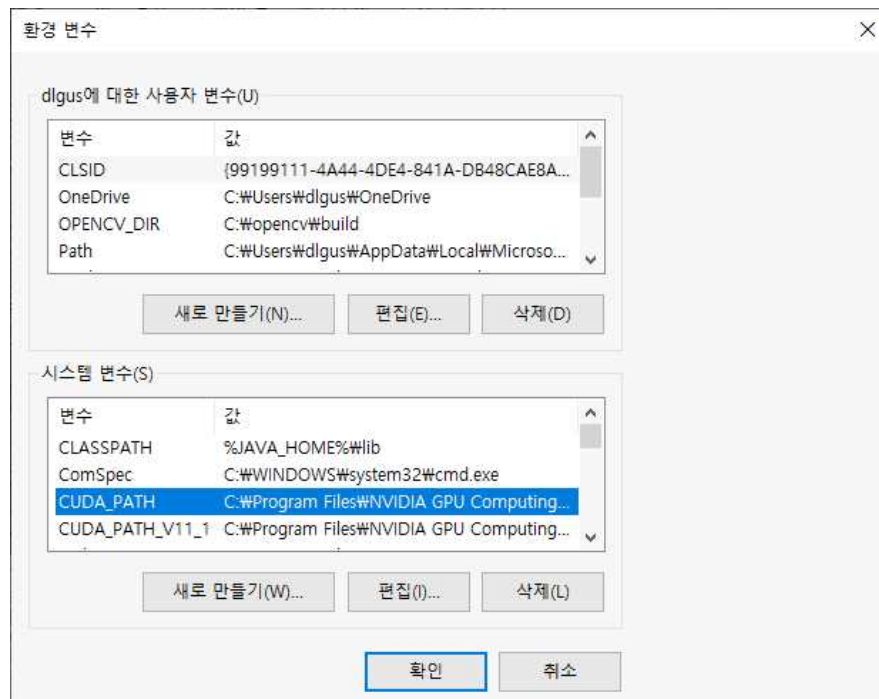
- darknet 학습은 최수정 팀원의 노트북에서, yolov3 실행은 윤종연 팀원의 노트북에서 진행하였습니다.

3. window + YOLO_mark + darknet-master

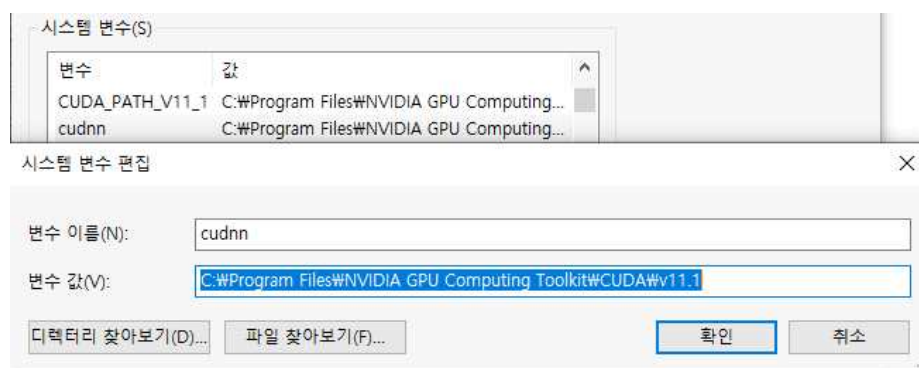
- 이현동 데스크톱에서 진행

3.1) CUDA, cuDNN 설치

현재 사용 중인 GPU의 Compute capability는 6.1 Pascal이기 때문에 이와 호환되는 CUDA SDK 11.1 버전을 다운받았습니다. 또한 cuDNN은 CUDA 11.1과 호환되는 8.0.5 버전을 설치하였고, 이에 대한 환경변수가 설정된 것을 확인 하였습니다.



cuDNN을 CUDA와 연동하기 위해 cuDNN의 bin, include, lib폴더에 있는 .dll, .h, .lib 파일들을 CUDA의 bin, include, lib폴더에 복사하여 넣어주었고, cuDNN의 환경변수를 설정해주었습니다.

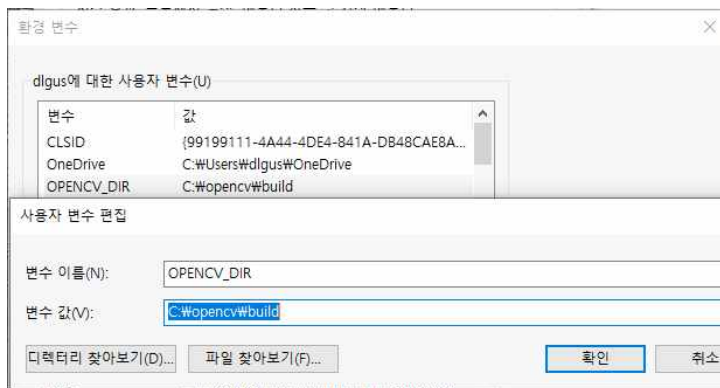


3.2) OpenCV 설치

OpenCV의 3.4.x 버전중에 가장 최신 버전인 3.4.12 버전을 설치하여 압축을 풀었습니다. 그리고 시스템 변수의 path를 OpenCV의 .dll 파일이 있는 경로를 추가하였습니다.



또한 Visual Studio에서 사용하기 위한 사용자 변수 OPENCV_DIR을 추가해 주었습니다.

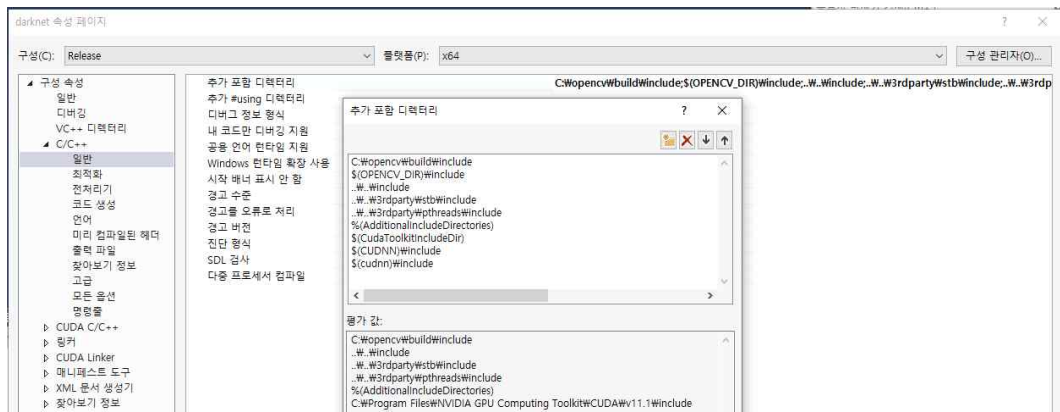


3.3) darknet-master 설치

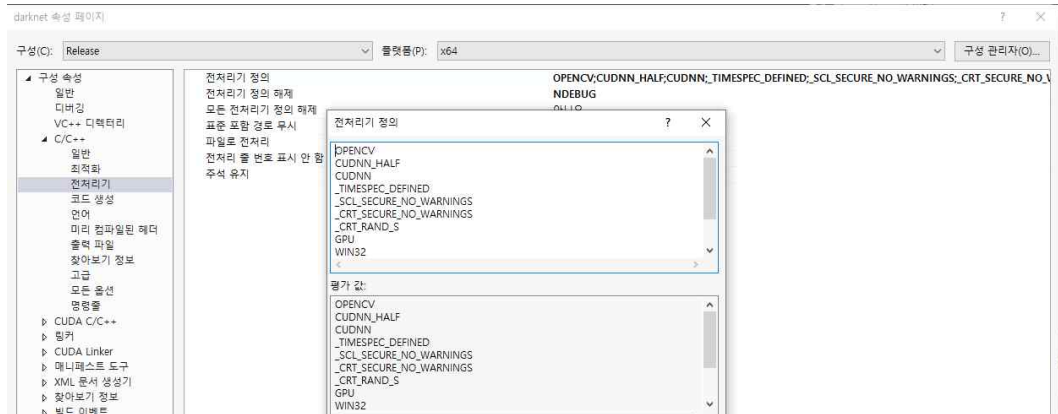
darknet GitHub 링크(<https://github.com/AlexeyAB/darknet>)에서 darknet-master.zip 파일을 받은 후 압축을 풀어 C:\workspace\ 폴더 안에 넣어주었습니다.

이후 darknet\build\darknet\ 안에 있는 darknet.sln 파일을 Visual Studio 2017을 통해 실행하였습니다.

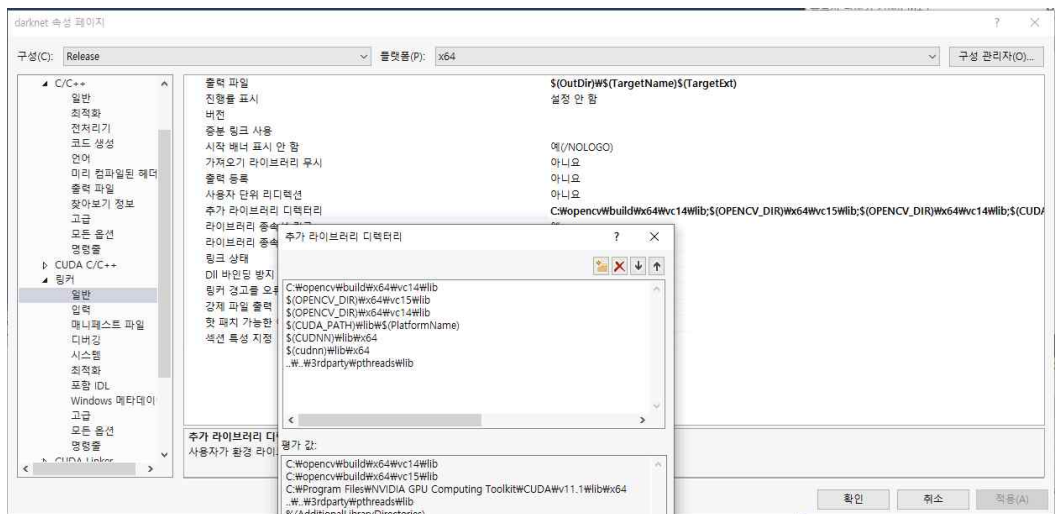
해당 솔루션 파일의 (프로젝트-속성)에서 Release, Debug 두 가지 구성 모두 (C/C++-일반-추가 포함 디렉터리)에 OpenCV와 CUDA, cuDNN을 추가하였습니다.



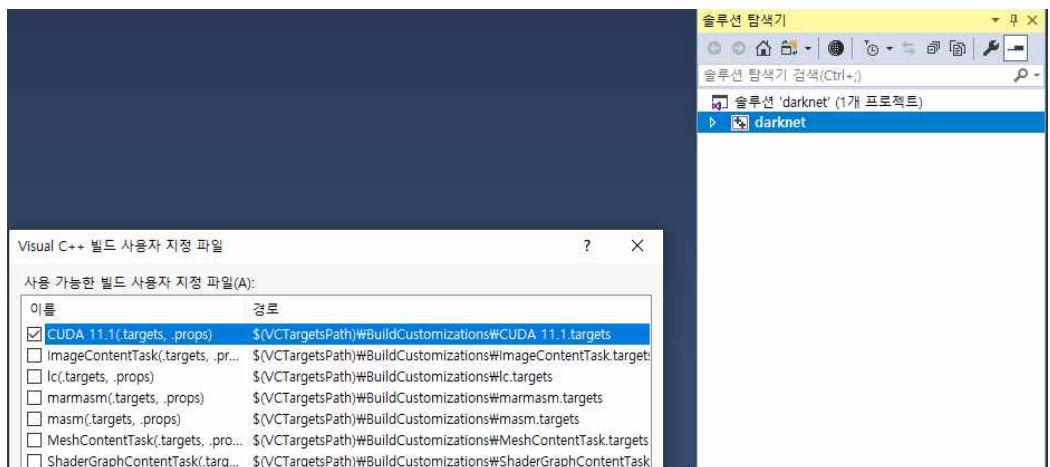
(C/C++-전처리기-전처리기 정의)에서 OPENCV와 CUDNN을 Release, Debug 모두에 추가해주었습니다.



(링커-일반-추가 라이브러리 디렉터리)에서 기존에 있던 opencv 3.0을 새로운 opencv library 디렉토리의 경로를 Release, Debug 모두에 추가하여 주었습니다.



(darknet-빌드 종속성-사용자 지정 빌드)에서 CUDA 11.1만 체크를 해주었습니다.



이제 프로젝트를 빌드해 주면 다음과 같이 성공하게 되고

```
1> 0 functions had inline decision re-evaluated but remain unchanged
1>코드를 생성했습니다.
1>darknet.vcxproj -> C:\workspace\darknet-master\darknet-master\build\darknet\x64\darknet.exe
===== 빌드: 성공 1, 실패 0, 최신 0, 생략 0 =====
|
```

C:\workspace\darknet-master\darknet-master\build\darknet\x64
darknet.exe이라는 실행파일이 생성됩니다.

경로에

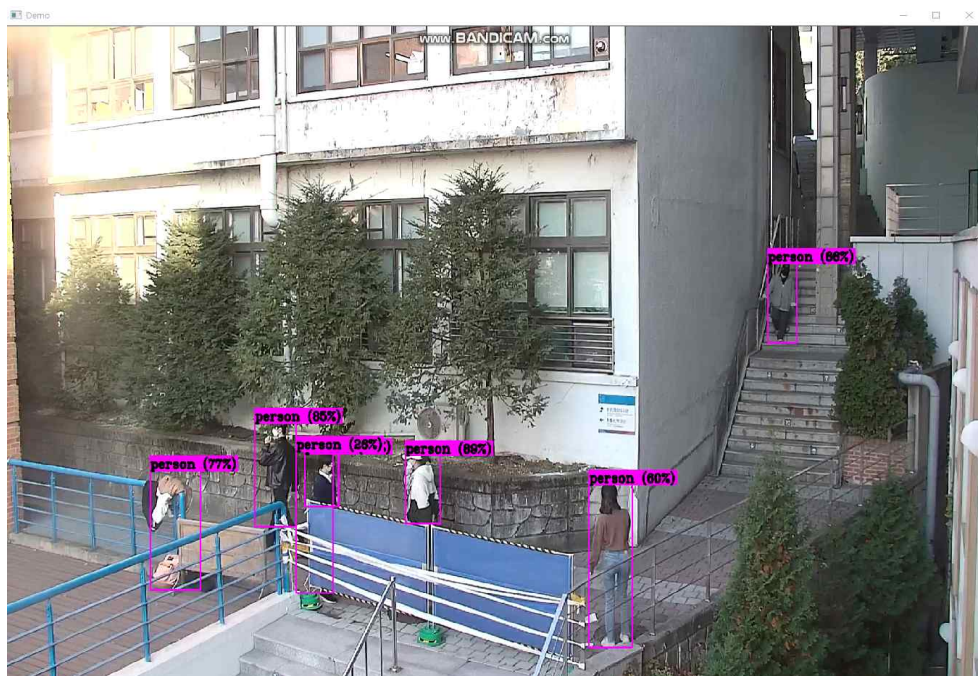
darknet	2020-12-17 오후 7:04	응용 프로그램
darknet.iobj	2020-12-15 오전 2:50	IOBJ 파일
darknet.ipdb	2020-12-15 오전 2:50	IPDB 파일
darknet.pdb	2020-12-17 오후 7:04	PDB 파일
darknet	2020-12-14 오전 7:19	Python File

YOLOv3의 배경 설정이 성공적으로 완료 되었는지 확인하기 위해 yolov3.weight 파일을 링크(<https://pjreddie.com/media/files/yolov3.weights>)를 통해 다운받았습니다.

그리고 weight 파일을 exe 파일이 있는 폴더에 저장한 다음 같은 폴더에 opencv_ffmpeg340_64.dll 파일과 opencv_world340.dll 파일을 추가하였습니다.

테스트를 진행하기 위해 darknet-master\build\darknet\x64\data 위치에 video 폴더를 생성한 후 test.mp4 파일을 넣어주었습니다. 그리고 cmd를 통해 darknet.exe 파일이 있는 위치까지 cd를 하여 진입한 다음 아래와 같은 명령어를 입력해주면
darknet.exe detector demo data/coco.data cfg/yolov3.cfg yolov3.weights -i 0
data/video/test.mp4

다음과 같이 YOLOv3가 실행되는 것을 볼 수 있었습니다.



이제 설치된 YOLOv3를 학습하기 위한 설정을 진행하였습니다. Yolo_mark를 설치하여 yolo_mark.sln 파일을 Visual Studio 2017을 통해 실행한 다음 앞서 darknet설정에 진행한 내용들을 프로젝트 속성을 통해 변경해 주었습니다. 모든 설정을 끝낸 다음 솔루션 파일을 빌드해 주면 다음과 같이 yolo_mark.exe 파일이 생성되었습니다.

yolo_mark	2019-04-24 오전 2:59	Windows 명령어 ...	1KB
yolo_mark	2020-12-15 오후 1:58	응용 프로그램	113KB
yolo_mark.iobj	2020-12-15 오후 1:58	IOBJ 파일	2,859KB
yolo_mark.ipdb	2020-12-15 오후 1:58	IPDB 파일	895KB

이미지 라벨링 작업을 진행하기 위해 Opencv_world348.dll, opencv_ffmpeg348.dll 파일을 yolo_mark.exe가 있는 폴더에 복사해 준 다음 dataWimg 폴더에 학습할 이미지 파일들을 넣고 data 파일에 있는 obj.names 파일을 열어 classes들을 지정해 주고 obj.data 파일을 열어 classes의 수를 변경해 주었고 같은 폴더에 있는 yolo_mark cmd 프로그램을 실행하였습니다.



위 사진과 같이 이미지에 직접 boundary box를 설정하여 class를 지정해 주면 해당 이미지가 있는 폴더에 boundary box 위치가 저장되어있는 roi 파일과 해당 이미지와 roi 파일의 경로를 알려주는 train.txt 파일 생성됩니다.

학습을 진행하기 위해 img폴더와 obj.data, obj.names, train.txt 파일을 darknet-master\build\darknet\x64\data 폴더에 추가한 다음 yolov3.cfg 파일을 다음과 같이 수정해 주었습니다.

```
6 batch=8
7 subdivisions=64
8 width=416
9 height=416
```

```

20 max_batches = 6200
21 policy=steps
22 steps=4800,5400

```

```

[convolutional]
size=1
stride=1
pad=1
filters=24
activation=linear

```

```

[yolo]
mask = 6,7,8
anchors = 24, 52, 40, 40, 50, 64, 90, 51, 63, 82, 71, 77, 125, 59, 83,100, 134, 80
classes=3

```

batch는 4의 배수, subdivisions는 2의 배수로 batch는 클수록, subdivisions는 작을수록 학습이 정교해지지만 더 좋은 하드웨어 환경을 원하기 때문에 8과 64로 설정하였습니다.

max_batches는 class의 수*2000에 여유분 200을 더해 6200으로 설정해 주었습니다.

steps는 max_batches의 80%, 90%로 4800, 5400으로 설정해 주었습니다.

총 3가지의 yolo의 class값과 anchor값, 그리고 convolutional의 filters를 수정해 주었습니다.

classes는 현재 클래스의 수, anchors는 cmd 창에서 다음과 같은 코드를 입력하여 darknet.exe detector calc_anchors data/obj.data -num_of_clusters 9 -width 416 -height 416 얻은 값을 입력해 주었습니다.

filters는 (5+class의 수)*3 으로 계산해 24를 입력하였습니다.

가중치 파일 darknet53.conv.74를 다운받아 같은 폴더에 넣어 모든 준비를 마친 뒤 다음 명령어를 입력하면 학습이 진행됩니다.

```

darknet.exe detector train data\obj.data testcfg\yolov3.cfg data\darknet53.conv.74 -gpu 0

```

하지만 제 PC의 하드웨어가 좋지 않아 out of memory오류가 발생하였습니다.

```

CUDA Error: out of memory

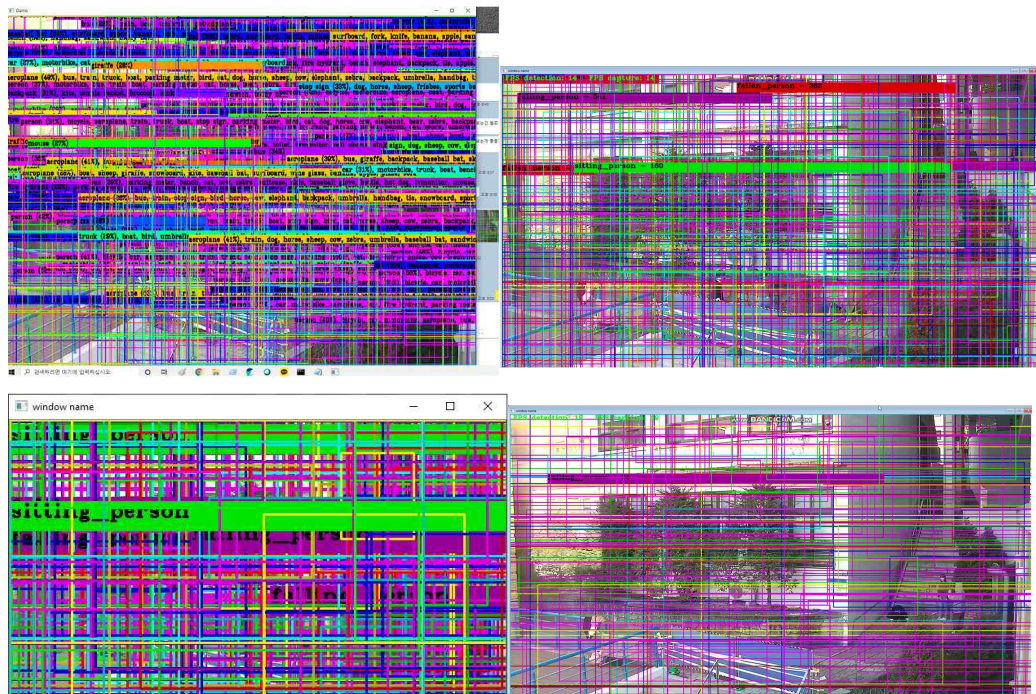
```

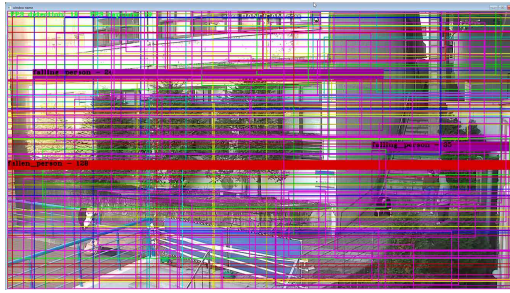
이를 해결하기 위해 이미지 파일의 개수를 줄이고 batch는 줄이고 subdivision은 늘려서 명령어를 다시 입력하였습니다. 하지만 아래 사진과 같이 학습 그래프는 생성되었지만 점이 찍히지 않고 out of memory가 뜨는 오류가 발생했습니다.

학습을 통해 얻어진 weight파일을 실행해 본 결과 다음과 같은 실행화면이 나왔습니다.



여러 방법으로 학습을 진행하여 실행하였지만 원하는 결과물을 얻지 못해 저희는 YOLOv3.cfg의 내용을 여러 경우의 수를 통해 수정해 보았고, 학습 이미지와 클래스의 수를 늘리고 줄이고를 하며 다수의 학습을 진행하였지만 아래와 같은 결과밖에 얻지 못 하였습니다.





저희는 이와 같이 이미지 데이터 학습이 잘 진행되지 못한 이유를 다음과 같이 분석하였습니다.

1. 충분하지 않은 이미지 데이터

저희는 크롤링하여 얻은 낙상 관련 이미지 66장과 직접 촬영한 영상을 프레임으로 나눠 얻은 50장의 이미지 데이터밖에 확보하지 못하였고 이는 모델을 학습시키기에 부족한 양이었다고 생각하였습니다.

2. 확실하지 않은 객체의 형태

저희가 직접 촬영한 영상에서는 주로 후드를 쓰고 있는 상태에서 촬영을 진행하여 객체의 형태가 명확하지 않아 제대로 학습이 되지 않았다고 생각합니다.

3. 데이터 학습을 진행하기에 충분하지 않은 하드웨어

저희는 학습을 진행하면서 out of memory 오류가 자주 발생하여 batch는 줄이고 subdivision은 늘려서 진행하여 객체가 명확하지 않게 인식되어 여기저기 많은 boundary box가 생성됐을 수도 있다고 생각하였습니다. 또한 다른 PC에서 진행한 내용을 세팅을 같게 했다고 해도 조금의 오차가 발생할 수도 있기 때문에 이 오차에서 발생한 오류 때문에 원하는 방향으로 학습이 안 됐을 수도 있다고 생각하였습니다.

4. 코드 수정의 오류

솔루션 파일의 컴파일이나 cmd 상에서의 error를 해결하는 과정에서 코드를 잘못 수정하였거나 백그라운드 설정에 오류가 있었을 수도 있다고 생각하였습니다.

5. 데이터 labeling의 오류

저희는 이미지 데이터를 labeling하기 위해 labelimg와 YOLO_mark를 사용하였습니다. 이 툴들을 처음 사용하였기 때문에 roi파일을 생성하는 과정에서 예상하지 못한 오류를 범해 객체의 boundary box가 제대로 설정되지 않았을 수도 있다고 생각하였습니다.

6. yolov3.cfg 파일의 변수 값

yolov3.cfg의 변수 값을 학습할 class에 따라 변경해야하지만 참고한 모든 사이트들이 조금씩 차이가 있어서 이를 설정함에 있어 오차가 발생하여 원하는 결과를 얻지 못하였을 수도 있다고 생각하였습니다.

<p>4.기대효과</p>	<p>낙상 사고로 인한 응급 환자 발생 시 빠른 후속조치를 취할 경우 생존율이 아무 조치를 하지 않았을 때보다 최대 3.3배 이상 나타납니다.[2] Zinnia가 낙상자를 잘 검출한다면 CCTV 상에서 낙상 사고가 발생할 경우 알림을 발생시켜 빠른 후속조치 가능해지기 때문에 낙상자의 생명을 구할 수 있는 기대 효과가 있습니다. 이에 대한 내용은 논문에 자세하게 기재하였습니다.</p>
<p>5.참고문헌</p>	<p>[1] 김정옥, 「낙상 사고 12월에 가장 많아...낙상 환자 중 50대 이상이 70%」, 서울경제, 2019.12.03, pp. 1. https://www.sedaily.com/NewsView/1VRXJI2624</p> <p>[2] 전진우, 「"심장마비 환자 심폐소생술부터" ... 생존율 최대 3.3배」, 동아닷컴, 2019.11.26, pp. 1. https://www.donga.com/news/Society/article/all/20191126/98538887/1</p> <p>[3] 최형진, 김정수, 「낙상사고 예방을 위한 국내·외 관련 규정 분석 연구」, RISS 학술연구정보서비스, 2014, pp. 1. http://www.riss.or.kr/search/detail/DetailView.do?p_mat_type=1a0202e37d52c72d&control_no=7b3887969f528d527f7a54760bb41745</p> <p>[4] 임세경, 「국토교통 R&D Report 오픈소스 하드웨어 기반 실내 환경 모니터링 시스템 개발 및 클라우드를 통한 지속적 자가 학습 딥러닝 시스템 구현」, 국토교통부 (국토교통과학기술진흥원), 2016.06.24, pp. 1. https://www.kaia.re.kr/portal/landmark/readTskFinalView.do?tskId=114867&yearCnt=2&menuNo=200100</p> <p>[5] 장덕성, 최승찬, 선주형, 김상현, 한송희, 「응급호출기의 개발과 낙상판단 알고리즘」, 한국HCI학회, 2011.01, pp. 1. http://www.dbpia.co.kr/Journal/articleDetail?nodeId=NODE01880334</p>
<p>6. 소감 및 향후 발전 방향</p>	<p>앞서 분석한 내용을 개선하여 낙상 데이터 학습을 성공적으로 진행할 것이고 이를 통해 당초 계획한 프로그램을 구현할 예정입니다.</p>