

Jetson TX2 기반 YOLO 응용 과정

- Day 3 -

2020.00



양재 R&CD 혁신허브
Yangjae Innovation Hub

 모두의연구소

목 차

01

Jetson TX2 Darknet 셋업

02

YOLOv3 실행 및 예제

03

YOLOv3를 통한 물체추적



01. Jetson TX2 Darknet 셋업

[1] Darknet 설치

① Darknet 소스코드를 Github에서 다운로드한다.

```
$ mkdir ~/project
$ cd ~/project
$ git clone https://github.com/pjreddie/darknet.git
$ cd darknet
```

② Makefile을 gedit으로 열어서 아래와 같이 수정 _TX2의 CUDA 아키텍처는 “62” 이다.

```
$ gedit Makefile
```

▶ (참조: <https://developer.nvidia.com/cuda-gpus>)



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

ARCH= -gencode arch=compute_62,code=[sm_62,compute_62]
#ARCH= -gencode arch=compute_30,code=sm_30 \
#      -gencode arch=compute_35,code=sm_35 \
#      -gencode arch=compute_50,code=[sm_50,compute_50] \
#      -gencode arch=compute_52,code=[sm_52,compute_52]
#      -gencode arch=compute_20,code=[sm_20,sm_21] \ This one is deprecated?
```

01. Jetson TX2 Darknet 셋업

[1] Darknet 설치

③ Make 명령어로 컴파일하면, darknet 파일이 생성된다.

```
$ make -j4  
$ sudo ldconfig
```

01. Jetson TX2 Darknet 셋업

[2] YOLOv3 실행 준비

- 일반적으로 물체 인식을 위해서는 두개의 파일이 필요하다
 - 1) 신경망 레이어 정보가 담긴 .cfg 파일
 - 2) 가중치 파라미터 정보가 담긴 .weight 파일
- darknet 소스 코드에서 cfg 파일이 존재하나, weights 파일이 없다.
따라서, weights 이름의 경로를 만들어 yolov3-tiny.weights 파일을 저장한다.

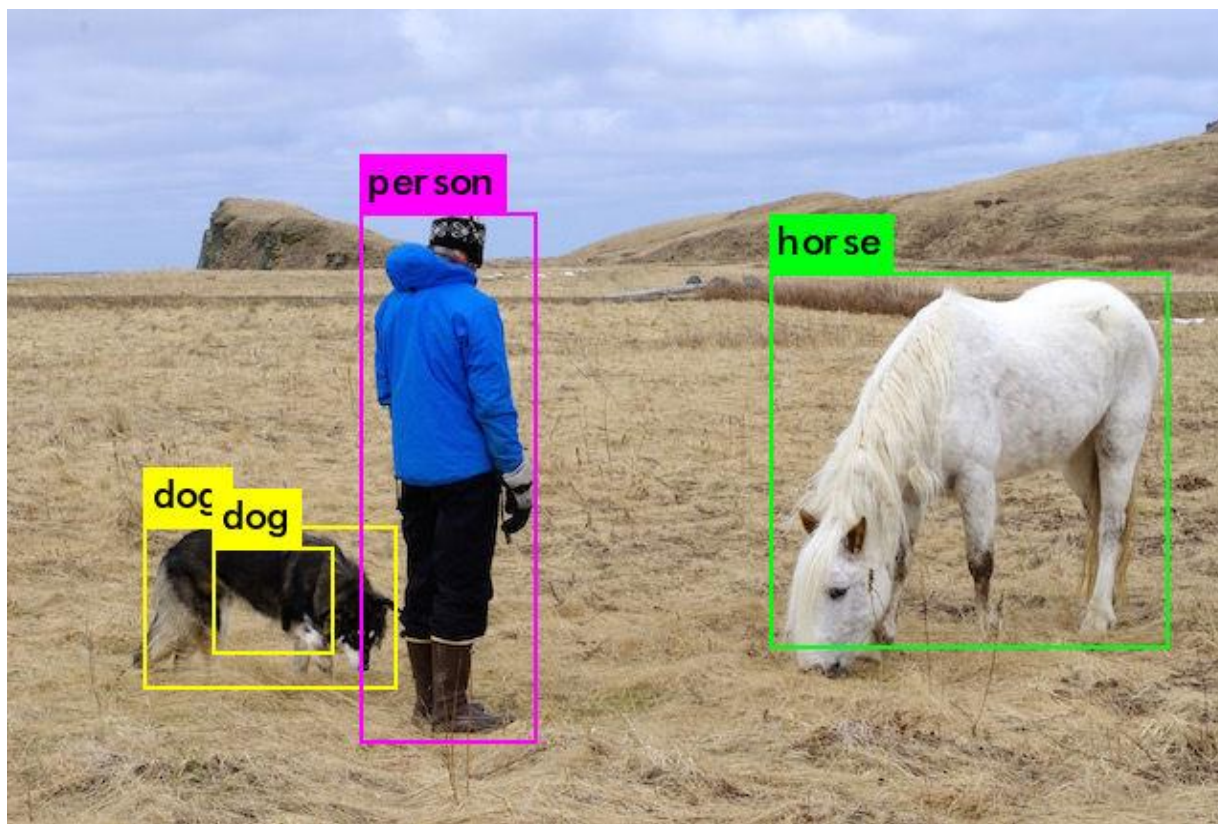
```
$ mkdir ~/project/darknet/weights  
$ cd ~/project/darknet/weights  
$ wget https://pjreddie.com/media/files/yolov3-tiny.weights
```

02. YOLOv3 실행 및 예제

[1] YOLOv3 실행

- .cfg 파일과 .weight 파일을 불러와 YOLOv3로 물체 인식을 한다.
- 맨 뒤에는 원하는 사진을 인터넷에서 다운로드하여 다양하게 직접 시험해 볼 수 있다.

```
$ ./darknet detect cfg/yolov3-tiny.cfg weights/yolov3-tiny.weights data/person.jpg
```

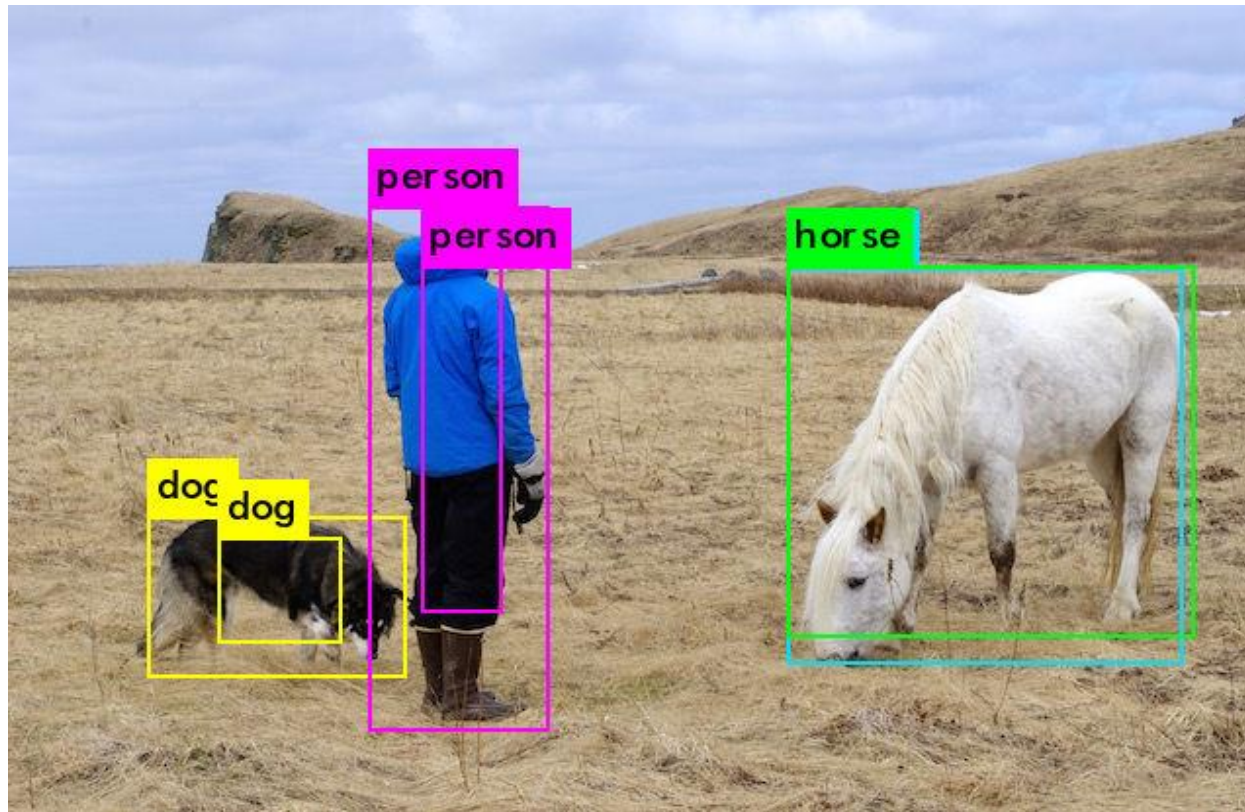


02. YOLOv3 실행 및 예제

[2] YOLOv3 실행 _ 임계치 설정

- 임계치 기본값은 “-thresh 0.5” 이다. 이를 “-thresh 0.1”로 수정 후 실행 해본다.
- thresh 0.1 : 인식률이 10% 이상인 것을 바운딩 박스로 보여준다.

```
$ ./darknet detect cfg/yolov3-tiny.cfg weights/yolov3-tiny.weights data/person.jpg -thresh 0.1
```

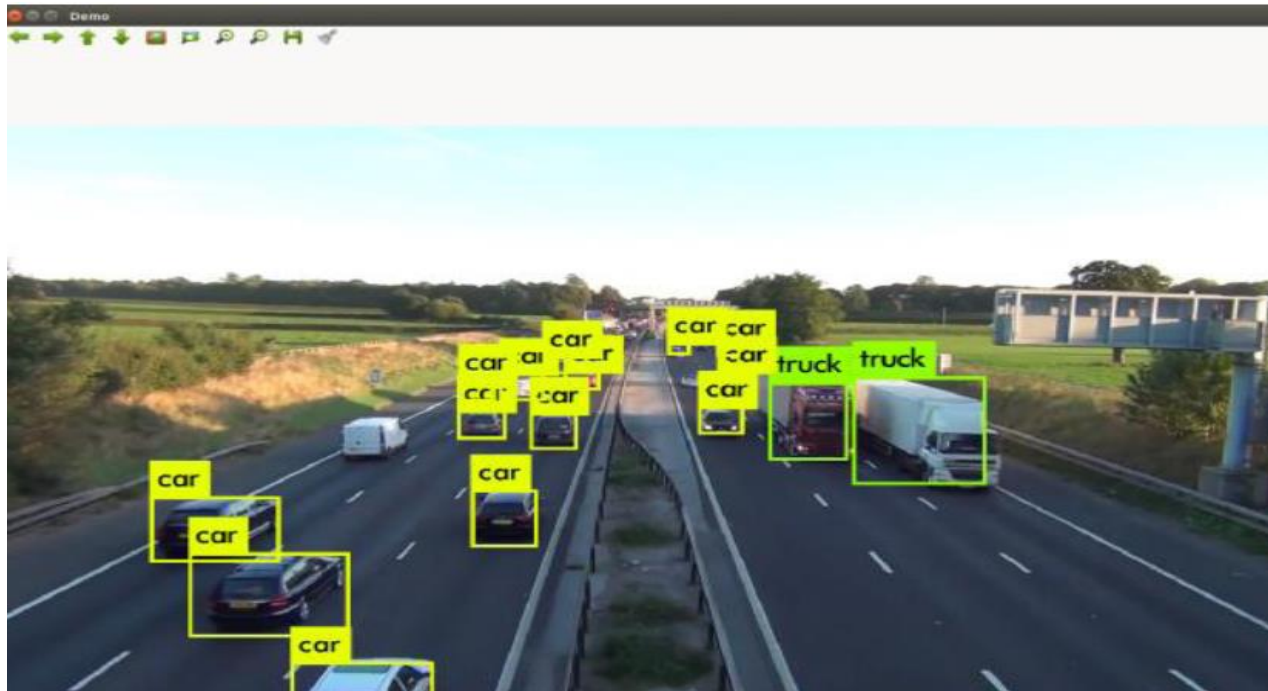


02. YOLOv3 실행 및 예제

[2] YOLOv3 실행 _ 웹캠 사용

```
$ ./darknet detector demo cfg/coco.data cfg/yolov3-tiny.cfg weights/yolov3-tiny.weights -c 1
```

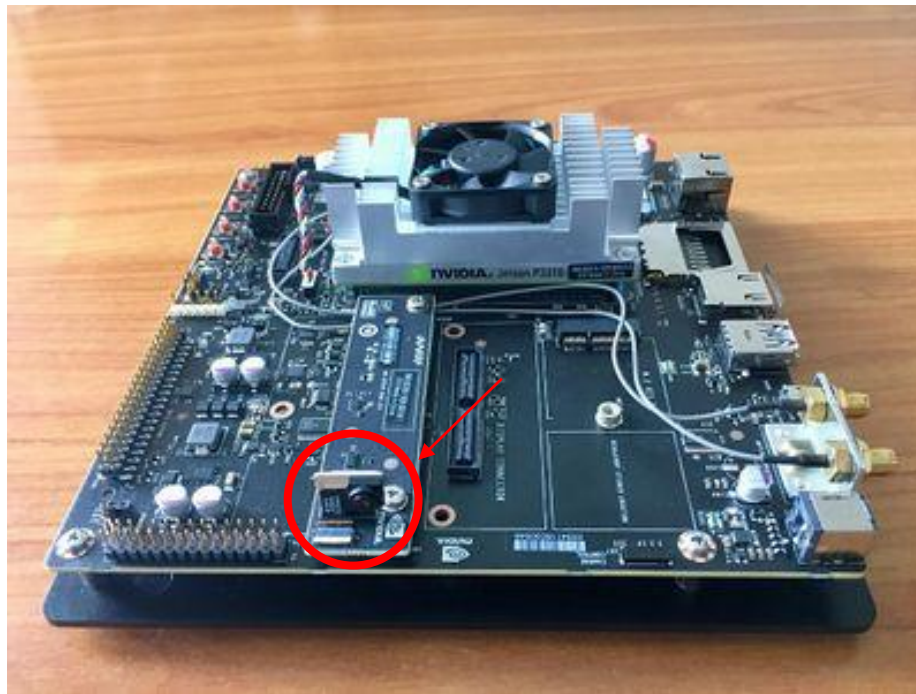
- 웹캠을 사용하여 실시간 영상 물체인식을 한다.
- **-c 1** 은 USB Camera 장치에 해당하는 `/dev/video1` 설정하는 것을 의미한다.
- 만일 작동하지 않으면, USB Camera 가 `/dev/video0` 이나 `/dev/video2`로 설정되어 있을 수 있다.
(비디오 장치 확인은 `<$ ls /dev/video*>` 커맨드로 확인할 수 있다.



02. YOLOv3 실행 및 예제

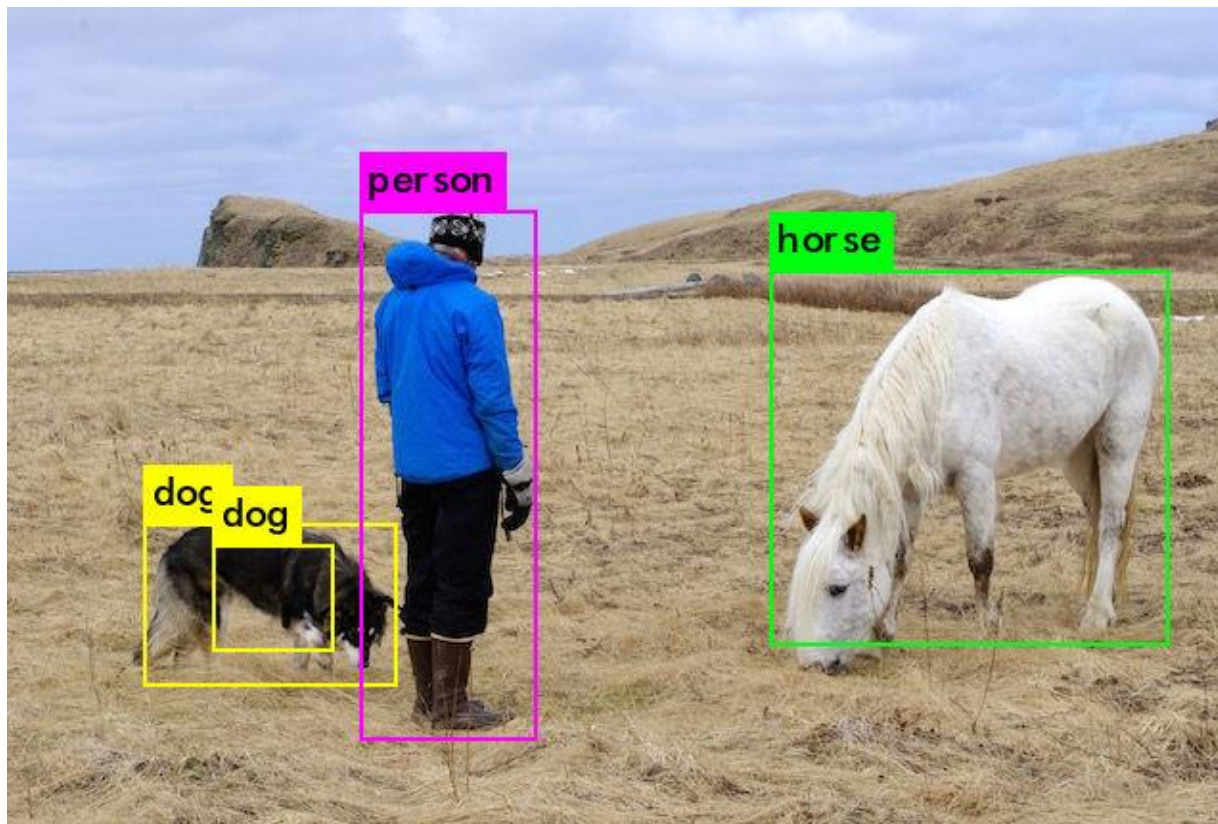
[2] YOLOv3 실행 _ Jetson TX2 온보드 카메라 사용

```
$ ./darknet detector demo cfg/coco.data cfg/yolov3-tiny.cfg weights/yolov3-tiny.weights "'nvarguscamerasrc  
! video/x-raw(memory:NVMM), width=1920, height=1080, format=(string)NV12, framerate=(fraction)30/1 ! nvtee  
! nvvidconv flip-method=0 ! video/x-raw, width=(int)1280, height=(int)720, format=(string)BGRx !  
videoconvert ! appsink'"
```



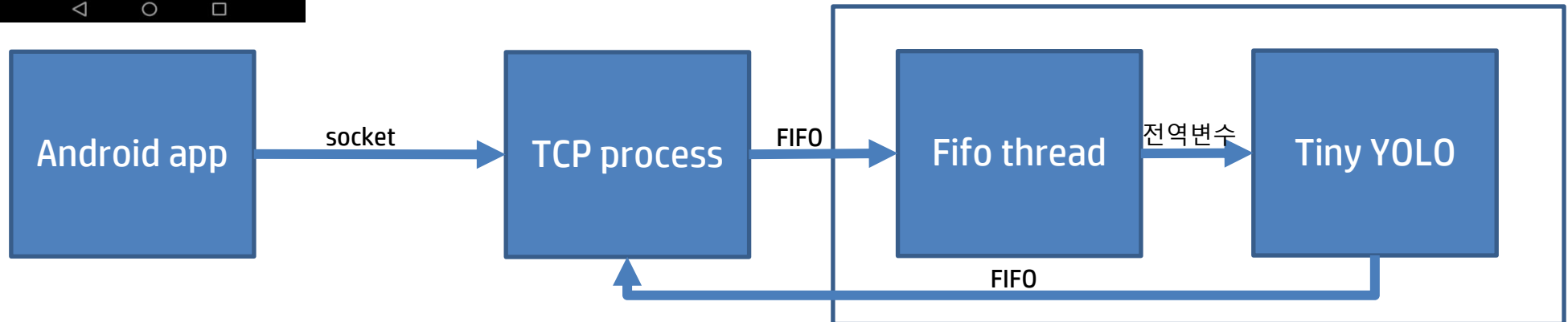
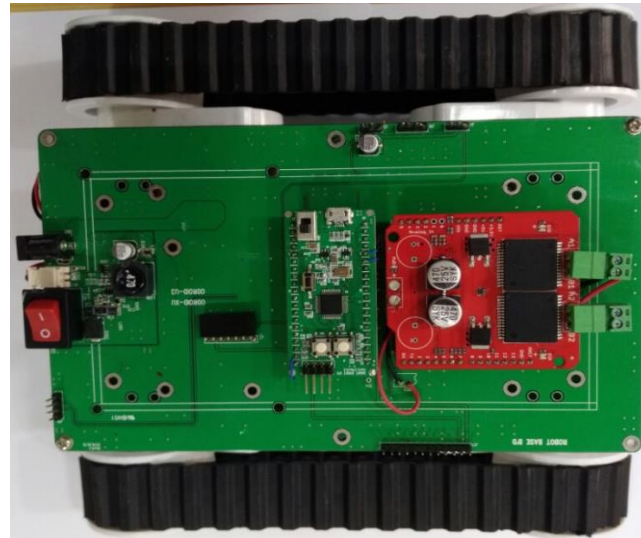
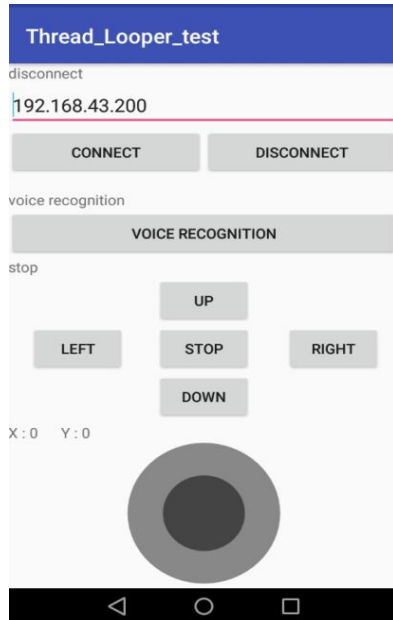
03. YOLOv3를 통한 물체추적

- ▶ Bounding Box 위치 = 물체의 위치
- ▶ Bounding Box 위치 값을 이용하여, 물체를 추적하는 알고리즘을 생성.
- ▶ Darknet 에서 Bounding Box 위치 값을 나타내는 함수는 **draw_detections()**
- ▶ 따라서, draw_detections() 함수를 확장하여, 물체추적 시스템 제작.



03. YOLOv3를 통한 물체추적

▶ 개요



03. YOLOv3를 통한 물체추적

Rover5_control

① e2gedu 카페 -> 임베디드A.I.-> Rover5 제어에서 Rover5_control.c 소스를 다운로드 한다.

② 컴파일 및 실행

```
$ cd ~/Downloads
$ gcc -o Rover5_control Rover5_control.c
$ sudo ./Rover5_control
```

@ 동작되지 않을 시



@ port 설정 변경

```
$ sudo minicom -s
```

- ① Serial port setup 선택.
- ② A 버튼 입력
- ③ tty8 → ttyTHS2 로 변경
- ④ save setup as dfl 선택
- ⑤ Exit 선택

03. YOLOv3를 통한 물체추적

Rover5_tcp

① e2gedu 카페 -> 임베디드A.I.-> rover5_tcp에서 rover5_tcp.c 소스를 다운로드 한다.

② 컴파일 및 실행

```
$ cd ~/Downloads
$ gcc -o rover5_tcp rover5_tcp.c -lpthread
$ sudo ./rover5_tcp
```

③ 안드로이드 폰에서 e2gedu 카페 -> 임베디드A.I.-> 와이파이 구동 어플리케이션의 autoCarControl2.apk를 다운 및 설치한다.

④ TX2보드의 ip를 확인 후 앱에 입력 -> CONNECT로 연결한다.

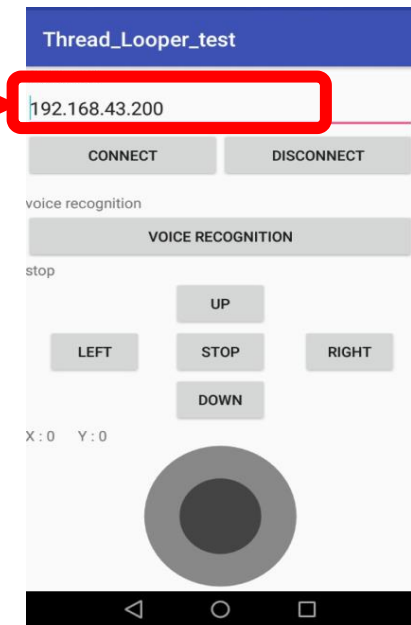
@TX2 ip 확인

```
$ ifconfig

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.16.0.140 netmask 255.255.255.0 broadcast 172.16.0.255
    ether 00:04:4b:c4:d7:29 txqueuelen 1000 (Ethernet)
    RX packets 13940 bytes 20528747 (20.5 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 5088 bytes 377254 (377.2 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

nvidia@nvidia-desktop:~$
```

*TX2가 와이파이를 사용하는 경우 wlan0의 inet주소를 확인



03. YOLOv3를 통한 물체추적

Yolo v3 코드 수정

① e2gedu 카페 -> 임베디드A.I.-> YOLO 코드 수정 내용 정리(Rover5 구동체)에서 소스코드를 다운

② darknet 폴더의 소스에 덮어쓰기

```
$ cd ~/Downloads
$ mv -f darknet.h ~/project/darknet/include
$ mv -f coco.c detector.c yolo.c ~/project/darknet/examples
$ mv -f demo.c image.c ~/project/darknet/src
```

③ Make 명령어로 컴파일

```
$ cd ~/project/darknet
$ make clean
$ make -j4
$ sudo ldconfig
```

03. YOLOv3를 통한 물체추적

Yolo v3 를 통한 물체추적

① darknet 실행(onboard camera)

```
$ cd ~/project/darknet
$ ./darknet detector demo cfg/coco.data cfg/yolov3-tiny.cfg weights/yolov3-tiny.weights
'''nvguscamerasrc ! video/x-raw(memory:NVMM), width=1920, height=1080, format=(string)NV12,
framerate=(fraction)30/1 ! nvtee ! nvvidconv flip-method=0 ! video/x-raw, width=(int)1280,
height=(int)720, format=(string)BGRx ! videoconvert ! appsink'''
```

②새 터미널을 열고 rover5_tcp 실행

```
$ cd ~/Downloads
$ sudo ./rover5_tcp
```

* 추적물체 변경 시

```
$ gedit ~/project/darknet/src/demo.c
```

Line22의 target_class_a 의 값을 변경 후 컴파일 (아래표 참조)

03. YOLOv3를 통한 물체추적

- ▶ YOLO에서 사용하는 객체명은 classlist 버퍼에 아래와 같이 정의 되어 있다.
/data/coco.names

0	person,	20	elephant,	40	wine glass,	60	diningtable,
1	bicycle,	21	bear,	41	cup,	61	toilet,
2	car,	22	zebra,	42	fork,	62	tvmonitor,
3	motorbike,	23	giraffe,	43	knife,	63	laptop,
4	aeroplane,	24	backpack,	44	spoon,	64	mouse,
5	bus,	25	umbrella,	45	bowl,	65	remote,
6	train,	26	handbag,	46	banana,	66	keyboard,
7	truck,	27	tie,	47	apple,	67	cell phone,
8	boat,	28	suitcase,	48	sandwich,	68	microwave,
9	traffic light,	29	frisbee,	49	orange,	69	oven,
10	fire hydrant,	30	skis,	50	broccoli,	70	toaster,
11	stop sign,	31	snowboard,	51	carrot,	71	sink,
12	parking meter,	32	sports ball,	52	hot dog,	72	refrigerator,
13	bench,	33	kite,	53	pizza,	73	book,
14	bird,	34	baseball bat,	54	donut,	74	clock,
15	cat,	35	baseball glove,	55	cake,	75	vase,
16	dog,	36	skateboard,	56	chair,	76	scissors,
17	horse,	37	surfboard,	57	sofa,	77	teddy bear,
18	sheep,	38	tennis racket,	58	pottedplant,	78	hair drier,
19	cow,	39	bottle,	59	bed,	79	toothbrush

Object 변경시 :
demo.c 의 target_class_a 의 값을 변경 후
\$ make

- 영상 인식 객체명 전달

앱->TX2(Wifi)	TX2(Wifi)->YOLO	Class index
apple	A	47
banana	B	46
bicycle	C	1
dog	D	16

- 모터 제어 명령 전달

	YOLO>TX2(WiFi)	TX2(WiFi)->STM32
left	a	a
right	b	b
up	c	c
down	d	d
stop	i	i

03. YOLOv3를 통한 물체추적

▶ `draw_detections()` 함수가 프로토 타입 선언, 정의, 호출되고 있는 위치

선언	함수를 사용할 수 있도록 미리 알리는 부분
	<code>darknet/include/darknet.h</code>

정의	함수가 수행하는 기능이 작성된 부분
	<code>darknet/src/image.c</code>

호출	함수를 사용하는 부분
	<code>darknet/src/demo.c</code> TCP 프로세스와 통신을 위해서 thread 및 fifo 생성 및 사용
	<code>darknet/examples/detector.c</code> 단순히 인자만 확장
	<code>darknet/examples/coco.c</code> 단순히 인자만 확장
	<code>darknet/examples/yolo.c</code> 단순히 인자만 확장

03. YOLOv3를 통한 물체추적

▶ 요약

1. thread 및 fifo 생성 - src/demo.c
2. draw_detections() 함수 프로토 타입 수정 - include/darknet.h
 - return : void -> int (target class가 있는 경우 1)
 - target class와 좌표값 인자로 추가
3. draw_detections() 함수 원형 수정 - src/image.c
 - target class가 있는 경우 좌표값 인자로 전달
4. draw_detections() 함수 호출 후 모터 구동 - src/demo.c
 - return이 1인 경우 모터 구동 명령을 fifo를 통하여 tcp thread에 전달

examples/detector.c, examples/coco.c, examples/yolo.c : draw_detections() 호출 함수 수정
draw_detections(im, dets, nboxes, thresh, names, alphabet, l.classes, -1, NULL, NULL, NULL);

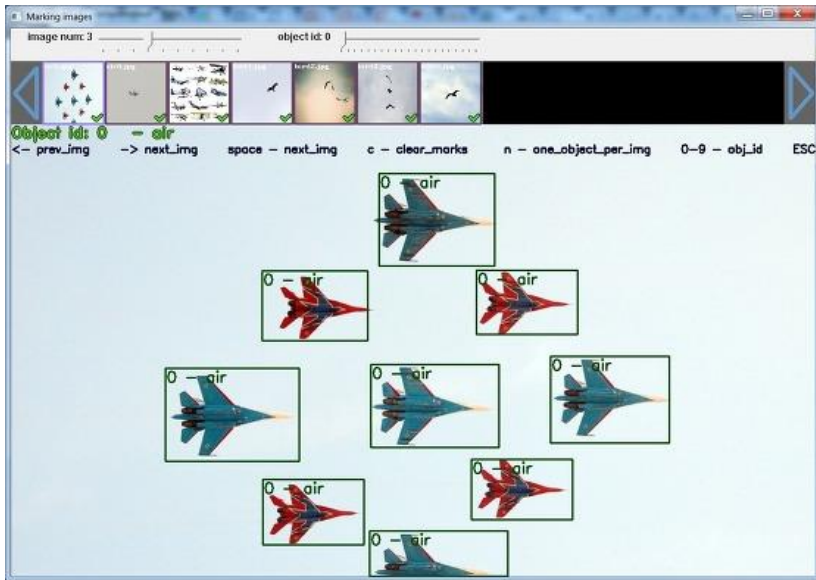
04. YOLOv3를 통한 데이터학습

① Yolo_mark 다운로드 및 설치

```
$ sudo apt install cmake
$ cd ~/project
$ git clone https://github.com/AlexeyAB/Yolo_mark.git
$ cd Yolo_mark
$ cmake .
$ make
```

② Yolo_mark 실행

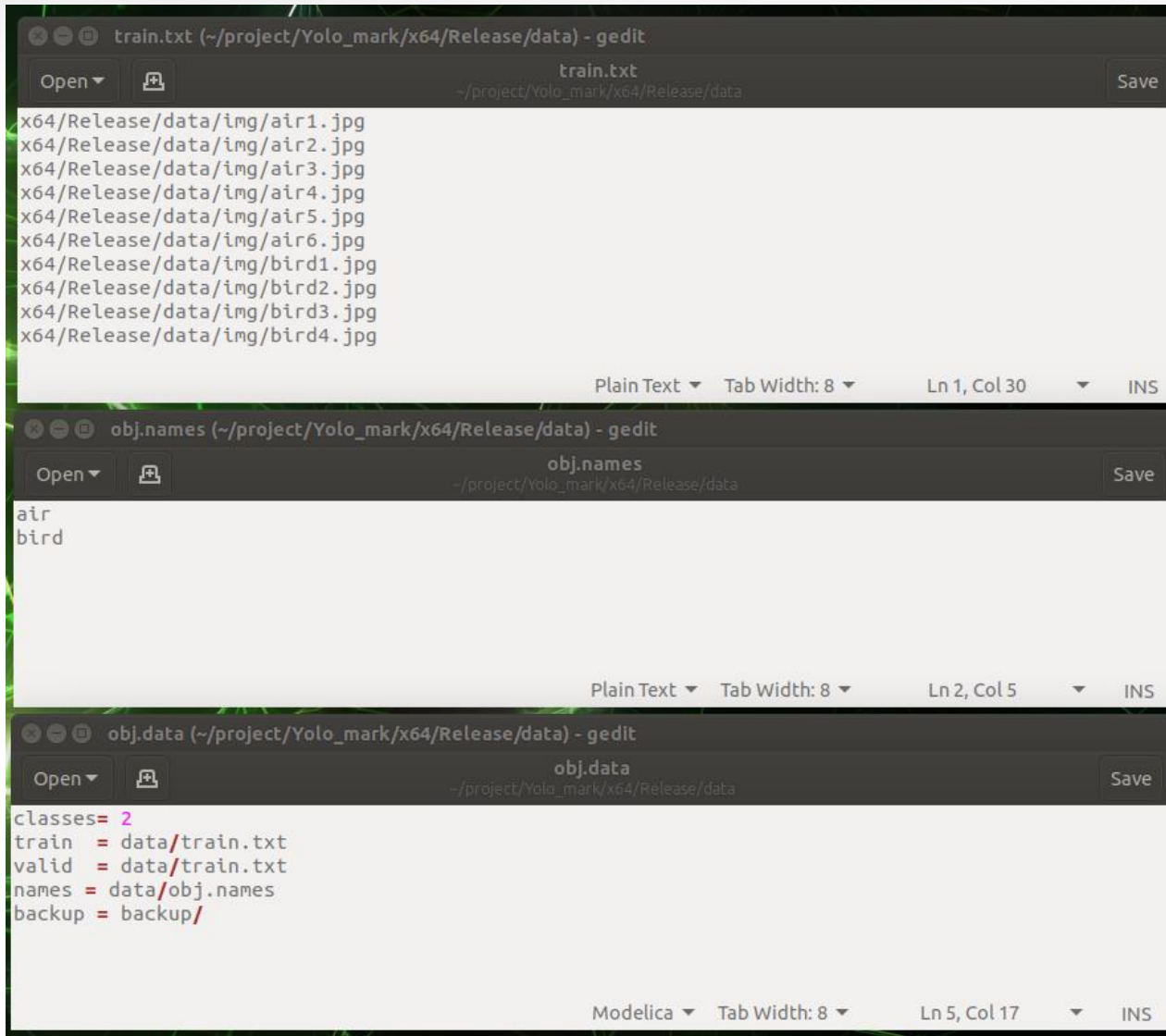
```
$ chmod u+x linux_mark.sh
$ ./linux_mark.sh
```



- object id를 선택하고 해당 물체에 박스 생성

- image를 넘기면서 해당작업 반복

04. YOLOv3를 통한 데이터학습



```
train.txt (~/.project/Yolo_mark/x64/Release/data) - gedit
x64/Release/data/img/air1.jpg
x64/Release/data/img/air2.jpg
x64/Release/data/img/air3.jpg
x64/Release/data/img/air4.jpg
x64/Release/data/img/air5.jpg
x64/Release/data/img/air6.jpg
x64/Release/data/img/bird1.jpg
x64/Release/data/img/bird2.jpg
x64/Release/data/img/bird3.jpg
x64/Release/data/img/bird4.jpg

obj.names (~/.project/Yolo_mark/x64/Release/data) - gedit
air
bird

obj.data (~/.project/Yolo_mark/x64/Release/data) - gedit
classes= 2
train  = data/train.txt
valid  = data/train.txt
names  = data/obj.names
backup = backup/
```

- 트레이닝에 사용될 이미지
파일 경로 설정

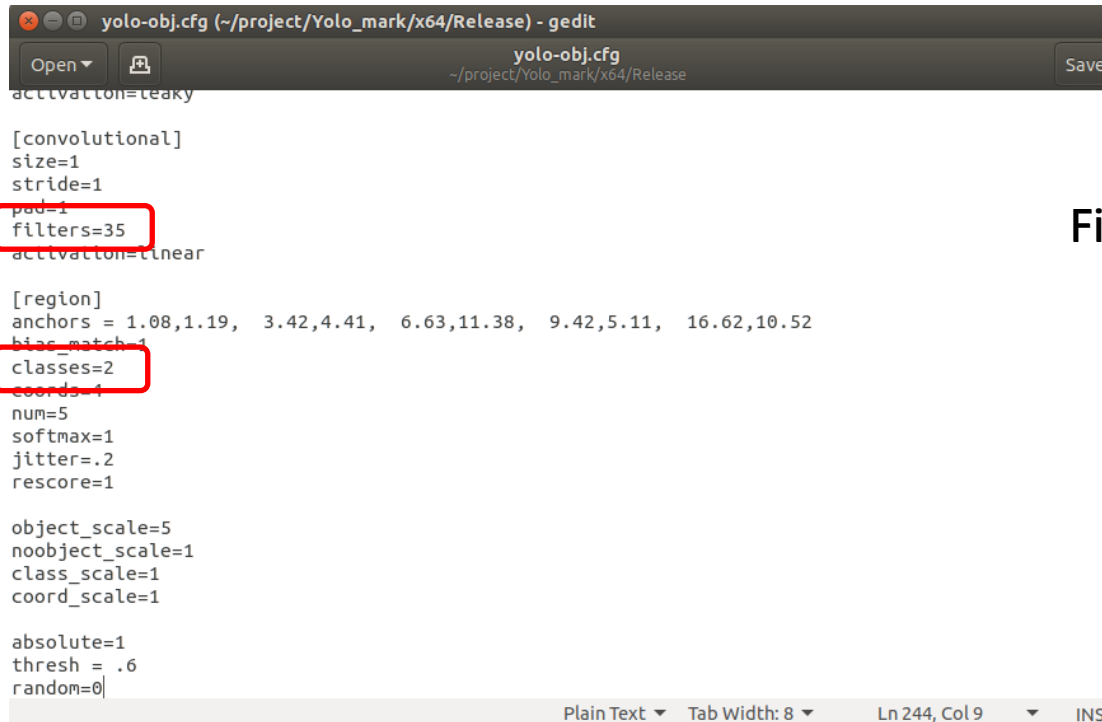
- object name

- class 개수,
Train data, Valid data, object
name, backup 파일 경로 설정

04. YOLOv3를 통한 데이터학습

Class 개수 변경 시

```
$ gedit ~/project/Yolo_mark/x64/Release/yolo-obj.cfg
```



```
yolo-obj.cfg (~/.project/Yolo_mark/x64/Release) - gedit
Open  Save
[convolutional]
size=1
stride=1
pool=1
filters=35
activation=leaky

[region]
anchors = 1.08,1.19, 3.42,4.41, 6.63,11.38, 9.42,5.11, 16.62,10.52
bias_match=1
classes=2
coords=4
num=5
softmax=1
jitter=.2
rescore=1

object_scale=5
noobject_scale=1
class_scale=1
coord_scale=1

absolute=1
thresh = .6
random=0

Plain Text  Tab Width: 8  Ln 244, Col 9  INS
```

Filter = (classes x 5) x 3

Yolo_obj.cfg 파일의 맨 아래쪽의 classes 값과 filter 값을 변경,
나머지 convolutional 은 변경하지 않음

04. YOLOv3를 통한 데이터학습

③ convolutional layer 설치

```
$ cd ~/project/darknet  
$ wget http://pjreddie.com/media/files/darknet19_448.conv.23
```

④ yolo_mark 파일을 darknet 폴더로 이동

- yolo_mark/x64/Release 경로의 yolo-obj.cfg파일을 darknet 디렉토리로 이동
- yolo_mark/x64/Release/data 경로안에 있는 image 디렉토리와 obj.names , obj.data , train.txt를 darknet/data 경로로 이동
- train.txt 내부의 이미지파일경로 변경

⑤ darknet 을 실행하여 학습 진행

```
$ cd ~/project/darknet  
$ ./darknet detector train data/obj.data yolo-obj.cfg darknet19_448.conv.23
```

*Jetson TX2 환경에서는 학습속도가 매우 느리므로 실행하지 않는다.

⑥ 생성된 weights파일을 이용하여 darknet 을 실행(webcam)

```
$ cd ~/project/darknet  
$ ./darknet detector demo data/obj.data yolo-obj.cfg backup/ yolo-obj_800.weights data
```

*참조 : https://github.com/AlexeyAB/Yolo_mark