실전 딥러닝 엔지니어를 위한 위한 필수

--> 모델 서빙 최적화를 위한 프레임워크 선정과 서빙 성능 극대화하기 | 카카오페이 기술 블로그 (kakaopay.com)

https://tech.kakaopay.com/post/model-serving-framework/

--> 파이토치 도커로 서빙까지

https://roytravel.tistory.com/363

NVIDIA TensorRT

아나콘다작업안됨. 강의장 PC 에서 세팅해서 작업하세요.

TensorRT란, NVIDIA사에서 개발한 딥러닝 연산 최적화 엔진입니다. 기존 딥러닝 프레임워크(PyTorch, TensorFlow, Caffe 등)가 네트워크를 구성하고 이를 모델로 만들었다면, TensorRT는 이러한 모델을 **자체적인 여러가지 최적화 과정**을 거쳐 NVIDIA의 GPU, DLA등에 효율적으로 싣는 역할을 합니다. 또한 최적화 과정을 거쳐 만들어낸 '엔진'을 binary 형태로 저장(**blob의 형태로 저장합니다**)하고 사용할때 마다 '엔진'을 deserialize하여 사용합니다.

Torch-TensorRT를 통해 PyTorch에서 추론 속도 최대 6배 향상

https://developer.nvidia.com/tensorrt

https://opac.tistory.com/6

https://opac.tistory.com/5?category=1023584

ONNX

ONNX는 Open Neural Network Exchange의 줄인 말로서 이름과 같이 다른 DNN 프레임워크 환경(ex Tensorflow, PyTorch, etc..)에서 만들어진 모델들을 서로 호환되게 사용할 수 있도록 만들어진 공유 플랫폼이다.
ps. ONNX 또한 DNN 프레임워크라고 부른다.

https://beeny-ds.tistory.com/entry/%EC%86%8C%EA%B0%9C-ONNX-%EB%9E%80

GPU병렬

텐서플로우 분산학습 → https://benban.tistory.com/76

파이토치 분산학습 → https://velog.io/@hsp/Pytorch%EB%A1%9C-Data-%EB%B6%84%EC%82%B0%ED%95%99%EC%8A%B5%ED%95%98%EA%B8%B0

jetson-nano

NDIVIA의 GPU탑재된 마이크로 프로세서, 70 x 45mm에 지나지 않는 Jetson Nano 모듈은 신용 카드보다 더 작은 크기를 자랑합니다. 하지만 프로덕션 준비를 마친 이 SOM(System on Module)은 스마트 시티 및 공장에서 농업 및 로보틱스에 이르기까지 다양한 업계의 엣지에서 디바이스에 AI를 배포할 수 있다는 측면에서 효용성이 매우 큽니다. https://leaderssys.com/sub/embedded_view01.php

https://www.nvidia.com/ko-kr/autonomous-machines/embedded-systems/jetson-nano/product-development/

용로의 기능을 탑재한 마이크로 프로세스, 현재는 단종되어 있으며 , **대체재로 Jetson Orin Nano**가 판매되고 있음. 위의 사이트에서 사용법 확인및 인증 가능
https://whiteknight3672.tistory.com/339

참고: 잭슨나노와 라즈베리파이 비교하기

https://makepluscode.tistory.com/2

당연한 얘기지만, 개발보드 선택은 사용 목적에 맞아야 한다.
높은 GPU 성능이 필요한 컴퓨터비전 또는 인공지능 관련으로 응용을 한다면 엔비디아 젝슨나노 보드가 적합하며, 하드웨어 성능 뿐만 아니라, 배포 이미지에 포함되어 있는 안정된 비전과 인공지능 개발관련 BSP 와 파이선 으로 구현된 샘플 어플리케이션이 유용하게 쓰일것이다.

다만, 무선랜 이나 블루투스 같은 통신 커넥티비티가 없기 때문에 추가로 USB 동글을 사야한다는 점은 아쉽다. 보드사이즈도 큰데 왜 무선랜 모듈을 넣지 않았을까? 반면에 상대적으로 크게 저렴한 라즈베리파이4 의 CPU 성능은 매우 훌륭하며, 4G 메모리로 구입하면 기존 라즈베리파이 3 보다 훨씬 더 쾌적한 컴퓨팅 환경에서 개발이 가능하다.

가격메리트도 있고, 기본적으로 무선LAN과 블루투스 콤보가 있다는 점과 전세계 개발자들의 오픈소스 지원이 확실하기 때문에 매력적이나, 아직 초기 하드웨어에는 몇가지 문제점 들이 이슈가 되고 있다. 참고로 두 보드의 외부 I/O는 매우 비슷하다. 젝슨나노 외부 I/O 설계 시, 라즈베리파이 40핀 헤더 설계를 최대한 참고했기 때문이고, 다만 젝슨나노는 40핀 외. 추가적으로 디버그 포트가 따로있다.

참고: 2023년의 멋진 로봇들: 주요 오토노머스 머신들을 만나보세요

https://blogs.nvidia.co.kr/2023/12/22/robot-roundup-jetson/

Yolov5 추론 내보내기

https://docs.ultralytics.com/ko/yolov5/tutorials/model_export/#exported-model-usage-examples

형식	`export.py --include`	모델
PyTorch	-	`yolov5s.pt`
TorchScript	`torchscript`	`yolov5s.torchscript`
ONNX	`onnx`	`yolov5s.onnx`
OpenVINO	`openvino`	`yolov5s_openvino_model/`
TensorRT	`engine`	`yolov5s.engine`
CoreML	`coreml`	`yolov5s.mlmodel`
TensorFlow SavedModel	`saved_model`	`yolov5s_saved_model/`
TensorFlow GraphDef	`pb`	`yolov5s.pb`
TensorFlow Lite	`tflite`	`yolov5s.tflite`
TensorFlow Edge TPU	`edgetpu`	`yolov5s_edgetpu.tflite`
TensorFlow.js	`tfjs`	`yolov5s_web_model/`
PaddlePaddle	`paddle`	`yolov5s_paddle_model/`

OpenCV가 딥러닝에서 필요한 이유

Opencv + 딥러닝 모델

OpenCV 3.1 버전부터는 딥러닝을 활용할 수 있는 DNN(deep neural network) 모듈을 제공합니다.
OpenCV dnn 모듈은 카페, 텐서플로, 토치 등의 프레임워크에서 학습된 모델과 ONNX(Open Neural Network Exchange) 파일 형식으로 저장된 모델을 불러와 실행할 수 있습니다.

OpenCV DNN 모듈

더북 OpenCV 4로 배우는 컴퓨터 비전과 머신 러닝 (C#으로 되어 있음)
https://thebook.io/006939/0002/

- 검증된 딥러닝 네트워크

[Image classification]
- AlexNet
- GoogLeNet
- VGG
- ResNet
- SqueezeNet
- DenseNet
- ShuffleNet
- Inception
- MobileNet
- Darknet
- Other ONNX formats

[Object detection]
- SSD VGG
- MobileNet-SSD
- Faster-RCNN
- R-FCN
- OpenCV face detector
- SSD, Faster-RCNN and Mask-RCNN
- EAST
- YOLOv2, tiny YOLO, YOLOv3

YOLOV~9

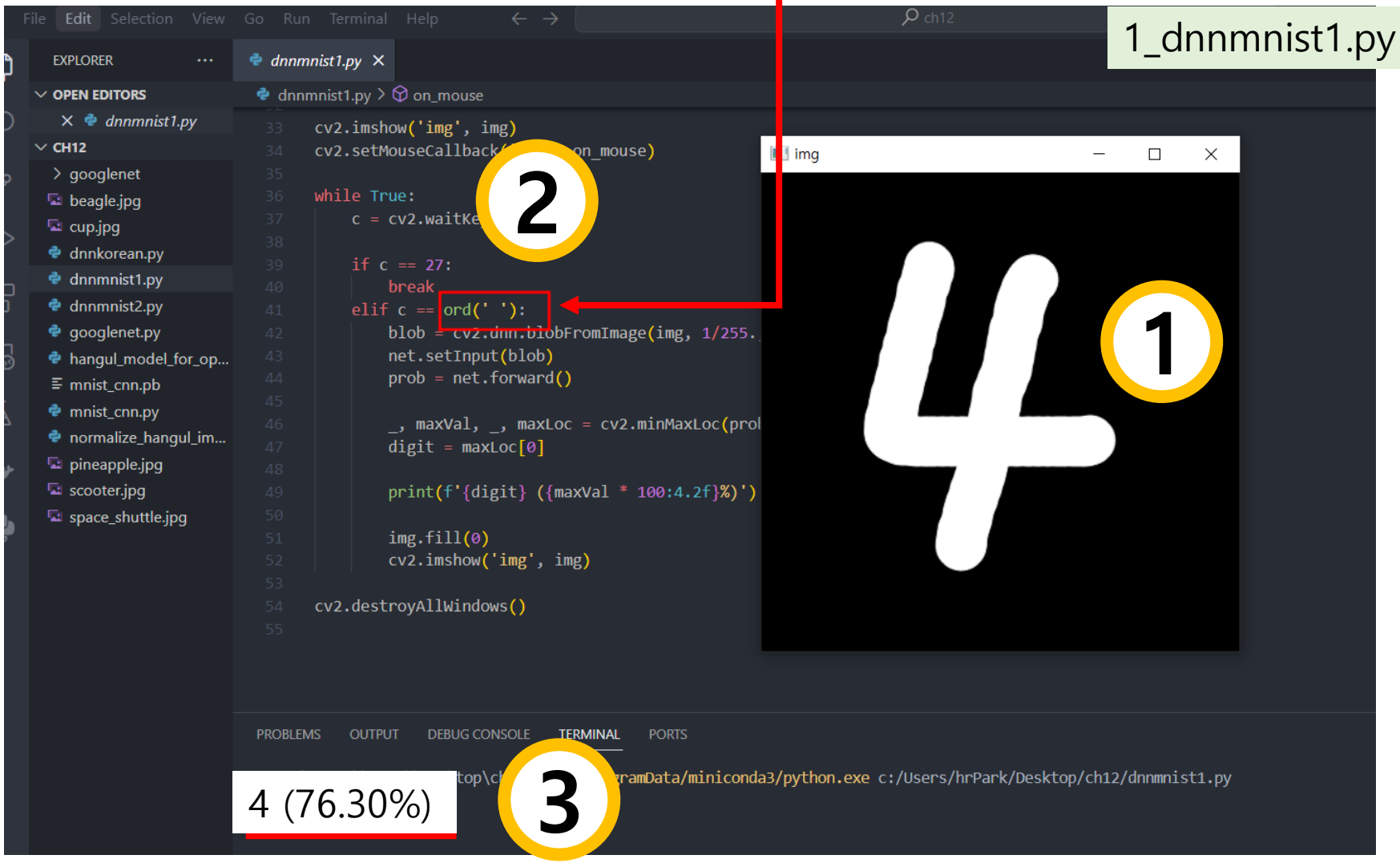
[Semantic segmentation]
- FCN
- ENet

[Pose estimation]
- OpenPose

[Image processing]
- Colorization
- Fast-Neural-Style

[Person identification]
- OpenFace

dnnmnist1.py 를 실행하면 openCV의 창이 실행되고 onMouse 함수가 실행되면서 빈창이 실행됨.
이때 사용자가 (1)창 글자를 입력한뒤 (2)**스페이스바**를 입력하면 화면이 클리어되면서 (3)결과값이 출력됨



코렙에서 실행안됨.
openCV에서 창이 별도로 뜨지 않음

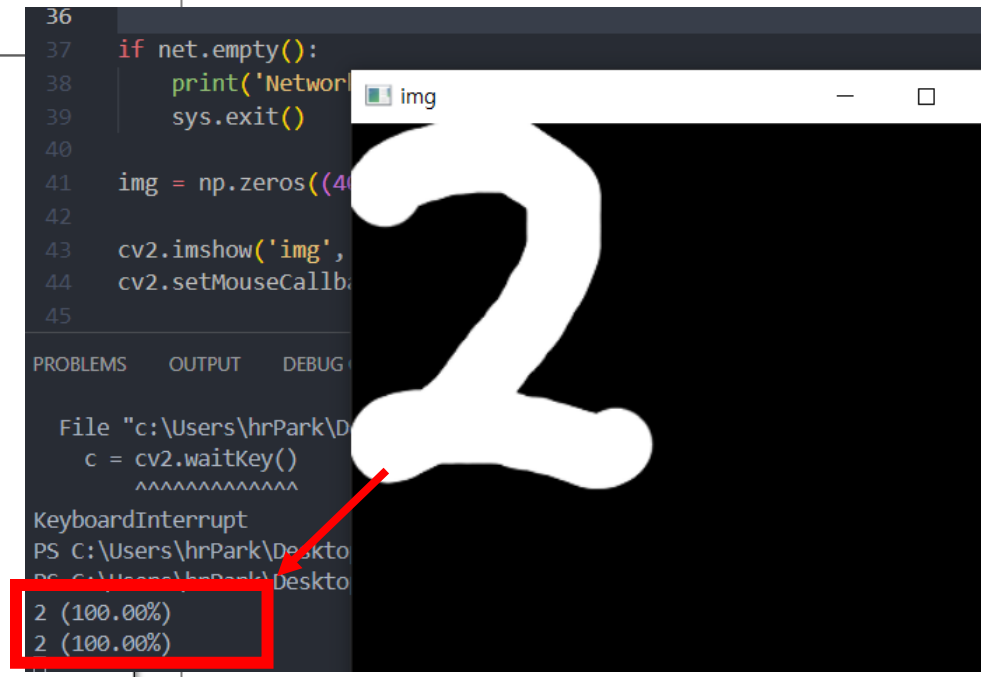
코드참조--> https://deep-learning-study.tistory.com/299?category=946336

***이슈**: 손글씨 mnist는 학습시 정중앙에 글씨를 학습한 자료임으로 사용자가 손글씨 입력을 정중앙에 입력하지 않는다면 아래의 1, 2와 같이 숫자 인식율이 떨어짐

2_dnnmnist2.py

- 위치 정규화 코드 추가

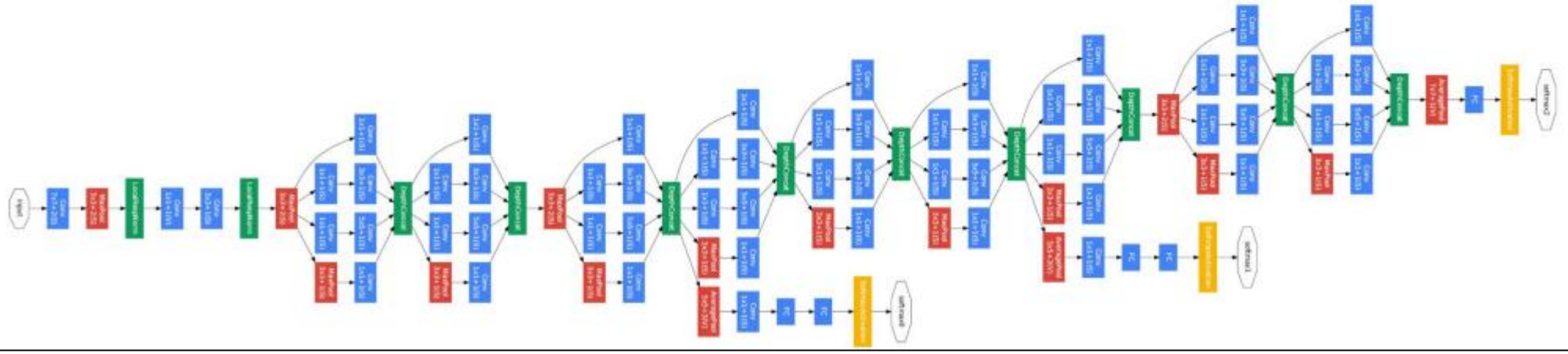
```python
def norm_digit(img):                                  # 무게 중심이 영상 정중앙이 되도록 이동 변환
    m = cv2.moments(img)
    cx = m['m10'] / m['m00']
    cy = m['m01'] / m['m00']
    h, w = img.shape[:2]
    aff = np.array([[1, 0, w/2 - cx], [0, 1, h/2 - cy]], dtype=np.float32)
    dst = cv2.warpAffine(img, aff, (0, 0))
    return dst

...

    if c == 27:
        break
    elif c == ord(' '):
        blob = cv2.dnn.blobFromImage(norm_digit(img), 1/255., (28, 28))
        net.setInput(blob)                            # 정규화 후 추론
        prob = net.forward()
    ...
```



이미 학습된 gooleNet 이미지인식 프로그램을 다운받아서 사용하는 방법임.

GoogLeNet 영상 인식

- GoogLeNet 영상 인식
  - 2014년 ILSVRC(ImageNet Large Scale Visual Recognition Competition) 영상 인식 분야 1위
    - 1000개의 카테고리, 120만개의 훈련 영상, 15만개의 테스트 영상
  - 입력: 224x224, BGR 컬러 영상, 평균 값 = (104, 117, 123)
  - 출력: 1x1000 행렬, 1000개 클래스에 대한 확률 값



이미 학습된 gooleNet 이미지인식 프로그램을 다운받아서 사용하는 방법임.

https://velog.io/@everglow83/GoogLeNet-%EC%98%81%EC%83%81-%EC%9D%B8%EC%8B%9D

3_googlenet.py 필수아님..

C에서 작업하는 구글넷 학습 모델 사용법

https://thebook.io/006939/0703/

OpenCV DNN 얼굴 검출

- OpenCV DNN 얼굴 검출 예제
  - OpenCV 예제에서 DNN 모듈을 사용한 얼굴 검출 기능을 지원
    - SSD(Single Shot MultiBox Detector) 기반 얼굴 검출 네트워크

https://github.com/opencv/opencv/tree/master/samples/dnn/face_detector

- 동시대 다른 객체 검출 알고리즘과 비교하여 성능과 속도 두 가지를 모두 만족시킨 알고리즘
  - Faster R-CNN: 73.2 mAP, 7 FPS
  - YOLOv1: 63.4 mAP, 45 FPS

➔ SSD: 74.3 mAP, 59 FPS

- 기존의 Haar-Cascade 방법보다 속도 & 정확도 면에서 더 좋은 성능을 나타냄

	Haar Cascade	DL
Size on disk	528KB	10MB (fp32), 5MB (fp16)
Efficiency @ 300x300**	30 ms	9.34 ms
Performance AP @ IoU = 0.5*	0.609 (FDDB) 0.149 (WIDER FACE, val.)	0.797 (FDDB) 0.173 (WIDER FACE, val.)

*PASCAL VOC metric using COCO evaluation tool, http://cocodataset.org/#detections-eval
**Intel® Core™ i5-4460 CPU @ 3.20GHz x 4

PB파일의 이해

https://jseobyun.tistory.com/99

Tensorflow framework로 학습한 모델을 C++에서 불러와서 Inference를 하기 위해서는 ckpt 혹은 h5 형식 파일을 pb 형식 파일로 변경을 먼저 해야한다.

pb 파일은 ckpt **파일**과는 달리 모델 구조와 가중치 값이 합쳐진 **파일**로서 재학습이 불가능하다.

tensorflow2에서 pb파일로 저장

https://dongle94.github.io/tensorflow/tf2-obj-detection-api-export/

(참고) 파이토치 모델 자징

https://076923.github.io/posts/Python-pytorch-10/#google_vignette

(참고)
pt(파이토치) -> onnx -> pb -> tfilite(안드로이드)

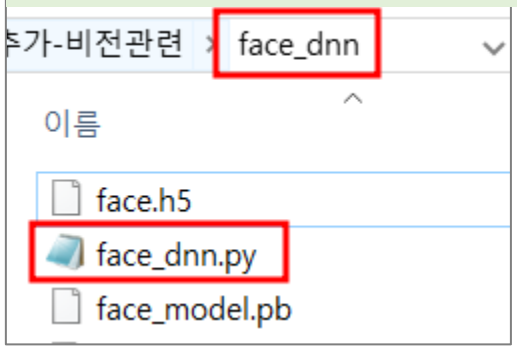
(참고코드) 고개숙임, 정면, 옆면 학습된 face.h5 파일을 .pb 파일로 변환 / 코렙에서 실행

텐서와 openCV에서 사용가능한 pb 파일의 버전 호환성 문제가 있어 2.8로 변환

```
!pip install tensorflow==2.8
```

```
import cv2
import tensorflow as tf

print("OpenCV version:", cv2.__version__)
print("TensorFlow version:", tf.__version__)

# GPT에서 질의 해서 나온 코드임.
# 텐서플로우 2.8 에서 코드 작성
# 1. '/content/face.H5' 파일에서 모델을 로드합니다.
# 2. 모델을 텐서플로우 그래프로 변환합니다.
# 3. 그래프를 '/content/face.pb' 파일로 변환합니다.

import tensorflow as tf

# 모델 로드
model = tf.keras.models.load_model('/content/face.h5')

# 모델을 텐서플로우 그래프로 변환
graph_def = tf.function(lambda x: model(x))
concrete_func = graph_def.get_concrete_function(tf.TensorSpec(model.inputs[0].shape, model.inputs[0].dtype))

# 그래프를 파일에 저장
tf.io.write_graph(concrete_func.graph, '/content', 'face.pb', as_text=False)

print("모델 변환이 완료되었습니다.")
```

fce_dnn 폴더의 face_dnn.py 로 실습하여 봅니다.
코렙에서 실행되지 않음. 웹카메라 안됨



face_dnn.py 를 실행하고
카메라에 얼굴을 정면, 옆면
숙임을 하여서
모델을 실행하고 q를 눌러서 종료후
차트로 확인합니다.

Yolo 객체이상 탐지 사이트 참조

객체 탐지 기법 적용 YCbCr 컬러모델의 화염 영역 검출 특성에 관한 연구

https://www.kifsejournal.or.kr/journal/view.php?number=2221&viewtype=pubreader

욜로 & 미디어파이프 (LSTM) 를 이용한 **절도 이상행동 탐지**

https://velog.io/@seonydg/DL-LSTM-Yolo%EB%A5%BC-%ED%99%9C%EC%9A%A9%ED%95%9C-%EC%A0%88%EB%8F%84-%EC%9D%B4%EC%83%81%ED%96%89%EB%8F%99-%ED%83%90%EC%A7%80

딥러닝 기반 도로위험객체 인식 시스템 성능 향상 방법 개발.pdf