

스마트 모빌리티 프로그래밍

## Ch 18. 심층학습(Deep Learning), 객체 탐지 및 인식, YOLOv5/PyTorch



영남대학교 정보통신공학과  
교수 김 영 탁

(Tel : +82-53-810-2497; E-mail : ytkim@yu.ac.kr)

# Outline

- ◆ 심층학습 (Deep Learning)
- ◆ TensorFlow와 Keras 기반 Deep learning
- ◆ 필기체 숫자 인식
- ◆ YOLOv5/PyTorch/CUDA
- ◆ Jetson Nano 환경에서의 YOLOv5기반 객체 탐지 (Object Detection)



## **심층학습 (Deep Learning)**

# 심층학습 (Deep Learning)

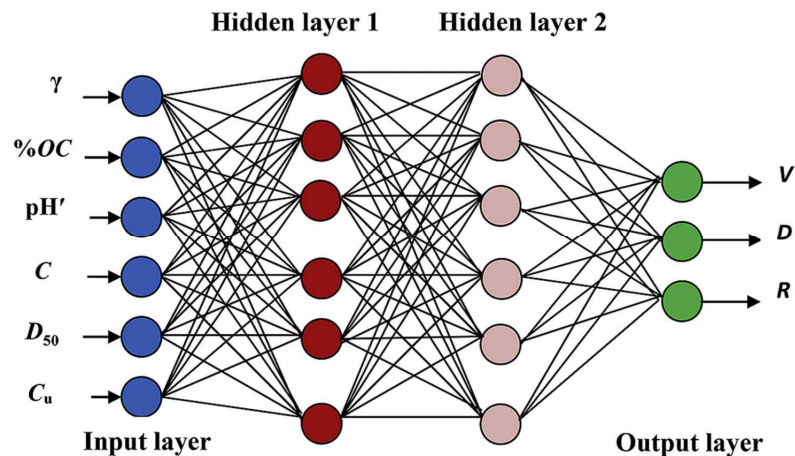
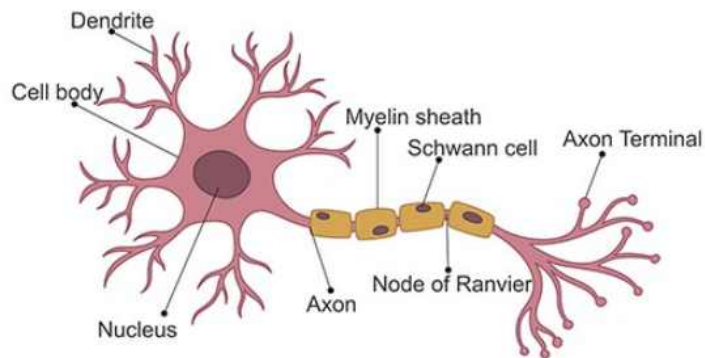
## ◆ 심층학습 (deep learning) 이란?

- 인간의 신경세포 구조를 인공적으로 모델링하여 구현한 인공신경망 (artificial neural network, ANN)을 이용한 기계학습 기법을 의미하며, 다층 구조의 신경망을 사용
- 화상인식, 음성인식, 번역 등의 다양한 분야에서 좋은 성능을 보여줌
- 심층학습 구조를 사용하여 화상인식이나 음성인식을 실시하려면 학습 데이터 (label이 부착된 데이터)를 사용하여 학습/훈련을 실시하여야 함
- 기존 기계학습에서는 학습 데이터의 여러 특징 중에서 어떤 특징을 추출할 지에 대하여 사람이 직접 분석하고 판단하여야 하였으나, 심층학습에서는 기계가 자동으로 학습 데이터에서 특징을 추출하여 자가 학습 (self learning)을 수행하게 됨
- 심층학습에서는 기계가 자동으로 대규모 데이터에서 자가 학습을 수행하여 중요한 패턴과 규칙을 파악하고, 이를 의사결정과 예측 등에 적용 함

# 신경망(neural network)과 딥러닝 (deep learning)

## ◆ 딥러닝 (deep learning)

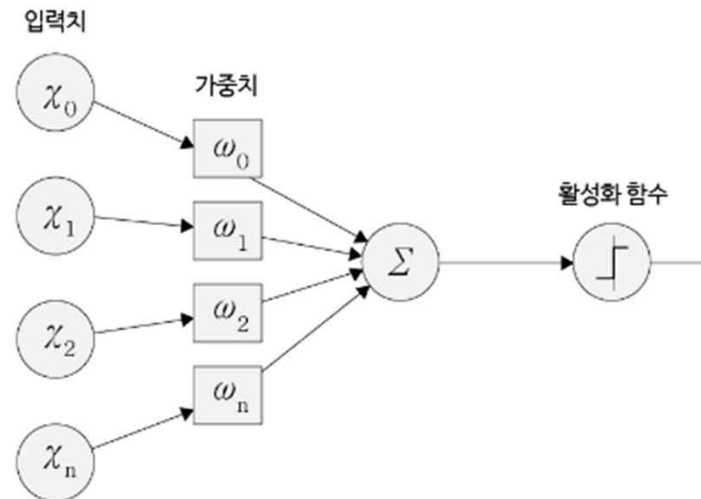
- 신경망 (neural network)을 사용하는 기계학습의 한 분야  
(참고: [https://ko.wikipedia.org/wiki/인공\\_신경망](https://ko.wikipedia.org/wiki/인공_신경망))
- 인간의 신경세포 (neuron)와 인공 지능망 (artificial neural network)
- Multi-Layer Perceptron (MLP)



# 퍼셉트론 (Perceptron)

## ◆ Perceptron (퍼셉트론)

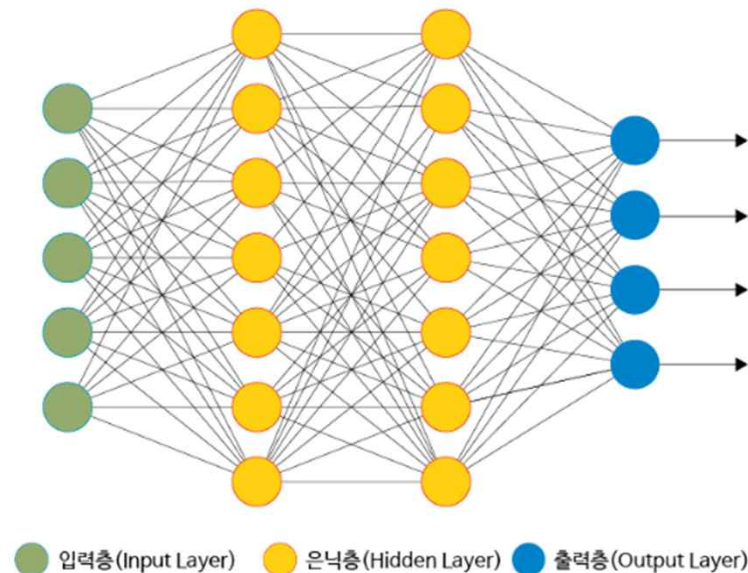
- 퍼셉트론(perceptron)이란 1957년 미국의 심리학자 프랑크 로젠블라트(Frank Rosenblatt)에 의해 고안된 인공신경망 이론을 설명한 최초의 알고리즘. 로젠블라트는 가장 간단한 퍼셉트론으로 입력층과 출력층만으로 구성된 단층 퍼셉트론(single layer perceptron)의 개념을 제안하였음
- 다수의 입력 데이터로부터 하나의 결과를 출력하도록 하는 복합논리회로
- 단층 퍼셉트론에서 가중치와 임계치를 적절히 변경하면, 상황에 맞는 적절한 의사결정을 내릴 수 있게 됨



# 다층 퍼셉트론 (Multi-layer Perceptron, MLP)

## ◆ Multi-layer Perceptron (MLP)

- 단층 퍼셉트론을 여러 개 조합하면 더욱 복잡한 문제도 판단할 수 있게 되며, 이를 다층 퍼셉트론(MultiLayer Perceptron, MLP)이라고 부릅니다.
- 다층 퍼셉트론은 단층 퍼셉트론을 사용해서는 풀지 못하는 비선형 문제까지도 풀 수 있음



# 합성곱 신경망 (Convolutional Neural Network, CNN)

## ◆ CNN (Convolutional Neural Network)

- 음성과 이미지 데이터 처럼 입력 데이터의 양이 많을 때 입력데이터를 압축하여 모델링하는 인공신경망이며, 필요한 수준으로 압축하여 성능과 속도를 모두 확보할 수 있음



# Generative Adversarial Network (GAN)

## ◆ 생산적 적대 신경망

- 특정 데이터의 진위를 구별하는 판별모델과 가짜 데이터를 생성하는 위조 모델을 두고 경쟁시켜 두 모델 모두 학습시켜 발전시킴

# Recurrent Neural Network (RNN)

## ◆ 순환 신경망 (RNN)

- 은닉층의 출력을 다음 단계에서 입력 데이터와 함께 다시 입력하여 사용
- 과거 입력 데이터를 요약하여 이후 처리에도 반영하여 사용하기 때문에 입력데이터 간의 상호 관계를 파악할 수 있음

# Deep Learning의 응용 – 영상 인식, 자동 번역

## ◆ 영상 인식

- 얼굴인식, 도로 표지판 인식
- 다양한 개체 인식: 사람, 동물, 개, 고양이
- 주로 OpenCV 패키지를 함께 사용함

## ◆ 자동 번역 및 통역

- 인터넷 웹 문서의 자동 번역, 문맥 (context) 분석
- 자동 통역

# TensorFlow, Keras

## ◆ TensorFlow

- 구글내 연구와 제품개발을 위한 목적으로 구글 브레인팀이 개발
- 2015년 11월 9일 [아파치](#) 2.0 [오픈소스 라이선스](#)로 공개
- <https://www.tensorflow.org/>
- tensor: 물리학에서 다차원 배열을 의미 (tensor에 차원을 지정하면 스칼라, 벡터, 행렬, 텐서를 모두 지원할 수 있음)
- flow: data flow

## ◆ Keras

- 파이썬으로 구현된 딥러닝 라이브러리이며, TensorFlow 2.0 이후 버전에서 딥러닝 모델을 쉽게 구성할 수 있게 함



# **TensorFlow/Keras 기반 심층학습 (deep learning)**

# Tesorflow와 Keras 설치

## ◆ Tesorflow 설치

- 참고: <https://www.tensorflow.org/install/pip#windows>
- Python 3.9 이전 버전에 설치 (2022년 1월 현재, 3.10 버전에서는 오류 발생)
  - > python -m pip install --user --upgrade tensorflow

```
C:\Users\Owner>python -V
Python 3.9.9

C:\Users\Owner>python -m pip install --user --upgrade tensorflow
Collecting tensorflow
  Downloading tensorflow-2.7.0-cp39-cp39-win_amd64.whl (430.8 MB)
    | 430.8 MB 656 kB/s
Collecting wheel<1.0,>=0.32.0
  Downloading wheel-0.37.1-py2.py3-none-any.whl (35 kB)
Collecting absl-py>=0.4.0
  Downloading absl_py-1.0.0-py3-none-any.whl (126 kB)
    | 126 kB ...
Requirement already satisfied: six>=1.12.0 in c:\Users\Owner\AppData\Local\Programs\Python\Python39\lib\site-packages (from tensorflow) (1.16.0)
Collecting tensorboard~2.6
```

...

```
Successfully installed absl-py-1.0.0 astunparse-1.6.3 cachetools-5.0.0 certifi-2021.10.8 charset-normalizer-2.0.10 flatbuffers-2.0 gast-0.4.0 google-auth-2.5.0 google-auth-oauthlib-0.4.6 google-pasta-0.2.0 grpcio-1.43.0 h5py-3.6.0 idna-3.3 importlib-metadata-4.10.1 keras-2.7.0 keras-preprocessing-1.1.2 libclang-12.0.0 markdown-3.3.6 oauthlib-3.1.1 opt-einsum-3.3.0 protobuf-3.19.3 pyasn1-0.4.8 pyasn1-modules-0.2.8 requests-2.27.1 requests-oauthlib-1.3.0 rsa-4.8 tensorboard-2.8.0 tensorboard-data-server-0.6.1 tensorboard-plugin-wit-1.8.1 tensorflow-2.7.0 tensorflow-estimator-2.7.0 tensorflow-io-gcs-filesystem-0.23.1 termcolor-1.1.0 urllib3-1.26.8 werkzeug-2.0.2 wheel-0.37.1 wrapt-1.13.3 zipp-3.7.0
```



# NVIDIA 그래픽 드라이버

## ◆ NVIDIA 그래픽 드라이버 설치 확인

>nvidia-smi

```
C:\Users\Owner>nvidia-smi
Thu Jan 13 20:38:26 2022
```

NVIDIA-SMI 496.49		Driver Version: 496.49		CUDA Version: 11.5	
GPU	Name	TCC/WDDM	Bus-Id	Disp.A	Volatile Uncorr. ECC
Fan	Temp	Perf Pwr:Usage/Cap		Memory-Usage	GPU-Util Compute M. MIG M.
0	NVIDIA GeForce ...	WDDM	00000000:01:00.0 On		N/A
0%	42C	P8 22W / 200W	1243MiB / 8192MiB	6%	Default N/A

NVIDIA 및 CUDA version 확인

# Keras와 Keras-nightly

## ◆ Keras-nightly 문제

- tensorflow 2.0 버전부터 Keras를 내장하여 배포하기 시작
- tensorflow 2.5버전을 설치하면 Keras-nightly 패키지가 함께 설치되지만 사용되지는 않음
- 2.5버전에서 해당 패키지와 충돌로 인해 Keras를 import를 하면 오류가 발생하는 경우가 있음

## ◆ 해결 방법

- cmd 창에서 `python -m pip uninstall keras-nightly` 입력
- 제거하여도 동작하지 않는다면 아래 명령어 입력하여 텐서플로를 재설치
  - `python -m pip install tensorflow --upgrade --force-reinstall`



# Keras Sequential Model 관련 함수

## ◆ Keras Sequential Model



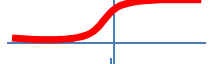

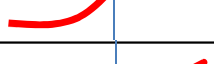

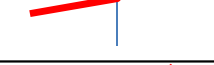
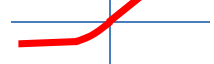

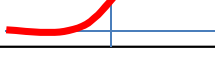
- [https://www.tensorflow.org/guide/keras/sequential\\_model?hl=ko](https://www.tensorflow.org/guide/keras/sequential_model?hl=ko)
- Sequential model: 각 레이어에 정확히 하나의 입력 텐서와 하나의 출력 텐서가 있는 일반 레이어 스택에 적합

## ◆ Model에 추가될 수 있는 layer

- <https://zereight.tistory.com/227>

Layer 종류	설명
Flatten	2차원의 특징 맵을 전결합층으로 전달하기 위하여 1차원 형식으로 변환
Dense	모든 입력 뉴런과 출력 뉴런을 연결하는 전 결합 층
Conv2D	필터를 사용하여 영상 특징을 추출하는 Convolution2D
MaxPooling2D	입력벡터에서 특정 구간마다 값을 골라 벡터를 구성한 후 반환
Dropout	과적합을 방지하기 위하여 학습시에 지정된 비율만큼 임의의 노드(뉴런)들을 제외시킴

# Activation Functions in Neural Networks

activation	equation	plot
identity	$f(x) = x$	
binary step	$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	
logistic (soft step)	$f(x) = \frac{1}{1 + e^{-x}}$	
tanh	$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	
arctan	$f(x) = \tan^{-1}(x)$	
rectified linear unit (relu)	$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	
parametric rectified linear unit (prelu)	$f(x) = \begin{cases} ax & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	
exponential linear unit (elu)	$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	
softplus	$f(x) = \ln_e(1 + e^x)$	
logistic sigmoid	$\phi(x) = \frac{1}{(1 + e^{-x})}$	

# Model related functions compile(), fit(), evaluate()

## ◆ CNN Model related methods

- Convolutional Neural Network (CNN) - 합성곱 신경망
- <https://keras.io/api/metrics/>

method	description
compile()	model의 optimizer, loss, metrics를 선택 - optimizer: sgd, rmsprop, adam, adadelta, adagrad, adamax, nadam, ftrl - loss: probabilistic, regression, hinge - metrics: accuracy, probabilistic, regression, classification, image segmentation, hinge
fit()	모델의 weight와 bias 값을 학습을 통해 결정
evaluate()	test data set에 대하여 성능을 평가
summary()	model의 구조와 관련 파라미터를 출력
save()	model을 파일로 출력
load_model()	지정된 파일로 부터 model을 읽고 설치
predict()	model을 사용하여 주어진 패턴에 대하여 예측

# Keras 기반 선형회귀예제

```
# Keras application - linear regression

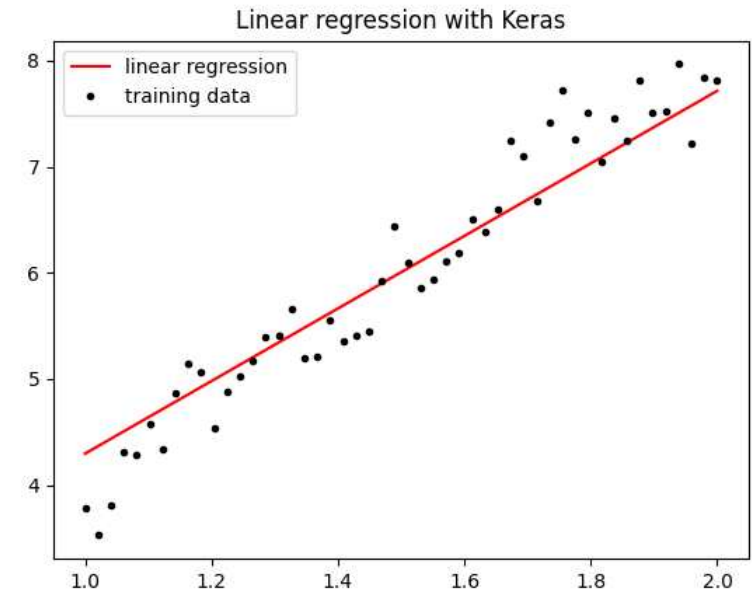
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

# generate data set
X = data = np.linspace(1, 2, 50)
y = X*4 + np.random.randn(50) * 0.3 # add noise

model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Dense(1, input_dim=1,
    activation='linear'))

model.compile(optimizer='sgd', loss='mse', metrics=['mse'])
model.fit(X, y, batch_size=1, epochs=20, verbose=2)

predict = model.predict(data)
print("model.summary() = ", model.summary() )
plt.plot(data, predict, 'r', label="linear regression")
plt.plot(data, y, 'k.', label="training data")
    # blue predict line, black dots of random data
plt.title("Linear regression with Keras")
plt.legend(loc="best")
plt.show()
```



# Keras 기반 다중 회귀 (Polynomial Regression)

```
# tensorflow, polynomial regression (1)
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Dense, Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from sklearn.preprocessing import PolynomialFeatures

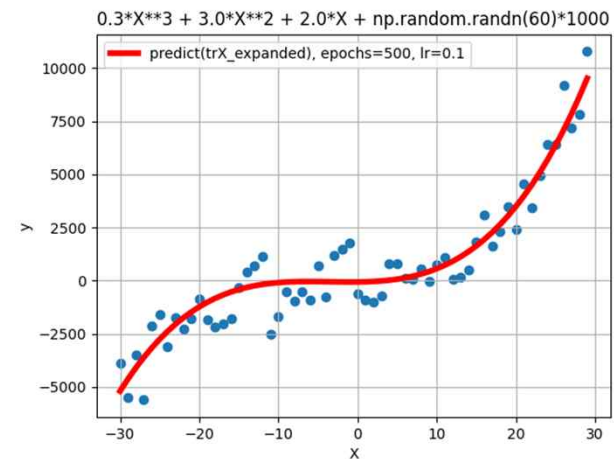
X = np.arange(-30, 30, 1)
y = 0.3*X**3 + 3.0*X**2 + 2.0*X + np.random.randn(60)*1000

#trX, trY = X/max(X), y/max(y)
trX, trY = X, y
n = 3
trX_expanded = np.expand_dims(trX, axis=1)
poly = PolynomialFeatures(n)
# returns: [1, x, x^2, x^3]
trX_expanded = poly.fit_transform(trX_expanded)

inp = Input((n+1))
#since one of the features is 1, we need an extra input
out = Dense(1)(inp)
model = Model(inputs=inp, outputs=out)
model.compile(optimizer=Adam(learning_rate=1e-1),
              loss="mean_squared_error")
model.fit(trX_expanded, trY, epochs=500)
```

```
# tensorflow, polynomial regression (2)
```

```
#plot the data
fig, ax = plt.subplots()
ax.scatter(trX, trY)
ax.plot(trX, model.predict(trX_expanded), linewidth=4, color="red",
        label="predict(trX_expanded), epochs=500, lr=0.1")
plt.xlabel("X")
plt.ylabel("y")
plt.title("0.3*X**3 + 3.0*X**2 + 2.0*X + np.random.randn(60)*1000")
plt.grid(True)
plt.legend(loc="best")
plt.show()
```



스마트 모빌리티 프로그래밍  
교수 김 영 탁

## Keras 기반 필기체 인식

### ◆ MNIST 데이터 셋 (<http://yann.lecun.com/exdb/mnist/>)

- 70,000개의 데이터 셋 (28x28 크기)
- 모델 학습을 위한 학습용 데이터 55,000개 : mnist.train
- 학습된 모델을 시험하기 위한 test data set 10,000개: mnist.test
- 모델을 검증하기 위한 검증 용 데이터 5,000개 : mnist.validation
- training : validation : test = 55,000 : 5,000 : 10,000

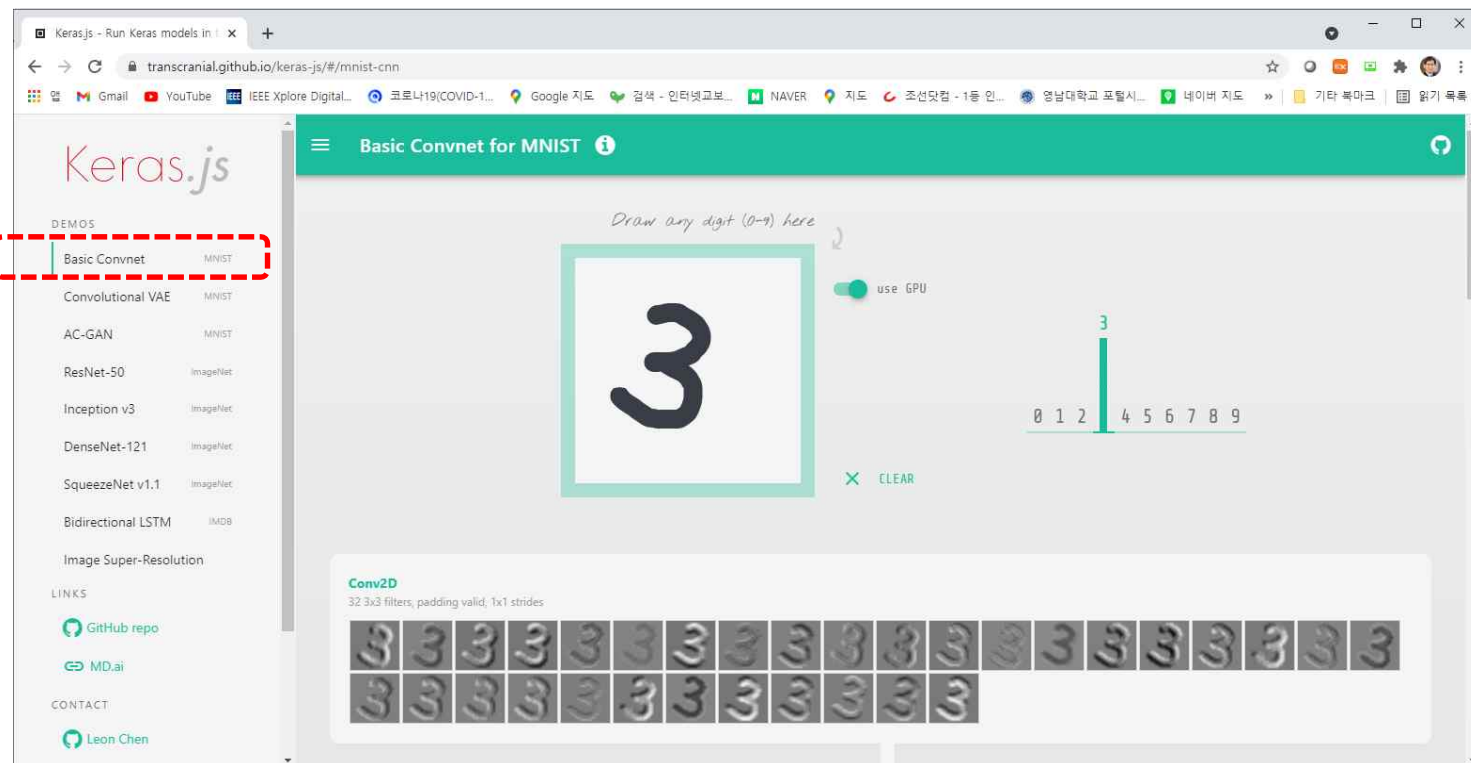
파일	목적
<a href="#">train-images-idx3-ubyte.gz</a>	학습 셋 이미지 - 55000개의 트레이닝 이미지, 5000개의 검증 이미지
<a href="#">train-labels-idx1-ubyte.gz</a>	이미지와 매칭되는 학습 셋 레이블
<a href="#">t10k-images-idx3-ubyte.gz</a>	테스트 셋 이미지 - 10000개의 이미지
<a href="#">t10k-labels-idx1-ubyte.gz</a>	이미지와 매칭되는 테스트 셋 레이블



# Keras 실습

## ◆ Keras 실습 - 필기체 인식

- <https://transcranial.github.io/keras-js/#/mnist-cnn>



# 필기체 숫자 인식

## ◆ CNN 구조의 필기체 숫자 인식 모델 구성

- Import the libraries and load the MNIST dataset
- Data Preprocess and Normalization
- Create the model
- Train the model
- Evaluate the model
- Save the model in file (CNN\_model\_digits)

## ◆ CNN 구조의 필기체 숫자 인식 GUI App 실행

- Load the model from file (CNN\_model\_digits)
- Create GUI to predict digits



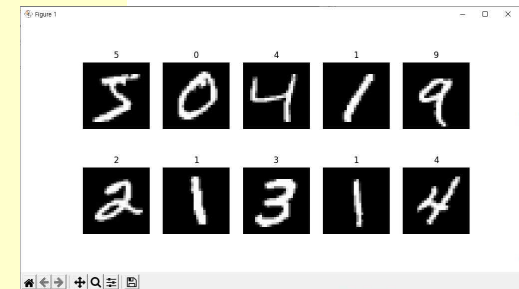
## # Handwritten Digits Recognition (1)

```
import tensorflow as tf
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
#from keras.utils import to_categorical
kr_utils = tf.keras.utils

from keras import backend as k
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

#load dataset directly from keras library
print("Loading MNIST data . . . .")
mnist_npz_path = 'C://MyPyLib//MNIST//mnist.npz'
(X_train, y_train), (X_test, y_test)\
    = mnist.load_data(path = mnist_npz_path)

digit_names = ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9"]
plt.figure(figsize=(10,5))
for i in range(10):
    plt.subplot(2, 5, i+1)
    plt.imshow(X_train[i], cmap="gray")
    plt.title(digit_names[y_train[i]])
    plt.axis('off')
plt.show()
```



```

# Handwritten Digits Recognition (2)

# reshape format [samples][width][height][channels]
print("Reshaping format . . . .")
X_train = X_train.reshape(X_train.shape[0], 28, 28, 1).astype('float32')
X_test = X_test.reshape(X_test.shape[0], 28, 28, 1).astype('float32')

# Converts a class vector (integer) to binary class matrix
print("Converting class vector . . . .")
# Converts a class vector (integers) to binary class matrix.
y_train = tf.keras.utils.to_categorical(y_train)
y_test = tf.keras.utils.to_categorical(y_test)

# normalize inputs
X_train = X_train / 255
X_test = X_test / 255

print("Preparing a CNN model . . . .")
# define a CNN model
num_classes = 10
model = Sequential([
    Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    Flatten(),
    Dense(256, activation='relu'),
    Dropout(0.5),
    Dense(num_classes, activation='softmax')])
model.compile(loss='categorical_crossentropy', optimizer='adam', \
    metrics=['accuracy'])

```



### # Handwritten Digits Recognition (3)

```
print("Fitting the model . . . .")
# fit the model
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=20,
batch_size=200, verbose=2)
print("The model has successfully trained")
```

```
# Save the model
model.save("CNN_model_Digits")
print("The model has successfully saved !!")
model.summary() # print model
```

```
# Evaluate the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("CNN error: %.2f%%"%(100 - scores[1]*100))
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_1 (Conv2D)	(None, 11, 11, 64)	18496
flatten (Flatten)	(None, 7744)	0
dense (Dense)	(None, 256)	1982720
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 10)	2570

Total params: 2,004,106  
Trainable params: 2,004,106  
Non-trainable params: 0

The model has successfully saved !!

CNN error: 0.80%

```
Loading MNIST data . . . .
Reshaping format . . . .
Converting class vector . . . .
Preparing a CNN model . . . .
Compiling the model . . . .
Fitting the model . . . .
Epoch 1/20
WARNING:tensorflow:AutoGraph could not transform <bound method Dense.call of <keras.layers.cc
2E50>> and will run it as-is.
Please report this to the TensorFlow team. When filing the bug, set the verbosity to 10 (on l
=10') and attach the full output.
Cause: invalid syntax (tmpaddsv49f.py, line 48)
To silence this warning, decorate the function with @tf.autograph.experimental.do_not_convert
300/300 - 41s - loss: 0.2205 - accuracy: 0.9333 - val_loss: 0.0488 - val_accuracy: 0.9840
Epoch 2/20
300/300 - 30s - loss: 0.0664 - accuracy: 0.9800 - val_loss: 0.0370 - val_accuracy: 0.9866
Epoch 3/20
300/300 - 31s - loss: 0.0475 - accuracy: 0.9855 - val_loss: 0.0298 - val_accuracy: 0.9896
Epoch 4/20
300/300 - 30s - loss: 0.0366 - accuracy: 0.9886 - val_loss: 0.0297 - val_accuracy: 0.9895
Epoch 5/20
300/300 - 30s - loss: 0.0306 - accuracy: 0.9902 - val_loss: 0.0268 - val_accuracy: 0.9913
Epoch 6/20
300/300 - 30s - loss: 0.0251 - accuracy: 0.9916 - val_loss: 0.0270 - val_accuracy: 0.9908
Epoch 7/20
300/300 - 31s - loss: 0.0210 - accuracy: 0.9931 - val_loss: 0.0265 - val_accuracy: 0.9911
Epoch 8/20
300/300 - 31s - loss: 0.0184 - accuracy: 0.9940 - val_loss: 0.0256 - val_accuracy: 0.9924
Epoch 9/20
300/300 - 29s - loss: 0.0150 - accuracy: 0.9951 - val_loss: 0.0275 - val_accuracy: 0.9917
Epoch 10/20
300/300 - 30s - loss: 0.0149 - accuracy: 0.9951 - val_loss: 0.0266 - val_accuracy: 0.9916
Epoch 11/20
300/300 - 30s - loss: 0.0135 - accuracy: 0.9953 - val_loss: 0.0320 - val_accuracy: 0.9896
Epoch 12/20
300/300 - 30s - loss: 0.0130 - accuracy: 0.9957 - val_loss: 0.0258 - val_accuracy: 0.9928
Epoch 13/20
300/300 - 30s - loss: 0.0101 - accuracy: 0.9964 - val_loss: 0.0276 - val_accuracy: 0.9922
Epoch 14/20
300/300 - 31s - loss: 0.0126 - accuracy: 0.9955 - val_loss: 0.0249 - val_accuracy: 0.9926
Epoch 15/20
300/300 - 31s - loss: 0.0082 - accuracy: 0.9972 - val_loss: 0.0275 - val_accuracy: 0.9931
Epoch 16/20
300/300 - 30s - loss: 0.0092 - accuracy: 0.9967 - val_loss: 0.0292 - val_accuracy: 0.9924
Epoch 17/20
300/300 - 30s - loss: 0.0077 - accuracy: 0.9974 - val_loss: 0.0282 - val_accuracy: 0.9926
Epoch 18/20
300/300 - 29s - loss: 0.0073 - accuracy: 0.9975 - val_loss: 0.0410 - val_accuracy: 0.9900
Epoch 19/20
300/300 - 29s - loss: 0.0075 - accuracy: 0.9974 - val_loss: 0.0290 - val_accuracy: 0.9920
Epoch 20/20
300/300 - 30s - loss: 0.0051 - accuracy: 0.9980 - val_loss: 0.0323 - val_accuracy: 0.9921
The model has successfully trained
```



```

# GUI for handwritten digits recognition (1)

import os
import PIL
import cv2
import glob
import numpy as np
from tkinter import *
from PIL import Image, ImageDraw, ImageGrab

# load model
from keras.models import load_model
model = load_model("CNN_model_Digits")
model.summary()
print("Model is loaded successfully ...")

# create a main window first (named as win)
win = Tk()
win.resizable(0, 0)
win.title("Handwritten Digits Recognition GUI App")

# Initialize variables
lastx, lasty = None, None
image_number = 0

# Create a canvas for drawing digits
cnv = Canvas(win, width=640, height=480, bg='white')
cnv.grid(row=0, column=0, pady=2, sticky=W, columnspan=2)

```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_1 (Conv2D)	(None, 11, 11, 64)	18496
flatten (Flatten)	(None, 7744)	0
dense (Dense)	(None, 256)	1982720
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 10)	2570

Total params: 2,004,106

Trainable params: 2,004,106

Non-trainable params: 0

The model has successfully saved !!

CNN error: 0.80%



```
# GUI for handwritten digits recognition (2)
```

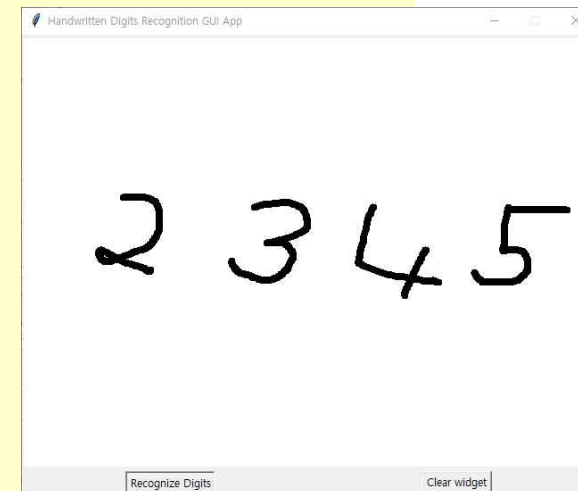
```
def draw_lines(event):  
    global lastx, lasty  
    x, y = event.x, event.y  
    cnv.create_line((lastx, lasty, x, y), width=8, fill='black',\  
                    capstyle=ROUND, smooth=TRUE, splinesteps=12)  
    lastx, lasty = x, y
```

```
def clear_widget():  
    global cnv  
    cnv.delete("all")
```

```
def activate_event(event):  
    global lastx, lasty  
    cnv.bind('<B1-Motion>', draw_lines)  
    lastx, lasty = event.x, event.y
```

```
def recognize_digit():  
    global image_number  
    predictions = []  
    percentage = []  
    #image_number = 0  
    filename = f'image_{image_number}.png'  
    widget = cnv
```

```
#get the cnv coordinates  
x = win.winfo_winx() + cnv.winfo_x()  
y = win.winfo_winy() + cnv.winfo_y()  
x1 = x + cnv.winfo_width()  
y1 = y + cnv.winfo_height()
```

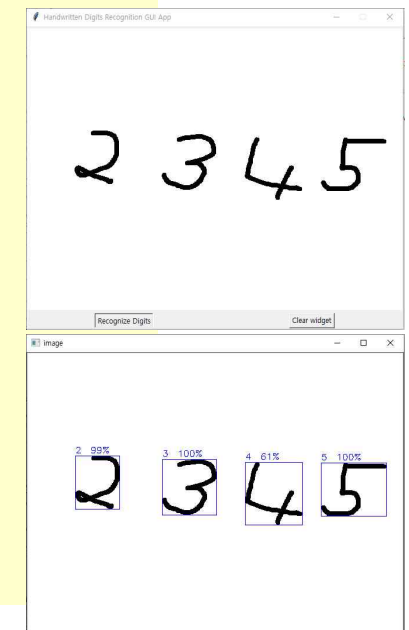


### # GUI for handwritten digits recognition (3)

```
# grab the image, crop it
ImageGrab.grab().crop((x, y, x1, y1)).save(filename)

# read the image in color format
image = cv2.imread(filename, cv2.IMREAD_COLOR)
# convert the image in grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
# applying thresholding (Bobuyuki Otsu's method: greyscale to monochrome)
ret, th = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)
# findContours() function helps in extracting the contours from the image
contours = cv2.findContours(th, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[0]

for cnt in contours:
    # get bounding box and extract ROI
    x, y, w, h = cv2.boundingRect(cnt)
    # create rectangle
    cv2.rectangle(image, (x, y), (x+w, y+h), (255, 0, 0), 1)
    top = int(0.05 * th.shape[0])
    bottom = top
    left = int(0.05 * th.shape[1])
    right = left
    th_up = cv2.copyMakeBorder(th, top, bottom, left, right, \
        cv2.BORDER_REPLICATE)
    # extract the image ROI
    roi = th_up[y-top:y+h+bottom, x-left:x+w+right]
    # resize roi image to 28x28 pixels
    img = cv2.resize(roi, (28, 28), interpolation=cv2.INTER_AREA)
    # reshaping the image to support our model input
    img = img.reshape(1, 28, 28, 1)
```



```

# GUI for handwritten digits recognition (4)

    # normalizing the image
    img = img / 255.0
    pred = model.predict([img])[0]
    final_pred = np.argmax(pred)
    data = str(final_pred) + ' ' + str(int(max(pred)*100)) + '%'
    font = cv2.FONT_HERSHEY_SIMPLEX
    fontScale = 0.5
    color = (255, 0, 0)
    thickness = 1
    cv2.putText(image, data, (x, y-5), font, fontScale, color, thickness)

cv2.imshow("image", image)
cv2.waitKey(0)

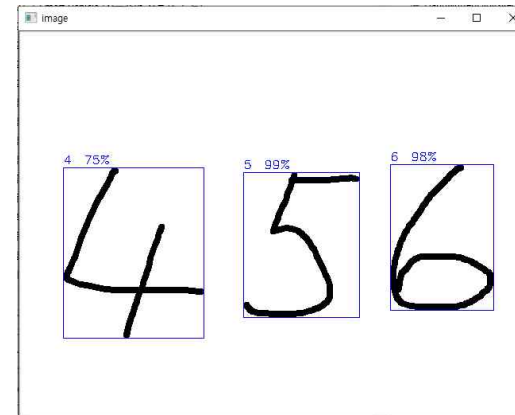
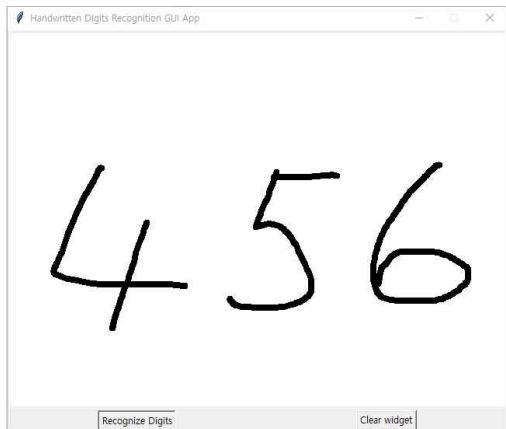
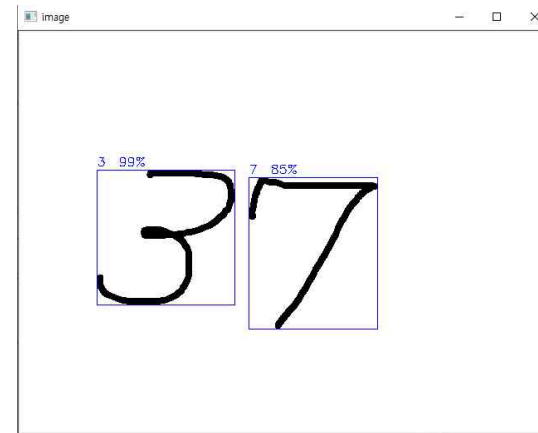
# Tkinter
cnv.bind('<Button-1>', activate_event)

# Add buttons and labels
btn_recog = Button(text="Recognize Digits", command = recognize_digit)
btn_recog.grid(row=2, column=0, padx=1, pady=1)
btn_clear = Button(text="Clear widget", command = clear_widget)
btn_clear.grid(row=2, column=1, padx=1, pady=1)

# mainloop()
win.mainloop()

```

# Results of Recognitions





# TensorFlow/CUDA 드라이버 관련 발생 가능 문제

## ◆ TensorFlow/CUDA 드라이버 관련 문제

5번 필기체인식 과제 중 tensorflow 프로그램을 실행시키면 이런 에러가 뜨는데 무엇이 문제인가요??

```
2021-06-07 16:02:05.836240: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlopen: cudart64_110.dll not found
2021-06-07 16:02:13.331979: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlopen error if you do not have a GPU set up on your machine.
```

## ◆ 문제 원인

- NVIDIA 그래픽카드 사용시 CUDA 드라이버를 추가로 설치
- 예제 코드를 실행하는 것에는 문제는 없음, 다만 크기가 큰 네트워크를 사용하려고 하는 경우에는 그래픽 카드를 사용한 연산이 빠르기 때문에 CUDA 드라이버를 설치 해야함

## ◆ 해결 방법

- 아래 링크를 통해 CUDA 드라이버를 설치
  - [https://developer.nvidia.com/cuda-11.0-download-archive?target\\_os=Windows&target\\_arch=x86\\_64&target\\_version=10&target\\_type=exenetwork](https://developer.nvidia.com/cuda-11.0-download-archive?target_os=Windows&target_arch=x86_64&target_version=10&target_type=exenetwork)

# **객체 탐지 (Object Detection)와 YOLO (You Only Look Once)**

# 객체 탐지 (Object Detection)



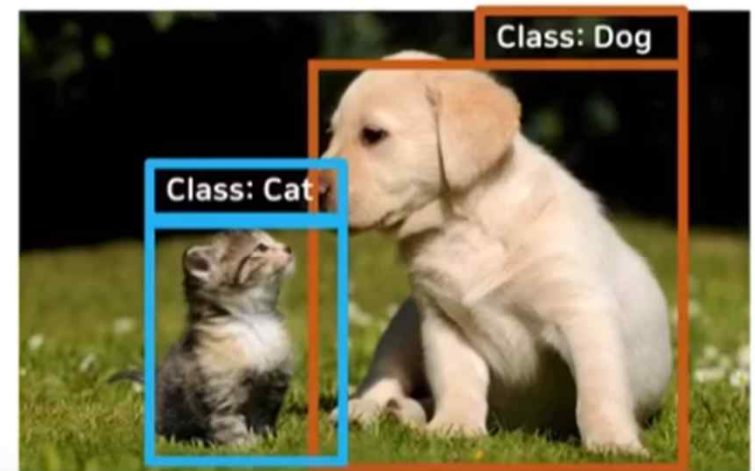
Object classification

- 이미지 내 single object
- Object class
- output: class probability



Object Localization

- 이미지 내 single object
- Object class
- Bounding Box(물체의 위치)
- output: (x, y, w, h)



Object Detection

- 이미지 내 multiple object
- Object class
- Bounding Box
- output:  
class probabilities + (x, y, w, h)  
class probabilities + (x, y, w, h)

# 객체 탐지 (Object Detection)

## ◆ 객체 탐지 (Object Detection)

- 참고: 객체 탐지 Object Detection - YOLO, 이수안 컴퓨터연구소, <https://www.youtube.com/watch?v=5ev0MMBzY3E>
- 주어진 이미지/영상에서 객체 (object)를 탐지하고 그 경계 상자 (bounding box) 제공
- object classification, localization 기능 포함
- 객체 탐지 알고리즘은 이미지/영상을 입력 받아 경계상자와 객체 클래스 리스트를 출력
- 경계 상자별로 대응하는 예측 클래스와 클래스의 신뢰도 (confidence)를 함께 출력

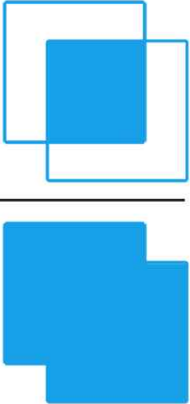
## ◆ 객체 탐지의 응용 분야

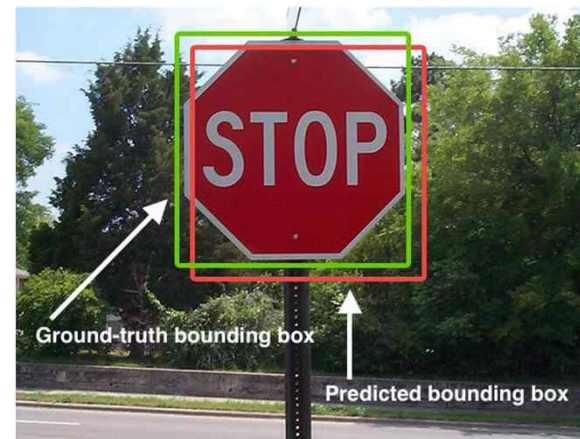
- 자율 주행 자동차에서 주변의 다른 자동차와 보행자, 자전거, 오토바이, 동물 (강아지, 고양이) 등을 인식
- 의료 분야에서 의료 영상 (X-ray / 초음파 / CT 사진)으로부터 종양이나 악성 조직을 탐색
- 제조업에서 조립 로봇이 제품을 조립하거나 수리할 때 부품 확인, 불량 부위 확인
- 보안 산업에서 위협 탐지, 출입 사람 확인, 사람 수 추출 및 확인

# 객체 탐지에서의 중요 성능/기능 요소 - IoU

## ◆ IoU (Intersection over Union)

- ref: <https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>
- ground-truth bounding box
- predicted bounding box

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$




## 객체 탐지에서의 중요 기능 요소 - NMS

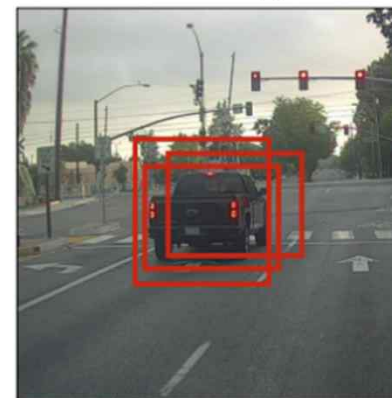
### ◆ Non-Maximum Suppression (NMS)

- input image에 대하여 object detection을 수행하면 object에 bbox(bounding box)가 그려지며, 어떤 물체일 확률값 (score)를 가짐
- 한가지 object에 많은 bbox가 생길 수 있음
- 동일한 object에 여러 개의 bbox가 구성되었다면, 가장 score가 높은 bbox만 남기고 나머지를 제거하는 것이 NMS임

#### Algorithm 1 Non-Max Suppression

```
1: procedure NMS( $B, c$ )
2:    $B_{nms} \leftarrow \emptyset$  Initialize empty set
3:   for  $b_i \in B$  do => Iterate over all the boxes
4:      $discard \leftarrow \text{False}$  Take boolean variable and set it as false. This variable indicates whether b(i) should be kept or discarded
5:     for  $b_j \in B$  do Start another loop to compare with b(i)
6:       if  $\text{same}(b_i, b_j) > \lambda_{nms}$  then If both boxes having same IOU
7:         if  $\text{score}(c, b_j) > \text{score}(c, b_i)$  then
8:            $discard \leftarrow \text{True}$  Compare the scores. If score of b(i) is less than that of b(j), b(i) should be discarded, so set the flag to True.
9:         if not  $discard$  then Once b(i) is compared with all other boxes and still the discarded flag is False, then b(i) should be considered. So
10:           $B_{nms} \leftarrow B_{nms} \cup b_i$  add it to the final list.
11:   return  $B_{nms}$  Do the same procedure for remaining boxes and return the final list
```

Before non-max suppression



After non-max suppression



Non-Max  
Suppression



## 객체 탐지/검출 알고리즘의 성능 평가 요소

### ◆ Precision (정밀도)

- $\text{Precision} = \text{true\_positive} / \text{all\_detections} = \text{true\_positive} / (\text{true\_positive} + \text{false\_positive})$

### ◆ Recall (재현율)

- $\text{Recall} = \text{true\_positive} / \text{all\_ground\_truths} = \text{true\_positive} / (\text{true\_positive} + \text{false\_negative})$

실제상황 (ground truth)	예측결과 (predict result)	
	Positive	Negative
Positive	TP(true positive) 옳은 검출	FN(false negative) 검출되어야 할 것이 검출되지 않았음
Negative	FP(false positive) 틀린 검출	TN(true negative) 검출되지 말아야 할 것이 검출되지 않았음

all detections

ground truth

## 객체 탐지/검출 알고리즘의 성능 평가 요소

### ◆ PR (precision-recall) curve

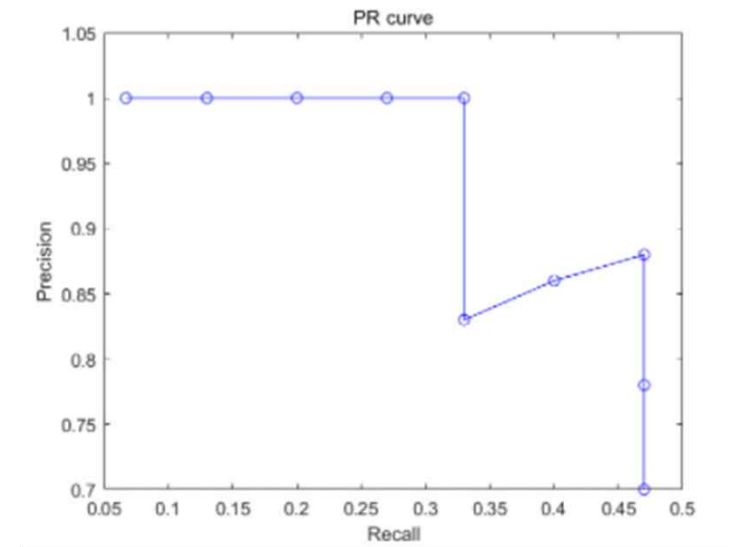
- x축은 recall (재현율), y축은 precision(정밀도)

### ◆ AP (average precision)

- PR curve 아래 부분의 면적

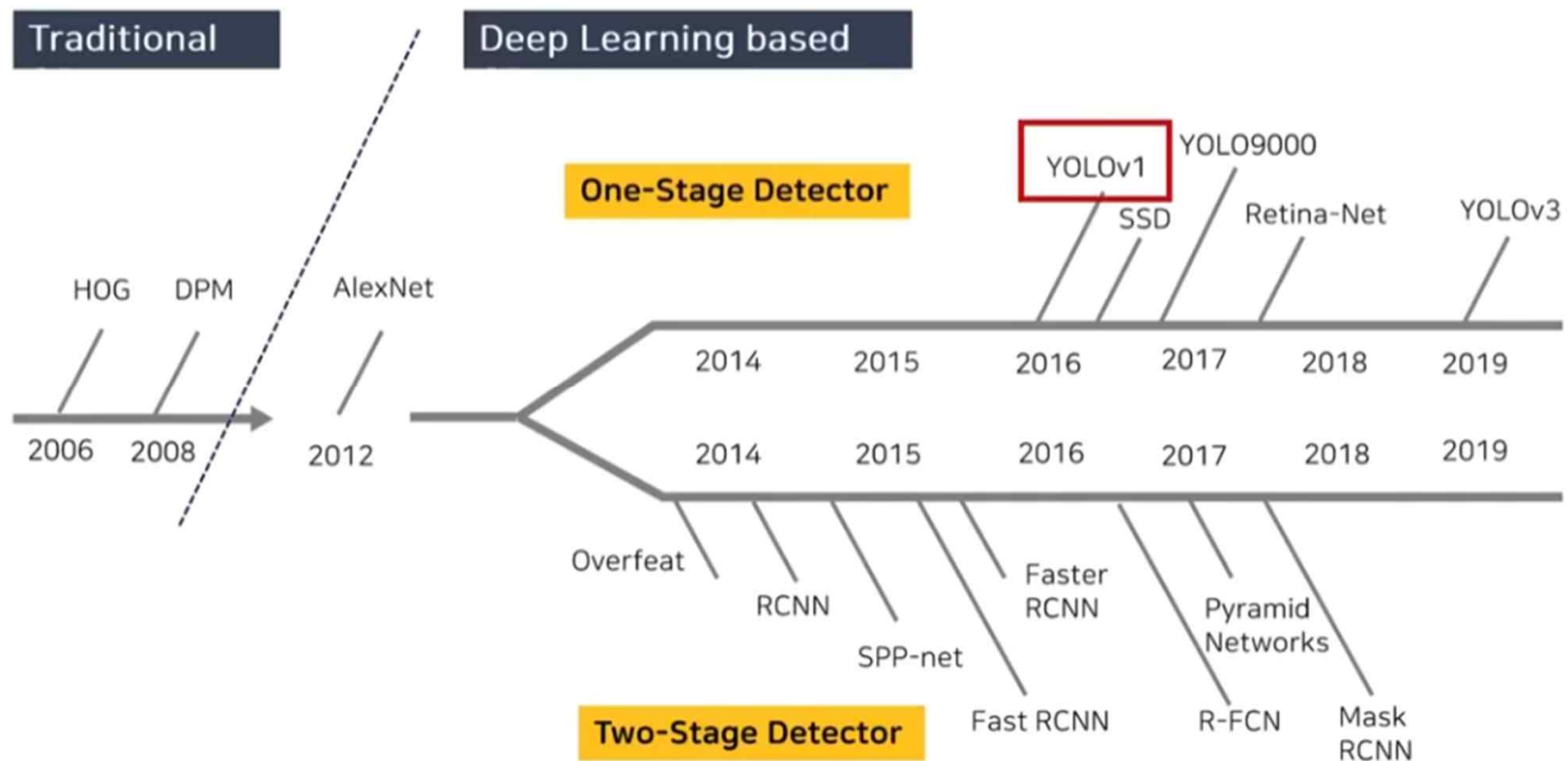
### ◆ mAP (mean average precision)

- 물체의 클래스가 여러 개인 경우, 각 클래스 당 AP를 구한 후, 그것을 모두 합한 다음, 물체 클래스의 개수로 나누어 평균값을 계산하여 알고리즘의 성능을 평가

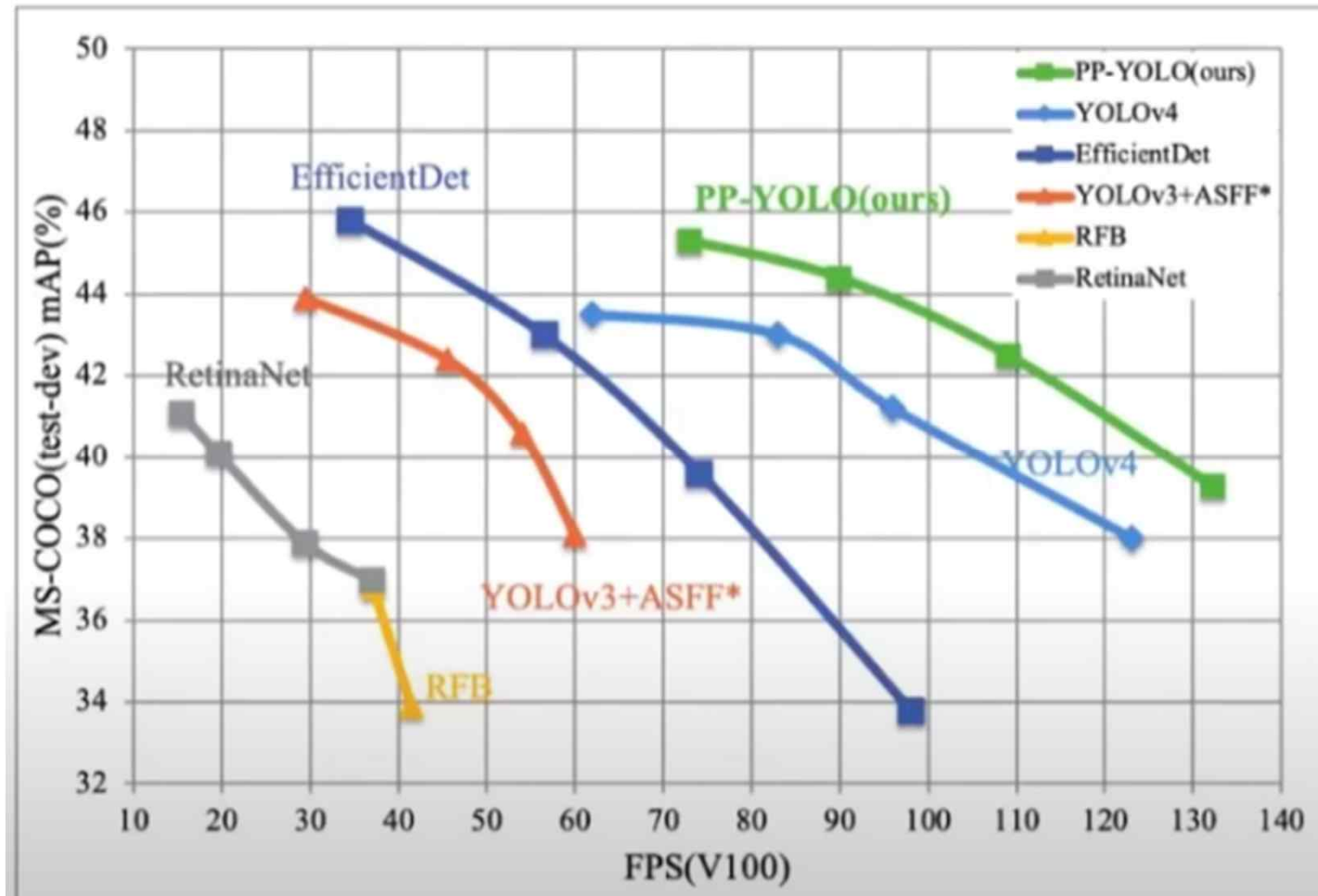




# Object Detection 기술 개발 역사



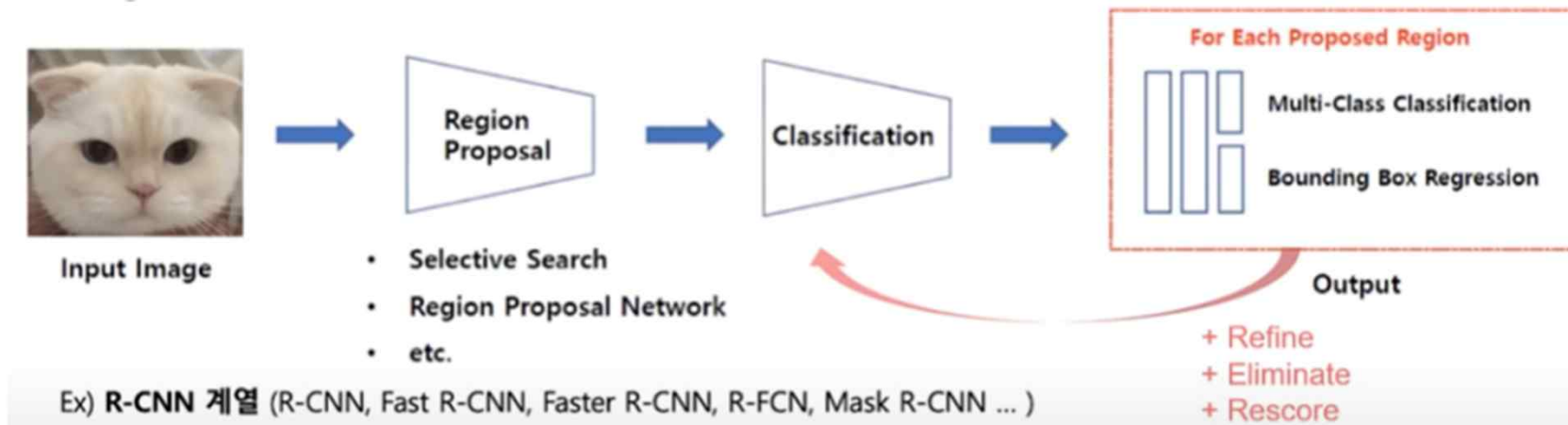
## Object Detection 알고리즘 들의 성능 비교



# Object Detection – DPIM, R-CNN

## ◆ 2 – stage Detector

- Regional proposal + classification

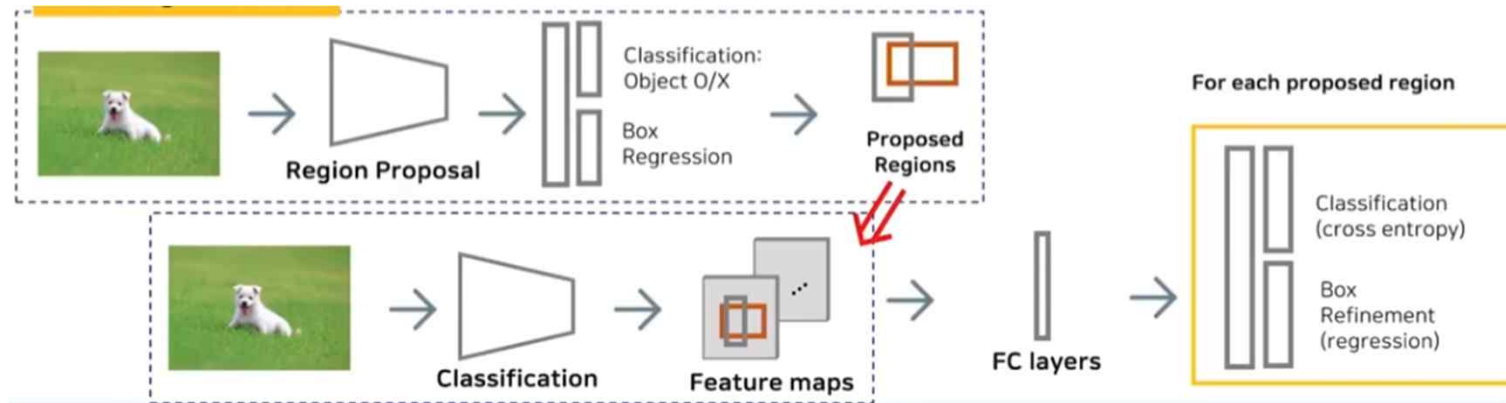


## ◆ 2 – stage Detector의 문제점

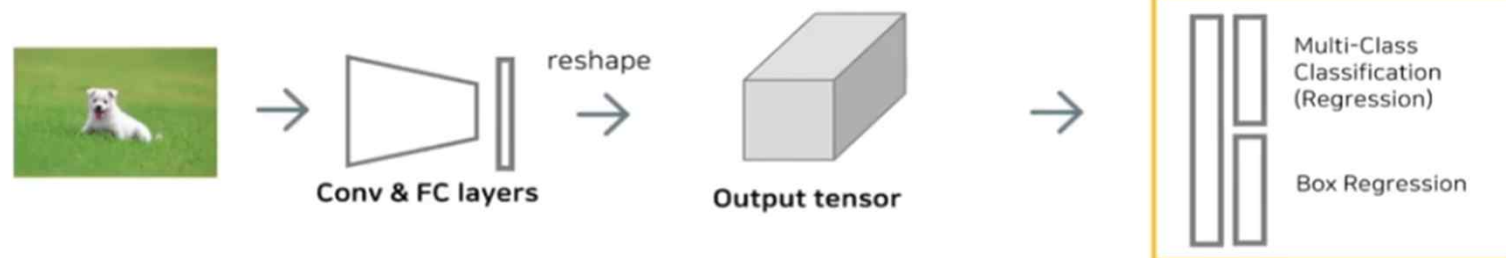
- 처리 속도가 느림
- 최적화하기가 어려움

# Two-stage Detection vs. One-stage Detection

## ◆ Two-stage Detection



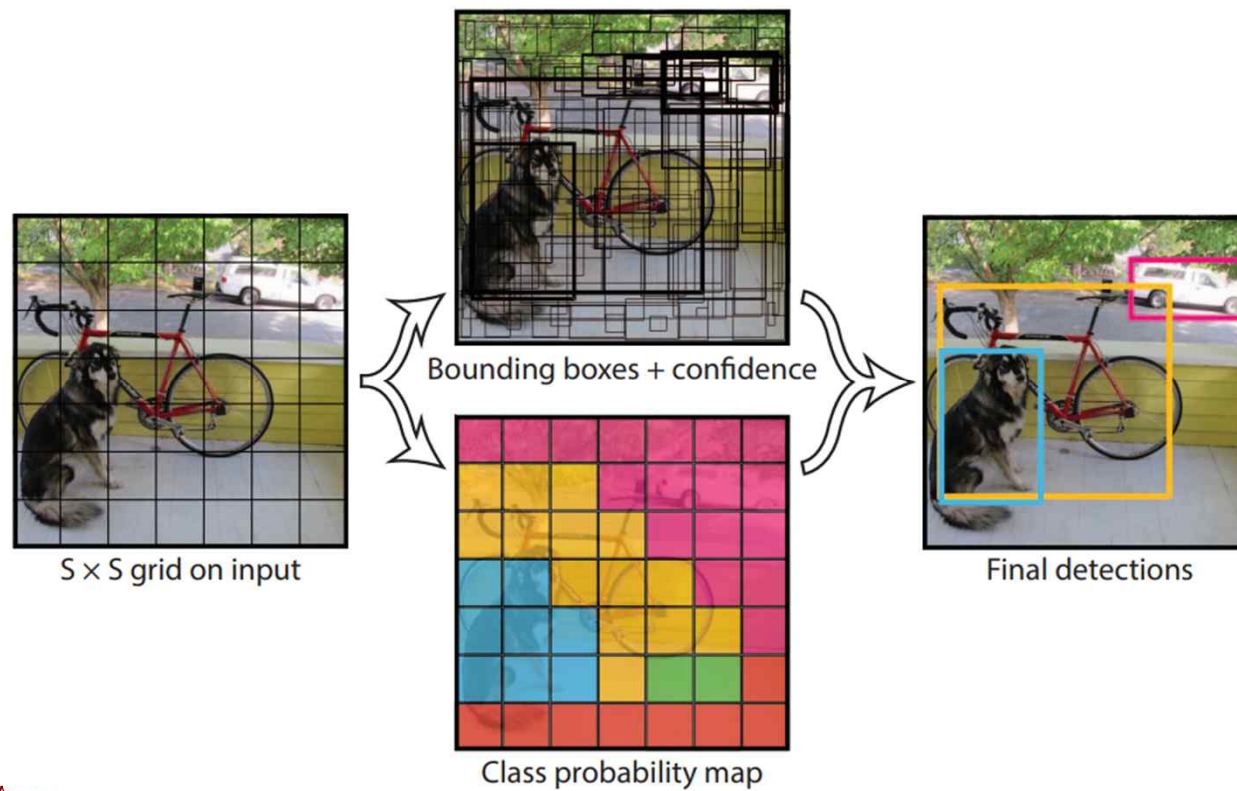
## ◆ One-stage Detection



# Unified Realtime Object Detection

## ◆ Unified Real-time Object Detection in YOLO

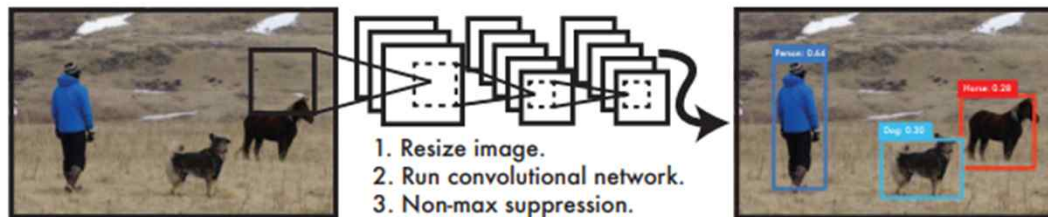
- 참고: You Only Look Once: Unified, Real-Time Object Detection, IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016 <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7780460>



# YOLO (You Only Look Once)

## ◆ YOLO 특징

- You Only Look Once: Unified, Real-Time Object Detection, <https://www.youtube.com/watch?v=NM6lrxy0bxs>
- Object detection을 regression problem으로 관점 전환
- Unified architecture – 하나의 신경망으로 classification과 localization을 처리
- Single-stage object detection
- bounding box 좌표 및 class 확률 계산을 동시에 진행
- 처리 속도가 빠름 (base: 45fps, fast: 150fps)
- 전체적인 이미지를 보고 추론하며, 배경을 객체로 인식하는 오류가 감소됨
- 일반화된 표현 학습을 사용하므로 새로운 도메인에 적용 용이
- 여러 도메인에서 object detection 가능



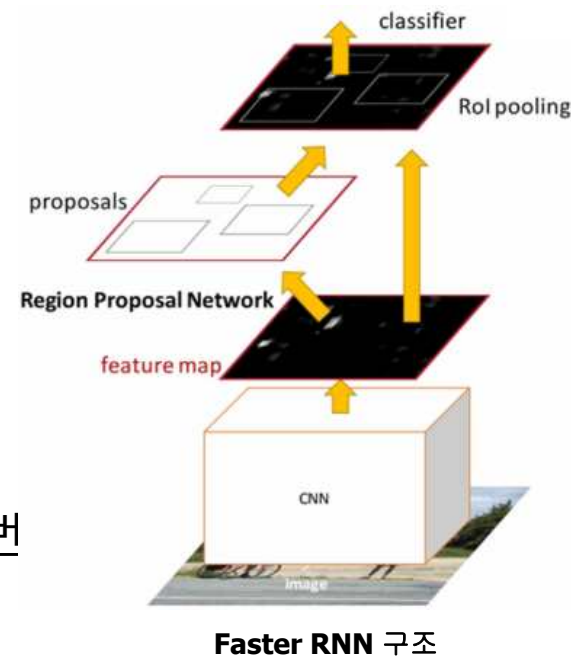
## 기존 방식과의 차이점

### ◆ R-CNN (Region Convolutional Neural Network)

- 경계박스 (Bound Box) 검출
- 영상에서 관심영역 (ROI – Region of Interest) 추출
- 분류기 (Classification Network)에 의해 클래스 분류

### ◆ R-CNN 특징

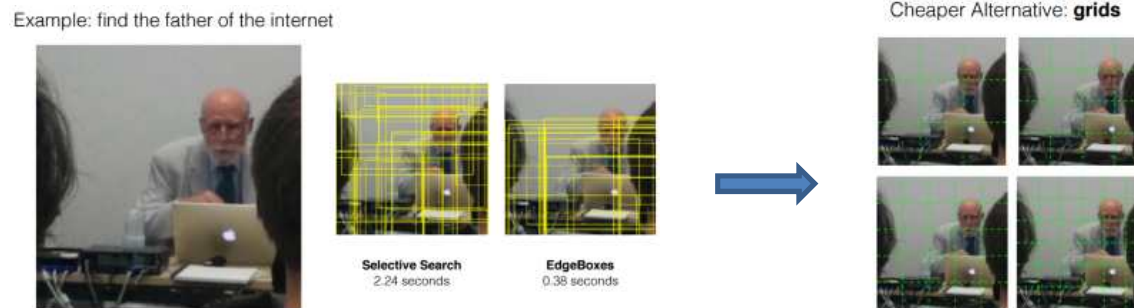
- CNN을 통해 경계박스를 추출
- RPN을 통해 경계박스를 추천
- Classifier (이것도 NN)로 클래스 검사 및 분류
- 역할이 다른 3개의 네트워크가 하나로 연결되어 한번 (End to End)
- Propose의 수가 많고 그 과정에서 오버헤드가 크다



## 기존 방식과의 차이점

### ◆ 경계박스를 찾는 방식

- Proposal 방식과 Grid 방식



- Proposal은 영상에서 수 많은 제안 (proposal)을 하는 selective search로 속도가 늦다
- Grid 방식은 각 셀이 proposal의 수가 되며, proposal을 구하는 과정에서 오버헤드가 없다

### ◆ YOLO는 실시간성을 확보하기 위해 proposal의 수가 적은 grid 방식을 발전시켜 사용



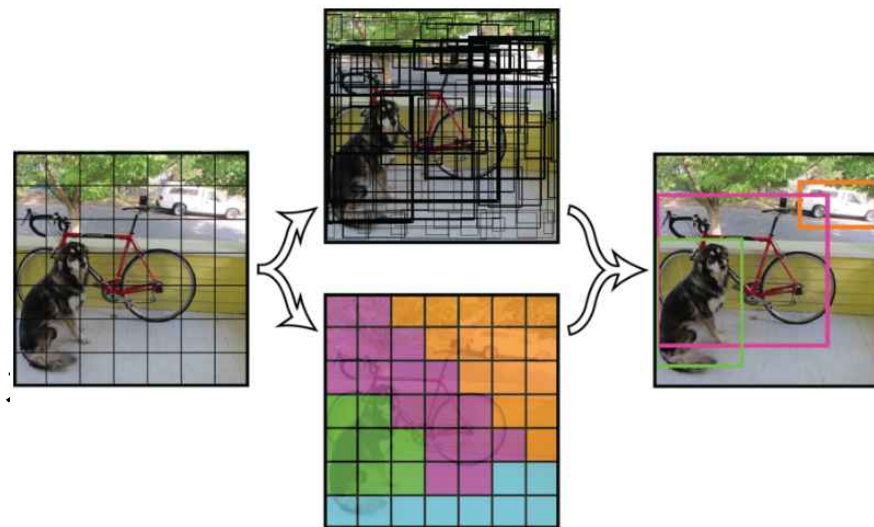
# YOLO 네트워크

## ◆ 간단하고 빠르다

- 하나의 네트워크로 특징 추출, 경계박스 생성, 클래스 분류를 한다
- 최종 출력단에서 경계박스 위치 찾기와 클래스 분류가 동시에 이루어진다

## ◆ 네트워크 출력 예

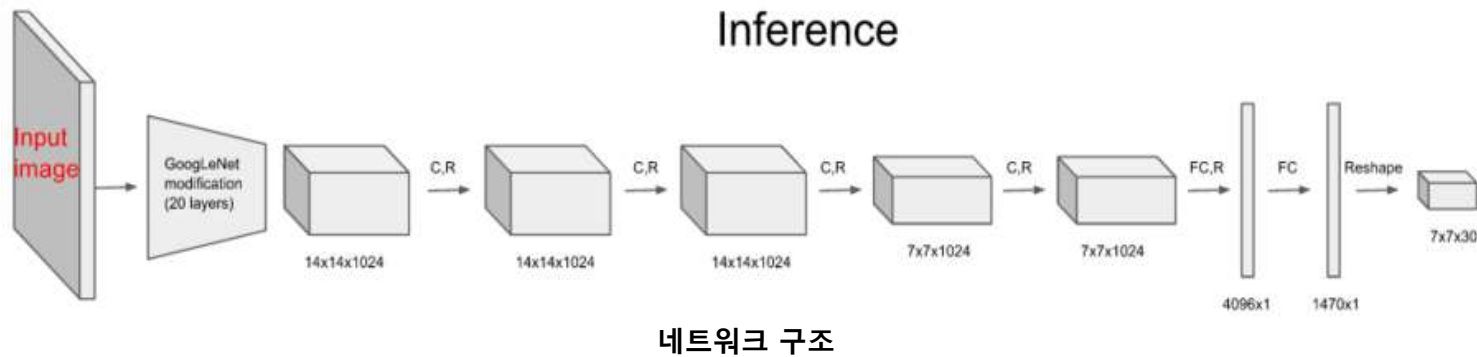
- 중앙 네트워크를 통해 두 가지 출력 획득
- 7 x 7 grid
- 각 그리드의 중심에서 크기가 일정하지 않은 2개의 경계박스 생성 (98개)
- ROI (region of interest) 혹은 오브젝트 후보의 경계들은 임계 값 (0.5) 보다 작으면 삭제



# YOLO 네트워크

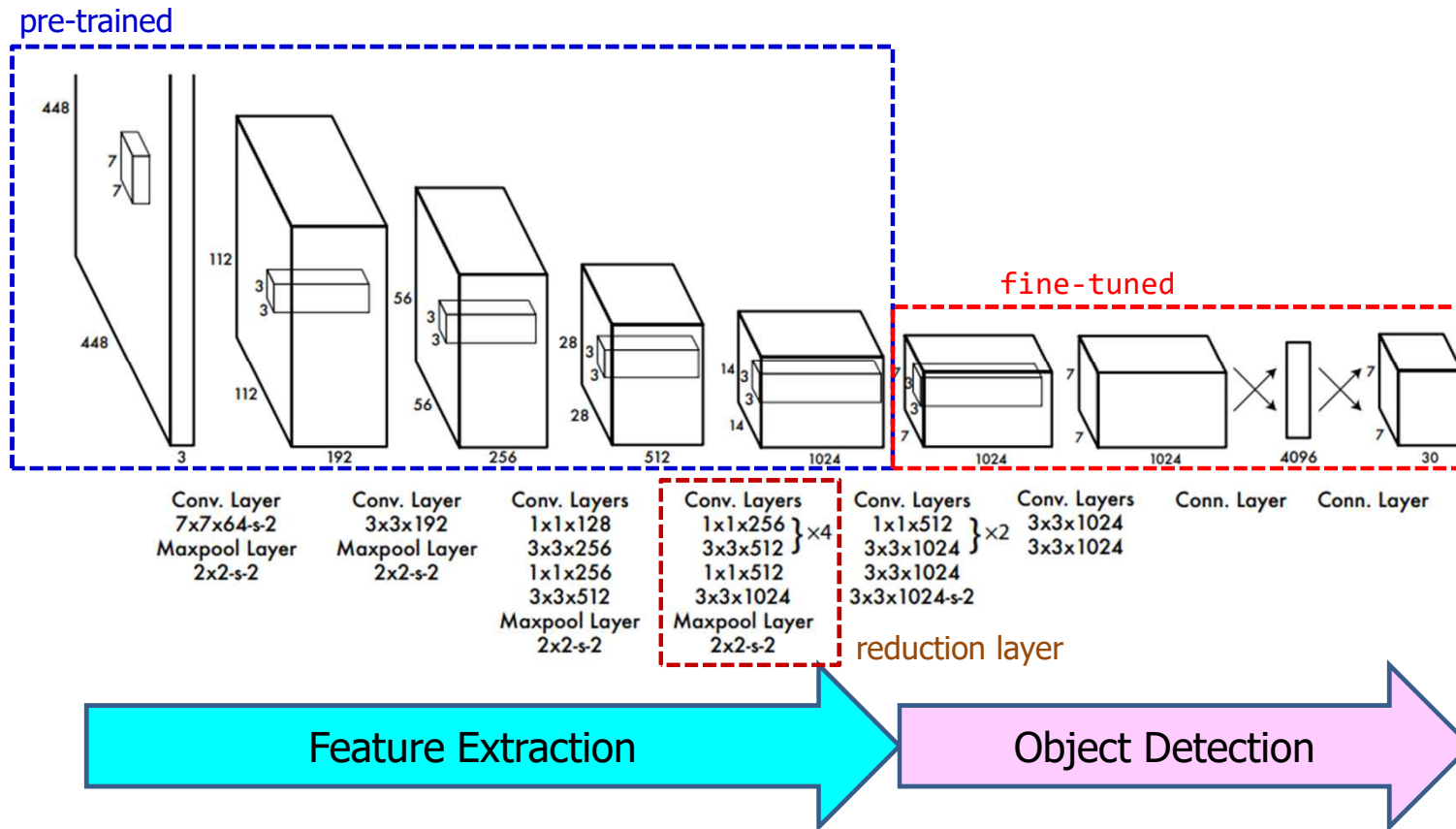
## ◆ 네트워크 출력

- 오브젝트를 확신 (confidence score) 할수록 경계를 굵게 표시
- 얇은 경계박스 (오브젝트가 없다고 추론)는 삭제
- 남은 경계박스를 NMS (Non-maximal suppression 비-최대값 억제) 알고리즘을 적용하여 선별
- 경계박스의 색깔로 클래스를 구별



# Unified Realtime Object Detection

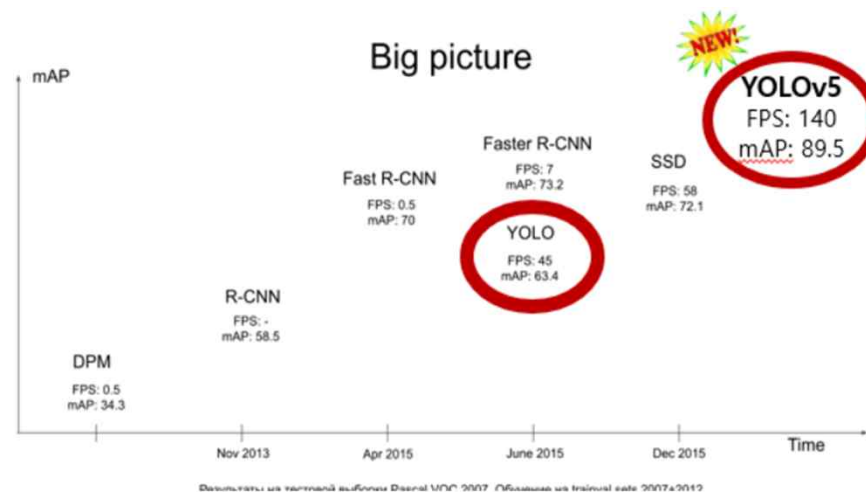
## ◆ YOLO - 24 convolutional layers followed by 2 fully connected layers



# YOLOv5






## ◆ Evolution of YOLO-v5 Algorithm for Object Detection

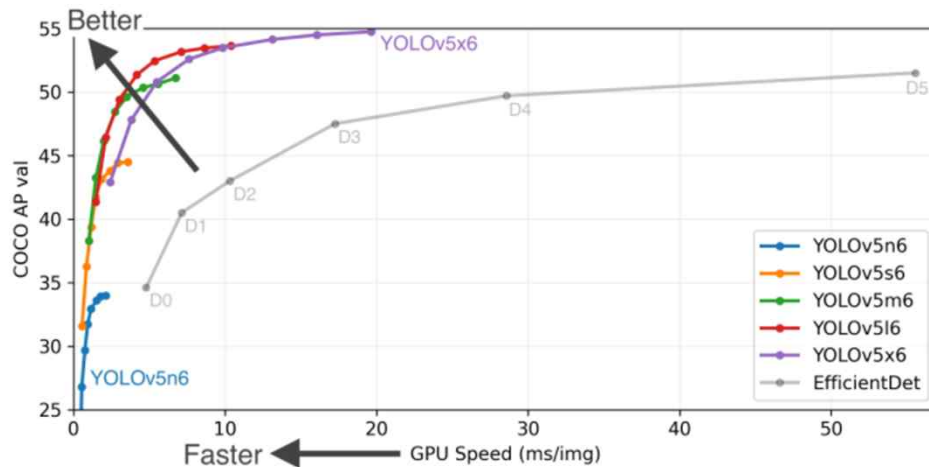
- <https://docs.ultralytics.com/>
- [https://pytorch.org/hub/ultralytics\\_yolov5/](https://pytorch.org/hub/ultralytics_yolov5/)
- use Pytorch framework
- released 18 May 2020



# YOLOv5

## ◆ Performance Enhancement in YOLOv5

				
Nano YOLOv5n	Small YOLOv5s	Medium YOLOv5m	Large YOLOv5l	XLarge YOLOv5x
4 MB <sub>FP16</sub> 6.3 ms <sub>V100</sub> 28.4 mAP <sub>COCO</sub>	14 MB <sub>FP16</sub> 6.4 ms <sub>V100</sub> 37.2 mAP <sub>COCO</sub>	41 MB <sub>FP16</sub> 8.2 ms <sub>V100</sub> 45.2 mAP <sub>COCO</sub>	89 MB <sub>FP16</sub> 10.1 ms <sub>V100</sub> 48.8 mAP <sub>COCO</sub>	166 MB <sub>FP16</sub> 12.1 ms <sub>V100</sub> 50.7 mAP <sub>COCO</sub>

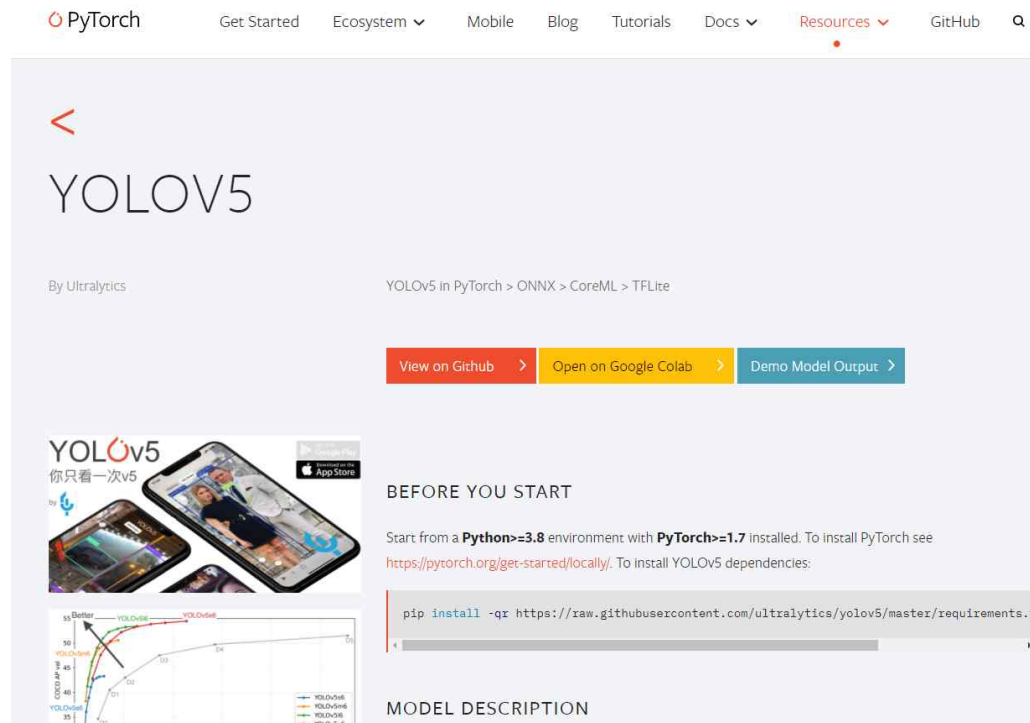


## **Desktop PC에 YOLOv8의 설치 및 객체 탐지**

# YOLOv5 in PyTorch 설치

## ◆ YOLOv5 in PyTorch 설치

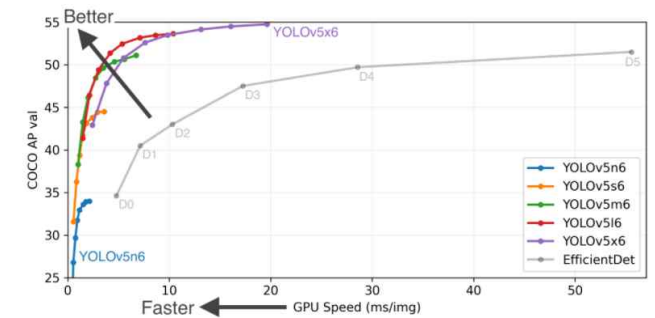
- [https://pytorch.org/hub/ultralytics\\_yolov5/](https://pytorch.org/hub/ultralytics_yolov5/)
- Start from a **Python**  $\geq 3.8$  environment with **PyTorch**  $\geq 1.7$  installed.



# YOLOv5

YOLOv5 🚀 is a family of compound-scaled object detection models trained on the COCO dataset, and includes simple functionality for Test Time Augmentation (TTA), model ensembling, hyperparameter evolution, and export to ONNX, CoreML and TFLite.

Model	size (pixels)	mAP <sup>val</sup> 0.5:0.95	mAP <sup>test</sup> 0.5:0.95	mAP <sup>val</sup> 0.5	Speed V100 (ms)	params (M)	FLOPS 640 (B)
YOLOv5s6	1280	43.3	43.3	61.9	<b>4.3</b>	12.7	17.4
YOLOv5m6	1280	50.5	50.5	68.7	8.4	35.9	52.4
YOLOv5l6	1280	53.4	53.4	71.1	12.3	77.2	117.7
YOLOv5x6	1280	<b>54.4</b>	<b>54.4</b>	<b>72.0</b>	22.4	141.8	222.9
YOLOv5x6 TTA	1280	<b>55.0</b>	<b>55.0</b>	<b>72.0</b>	70.8	-	-





# PyTorch가 설치되어 있지 않은 경우

## ◆ PyTorch 설치가 되어 있지 않은 경우

- <https://pytorch.org/get-started/locally/>에서 설치

PyTorch Build	Stable (1.10.2)	Preview (Nightly)	LTS (1.8.2)
Your OS	Linux	Mac	Windows
Package	Conda	Pip	LibTorch
Language	Python	C++/Java	Source
Compute Platform	CUDA 10.2	CUDA 11.3	ROCm 4.2 (beta)
Run this Command:	<pre>pip3 install torch==1.10.2+cu113 torchvision==0.11.3+cu113 torchaudio==0.10.2+cu113 -f https://download.pytorch.org/whl/cu113/torch_stable.html</pre>		

```
# testing torch module
import torch
x = torch.rand(5, 3)
print(x)
```

```
tensor([[0.1389, 0.9205, 0.2907],
        [0.6530, 0.5574, 0.1454],
        [0.4052, 0.0981, 0.3995],
        [0.9109, 0.4374, 0.6793],
        [0.3973, 0.0814, 0.0244]])
```

```
C:\Users\Owner>pip3 install torch==1.10.2+cu113 torchvision==0.11.3+cu113 torchaudio==0.10.2+cu113 -f https://download.pytorch.org/whl/cu113/torch_stable.html
Looking in links: https://download.pytorch.org/whl/cu113/torch_stable.html
Collecting torch==1.10.2+cu113
  Downloading https://download.pytorch.org/whl/cu113/torch-1.10.2%2Bcu113-cp39-cp39-win_amd64.whl (2442.3 MB)
    Collecting torch==1.10.2+cu113
      Downloading https://download.pytorch.org/whl/cu113/torchvision-0.11.3%2Bcu113-cp39-cp39-win_amd64.whl (3.1 MB)
    Collecting torchaudio==0.10.2+cu113
      Downloading https://download.pytorch.org/whl/cu113/torchaudio-0.10.2%2Bcu113-cp39-cp39-win_amd64.whl (336 kB)
Requirement already satisfied: typing-extensions in c:\Users\Owner\AppData\Roaming\Python\Python39\site-packages (from torch==1.10.2+cu113) (3.7.4.3)
Requirement already satisfied: pillow!=8.3.0, >=5.3.0 in c:\Users\Owner\AppData\Local\Programs\Python\Python39\lib\site-packages (from torchvision==0.11.3+cu113) (8.2.0)
Requirement already satisfied: numpy in c:\Users\Owner\AppData\Local\Programs\Python\Python39\lib\site-packages (from torchaudio==0.11.3+cu113) (1.19.4)
Installing collected packages: torch, torchvision, torchaudio
  Attempting uninstall: torch
    Found existing installation: torch 1.10.1+cu113
    Uninstalling torch-1.10.1+cu113:
      Successfully uninstalled torch-1.10.1+cu113
  Attempting uninstall: torchvision
    Found existing installation: torchvision 0.11.2+cu113
    Uninstalling torchvision-0.11.2+cu113:
      Successfully uninstalled torchvision-0.11.2+cu113
  Attempting uninstall: torchaudio
    Found existing installation: torchaudio 0.10.1+cu113
    Uninstalling torchaudio-0.10.1+cu113:
      Successfully uninstalled torchaudio-0.10.1+cu113
Successfully installed torch-1.10.2+cu113 torchaudio-0.10.2+cu113 torchvision-0.11.3+cu113
C:\Users\Owner>
```

# YOLOv5 in PyTorch 종속항목 설치

## ◆ Install YOLOv5 dependencies:

### BEFORE YOU START

Start from a **Python**≥3.8 environment with **PyTorch**≥1.7 installed. To install PyTorch see <https://pytorch.org/get-started/locally/>. To install YOLOv5 dependencies:

```
pip install -qr https://raw.githubusercontent.com/ultralytics/yolov5/master/requirements.txt
```

```
C:\Users\Owner>pip install -qr https://raw.githubusercontent.com/ultralytics/yolov5/master/requirements.txt  
C:\Users\Owner>
```

# requirements.txt

```
# pip install -r requirements.txt

# base -----
Cython
matplotlib>=3.2.2
numpy>=1.18.5
opencv-python>=4.1.2
Pillow
PyYAML>=5.3
scipy>=1.4.1
tensorboard>=2.2
torch>=1.6.0
torchvision>=0.7.0
tqdm>=4.41.0

# logging -----
# wandb

# plotting -----
seaborn
pandas

# export -----
# coremltools==4.0
# onnx>=1.8.0
# scikit-learn==0.19.2 # for coreml quantization

# extras -----
thop # FLOPS computation
pycocotools>=2.0 # COCO mAP
```



# YOLOv5 detect.py

## ◆ detect.py

- runs inference on a variety of sources and saving results to runs/detect

## ◆ sources that can be included

- image
- webcam
- video
- directory
- YouTube
- RTSP, RTMP, Http stream



# git 설치

## ◆ git 설치

- <https://git-scm.com/download/win>

### Download for Windows

Click [here to download](#) the latest (2.35.0) 64-bit version of **Git for Windows**. This is the most recent [maintained build](#). It was released **4 days ago**, on 2022-01-24.

#### Other Git for Windows downloads

##### Standalone Installer

[32-bit Git for Windows Setup.](#)

[64-bit Git for Windows Setup.](#)

##### Portable ("thumbdrive edition")

[32-bit Git for Windows Portable.](#)

[64-bit Git for Windows Portable.](#)

#### Using winget tool

Install [winget tool](#) if you don't already have it, then type this command in command prompt or Powershell.

```
winget install --id Git.Git -e --source winget
```

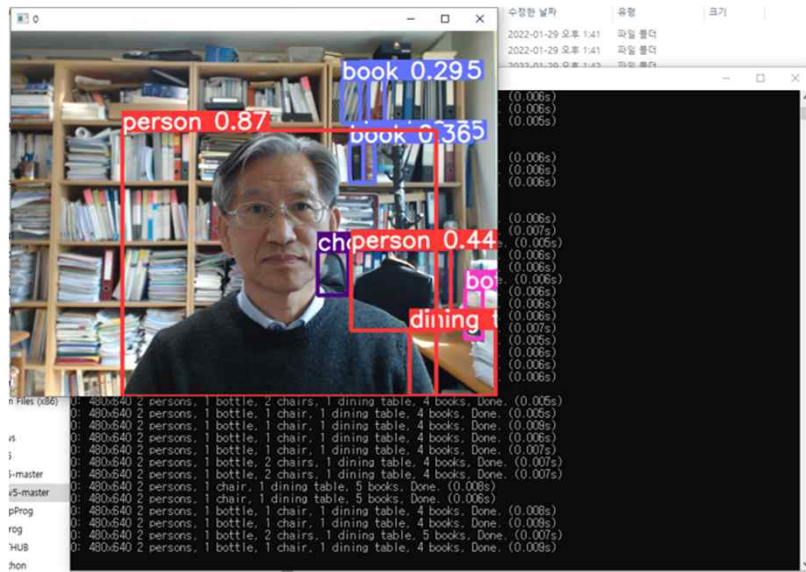
The current source code release is version **2.35.0**. If you want the newer version, you can build it from [the source code](#).



# Git CMD 창에서 clone 및 detect.py 실행

## ◆ Git CMD 창에서 다음 명령어 실행

- git clone https://github.com/ultralytics/yolov5
- cd yolov5
- python detect.py --source 0



```
C:\Users\Owner>git clone https://github.com/ultralytics/yolov5
Cloning into 'yolov5'...
remote: Enumerating objects: 10740, done.
remote: Total 10740 (delta 0), reused 0 (delta 0), pack-reused 10740
1.82 MiB/s
Receiving objects: 100% (10740/10740), 10.87 MiB | 10.96 MiB/s, done.
Resolving deltas: 100% (7418/7418), done.

C:\Users\Owner>cd yolov5

C:\Users\Owner\yolov5>dir
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: CEF4-00E1

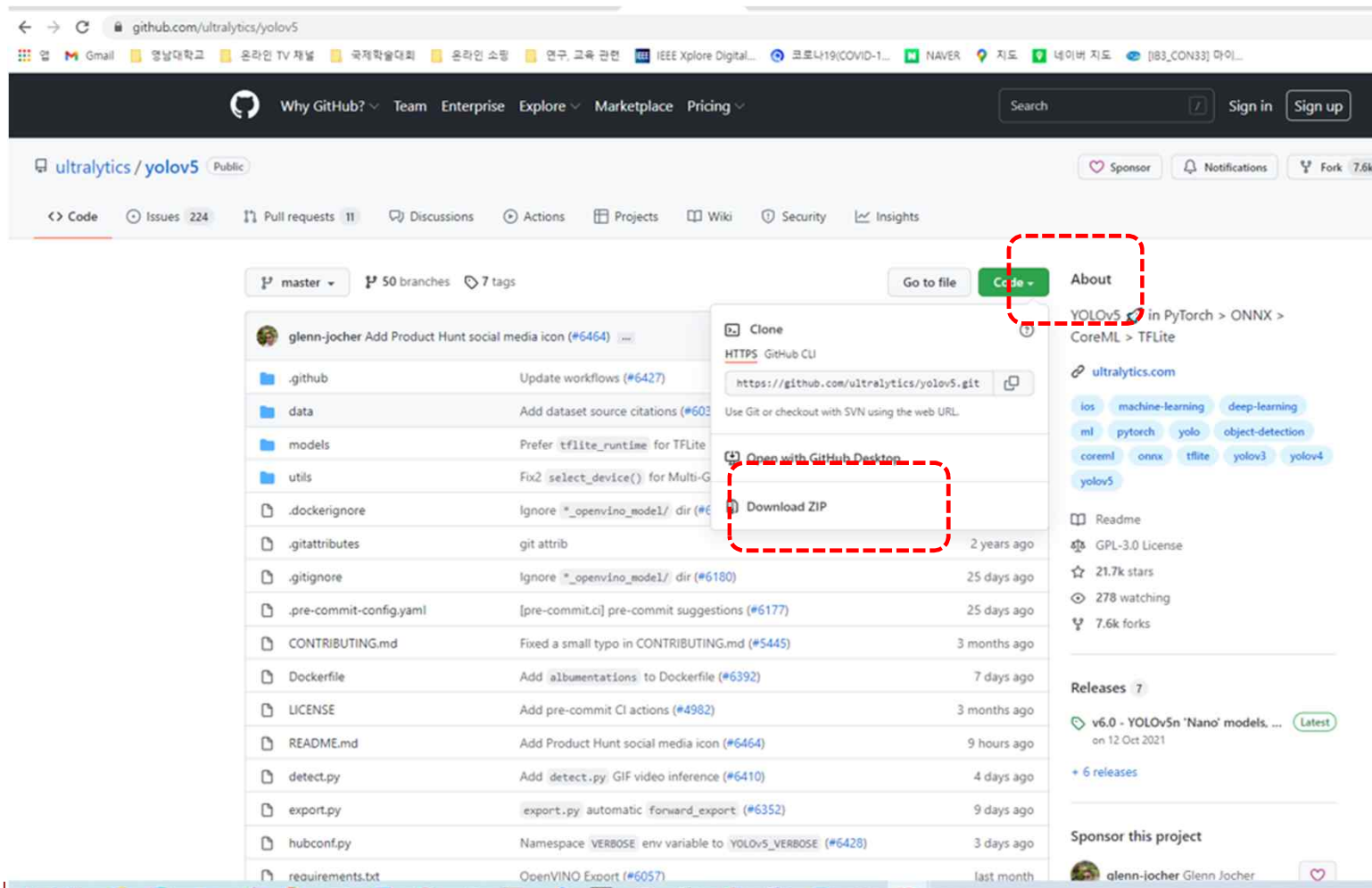
C:\Users\Owner\yolov5 디렉터리

2022-01-29 오후 03:32 <DIR> .
2022-01-29 오후 03:32 <DIR> ..
2022-01-29 오후 03:32 3,924 .dockerignore
2022-01-29 오후 03:32 77 .gitattributes
2022-01-29 오후 03:32 <DIR> .github
2022-01-29 오후 03:32 4,238 .gitignore
2022-01-29 오후 03:32 1,620 .pre-commit-config.yaml
2022-01-29 오후 03:32 5,085 CONTRIBUTING.md
2022-01-29 오후 03:32 <DIR> data
2022-01-29 오후 03:32 13,817 detect.py
2022-01-29 오후 03:32 2,242 Dockerfile
2022-01-29 오후 03:32 25,952 export.py
2022-01-29 오후 03:32 6,568 hubconf.py
2022-01-29 오후 03:32 35,801 LICENSE
2022-01-29 오후 03:32 <DIR> models
2022-01-29 오후 03:32 15,909 README.md
2022-01-29 오후 03:32 963 requirements.txt
2022-01-29 오후 03:32 974 setup.cfg
2022-01-29 오후 03:32 34,110 train.py
2022-01-29 오후 03:32 57,557 tutorial.ipynb
2022-01-29 오후 03:32 <DIR> utils
2022-01-29 오후 03:32 19,272 val.py
16개 파일 228,109 바이트
6개 디렉터리 1,027,441,836,032 바이트 남음

C:\Users\Owner\yolov5>
```

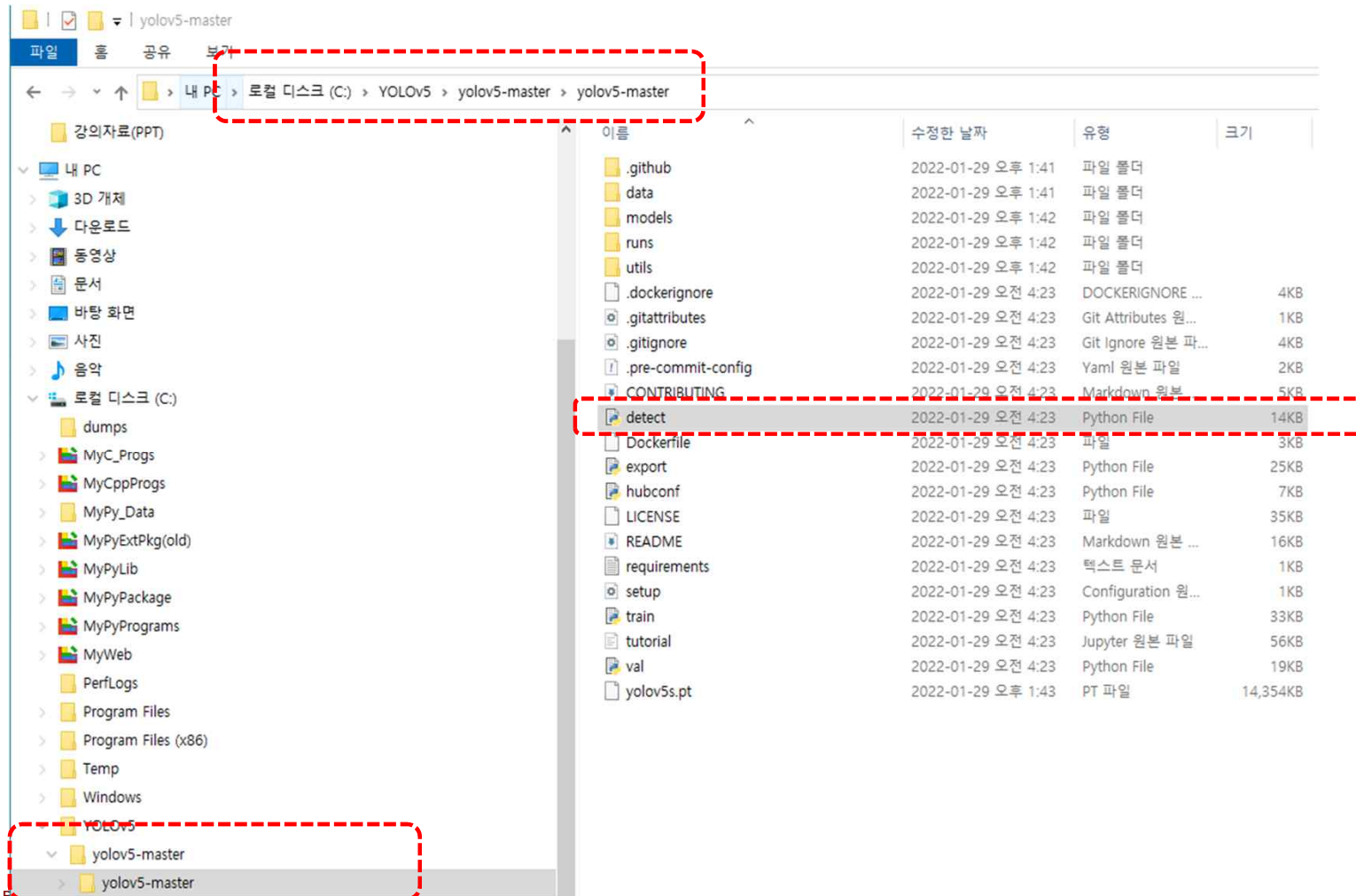


# <https://github.com/ultralytics/yolov5>에서 패키지 download 및 설치





## download된 패키지를 지정된 위치에 설치





# detect.py를 사용한 webcam의 영상 인식 기능 실행

## ◆ detect.py를 사용한 webcam의 영상 인식 기능 실행

- YOLOv5-master가 설치된 경로 (예)  
C:\YOLOv5\yolov5-master\yolov5-master
- detect.py 파이썬 프로그램 소스코드 확인
- Command 창에서 YOLOv5-master가 설치된 폴더로 이동한 후, 다음 명령어 입력
- >python detect.py --source 0

```
명령 프롬프트
Microsoft Windows [Version 10.0.19043.1466]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Owner>cd C:\YOLOv5\yolov5-master\yolov5-master
C:\YOLOv5\yolov5-master\yolov5-master>python detect.py --source 0
```



## detect.py를 사용한 video 영상 인식 기능 실행

### ◆ detect.py를 사용한 video 영상 인식 기능 실행

- YOLOv5-master가 설치된 경로 (예)  
C:\YOLOv5\yolov5-master\yolov5-master
- detect.py 파이썬 프로그램 소스코드 확인
- Command 창에서 YOLOv5-master가 설치된 폴더로 이동한 후, 다음 명령어 입력
- (또는 git CMD 창에서 cd yolov5 실행 후)

```
> python detect.py -source  
test_video.mp4
```



# YOLOv5 in PyTorch 기본 기능 시험 파이썬 프로그램

```
# Basic YOLOv5 test
# (ref: https://ultralytics.com/)

import torch

# Model
model = torch.hub.load('ultralytics/yolov5', 'yolov5s', pretrained=True)

# Images
imgs = ['https://ultralytics.com/images/zidane.jpg'] # batch of images

# Inference
results = model(imgs)

# Results
results.print()
results.save() # or .show()

results.xyxy[0] # img1 predictions (tensor)
results.pandas().xyxy[0] # img1 predictions (pandas)
#   xmin  ymin  xmax  ymax  confidence  class  name
# 0  749.50  43.50 1148.0 704.5    0.874023     0  person
# 1  433.50 433.50  517.5 714.5    0.687988    27   tie
# 2  114.75 195.75 1095.0 708.0    0.624512     0  person
# 3  986.00 304.00 1028.0 420.0    0.286865    27   tie
results.show()
```

```
Downloading: "https://github.com/ultralytics/yolov5/archive/master.zip" to C:\Users\Owner\.cache\torch\hub\master.zip
Downloading https://ultralytics.com/assets/Arial.ttf to C:\Users\Owner\AppData\Roaming\Ultralytics\Arial.ttf...
YOLOv5 2022-1-29 torch 1.10.1+cu113 CUDA:0 (NVIDIA GeForce RTX 3060 Ti, 8192MiB)
```

```
Downloading https://github.com/ultralytics/yolov5/releases/download/v6.0/yolov5s.pt to yolov5s.pt...
0%|          | 0.00/14.0M [00:00<?, ?B/s] 2%|          | 216k/14.0M [00:00<00:06, 2.20MB/s] 4%|          | 616k/14.0M [00:00<00:04, 3.14MB/s] 11%|          | 1.60M/14.0M [00:00<00:02, 5.11MB/s] 24%|          | 3.40M/14.0M [00:00<00:01, 9.71MB/s] 32%|          | 4.54M/14.0M [00:00<00:00, 10.4MB/s] 40%|          | 5.66M/14.0M [00:00<00:00, 10.8MB/s] 48%|          | 6.73M/14.0M [00:00<00:00, 10.1MB/s] 59%|          | 8.24M/14.0M [00:00<00:00, 11.7MB/s] 67%|          | 9.40M/14.0M [00:00<00:00, 11.7MB/s] 75%|          | 10.5M/14.0M [00:01<00:00, 11.8MB/s] 83%|          | 11.7M/14.0M [00:01<00:00, 11.8MB/s] 92%|          | 12.6M/14.0M [00:01<00:00, 11.8MB/s] 100%|          | 14.0M/14.0M [00:01<00:00, 10.0MB/s]
```

```
Fusing layers...
Model Summary: 213 layers, 7225885 parameters, 0 gradients
Adding AutoShape...
image 1/1: 720x1280 2 persons, 1 tie
Speed: 698.9ms pre-process, 16.9ms inference, 23.9ms NMS per image at shape (1, 3, 384, 640)
Saved 1 image to -[imruns\detect\exp-0m
>>>
```



# YOLOv5 Training

## ◆ Training

- 기본 데이터를 사용하여 훈련
- 사용자 지정 훈련 데이터 (train custom data)를 사용하여 훈련

## ◆ 기본 훈련 데이터

- PASCAL Visual Object Classes
- COCO (Common Objects in Context), <https://cocodataset.org/#home>
- roboflow (<https://public.roboflow.com/object-detection>): 35 object detection, 4 classification

# YOLOv5 Training

## ◆ Train Custom Data

- <https://docs.ultralytics.com/tutorials/train-custom-datasets/>

```
# train and val data as 1) directory: path/images/, 2) file: path/images.txt, or 3)
train: ../coco128/images/train2017/
val: ../coco128/images/train2017/

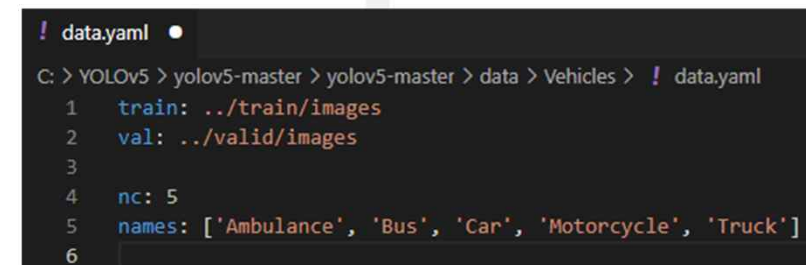
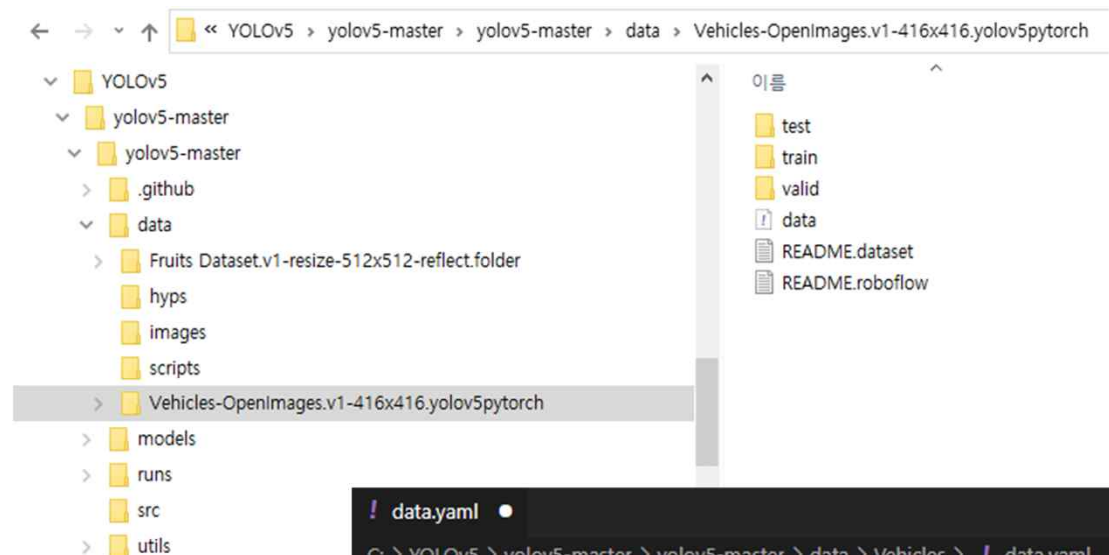
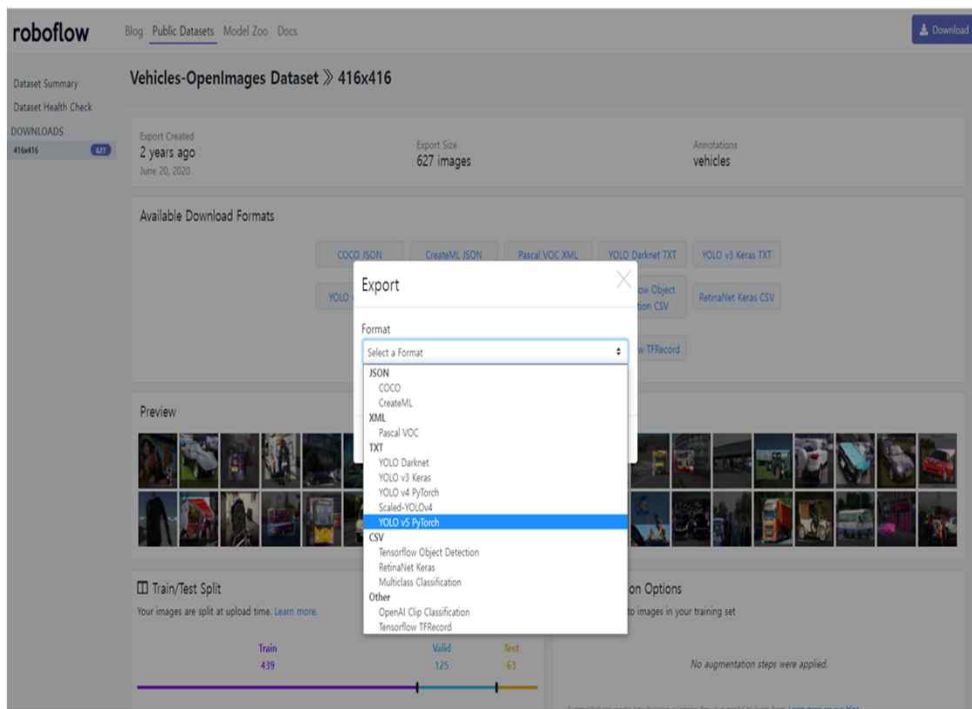
# number of classes
nc: 80

# class names
names: ['person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train', 'truck',
        'fire hydrant', 'stop sign', 'parking meter', 'bench', 'bird', 'cat', 'dog',
        'elephant', 'bear', 'zebra', 'giraffe', 'backpack', 'umbrella', 'handbag',
        'skis', 'snowboard', 'sports ball', 'kite', 'baseball bat', 'baseball glove',
        'tennis racket', 'bottle', 'wine glass', 'cup', 'fork', 'knife', 'spoon',
        'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza', 'donut', 'ice cream',
        'potted plant', 'bed', 'dining table', 'toilet', 'tv', 'laptop', 'mouse',
        'cell phone', 'microwave', 'oven', 'toaster', 'sink', 'refrigerator', 'book',
        'teddy bear', 'hair drier', 'toothbrush']
```

# Custom Dataset으로 YOLOv5 학습

## ◆ Training Dataset 준비

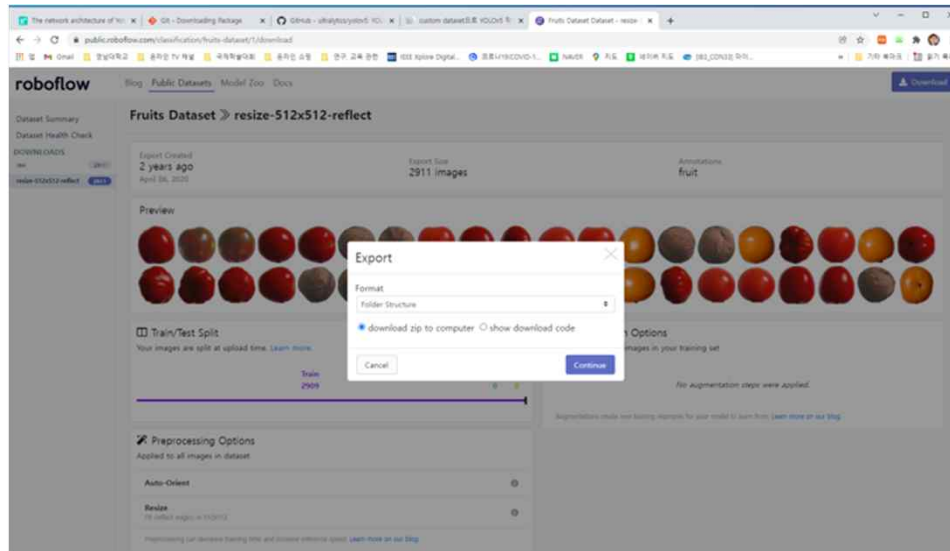
- 참고자료: <https://sguys99.github.io/ds01>
- roboflow (<https://public.roboflow.com/object-detection>)  
-> Object Detection -> vehicles-OpenImages



# Custom Dataset으로 YOLOv5 학습

## ◆ Training Dataset 준비

- 참고자료: <https://sguys99.github.io/ds01>
- roboflow (<https://public.roboflow.com/object-detection>)  
-> Classification -> fruits





# YOLO\_mark를 사용한 training data 생성

## ◆ YOLO\_mark

- [https://github.com/AlexeyAB/Yolo\\_mark](https://github.com/AlexeyAB/Yolo_mark)

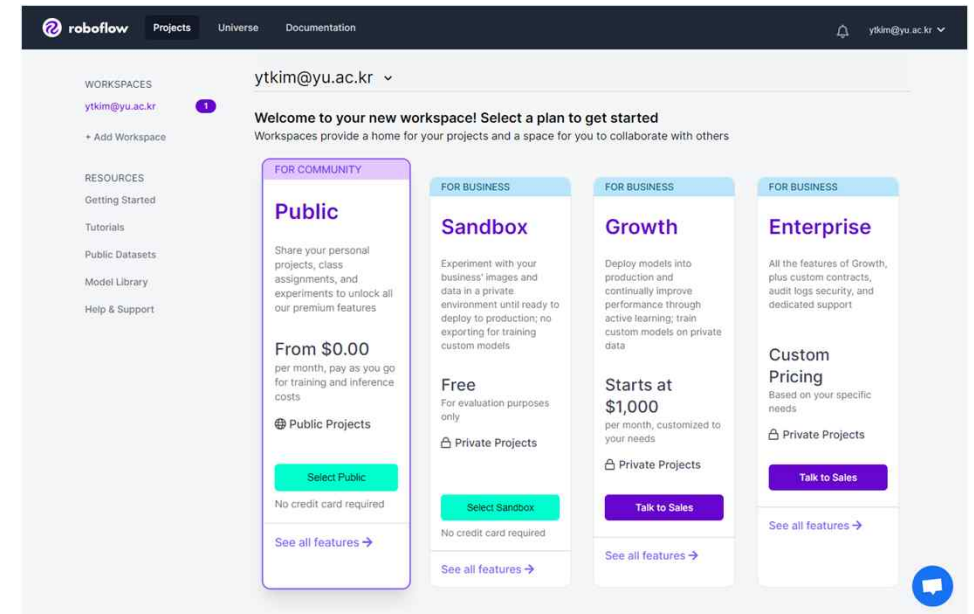
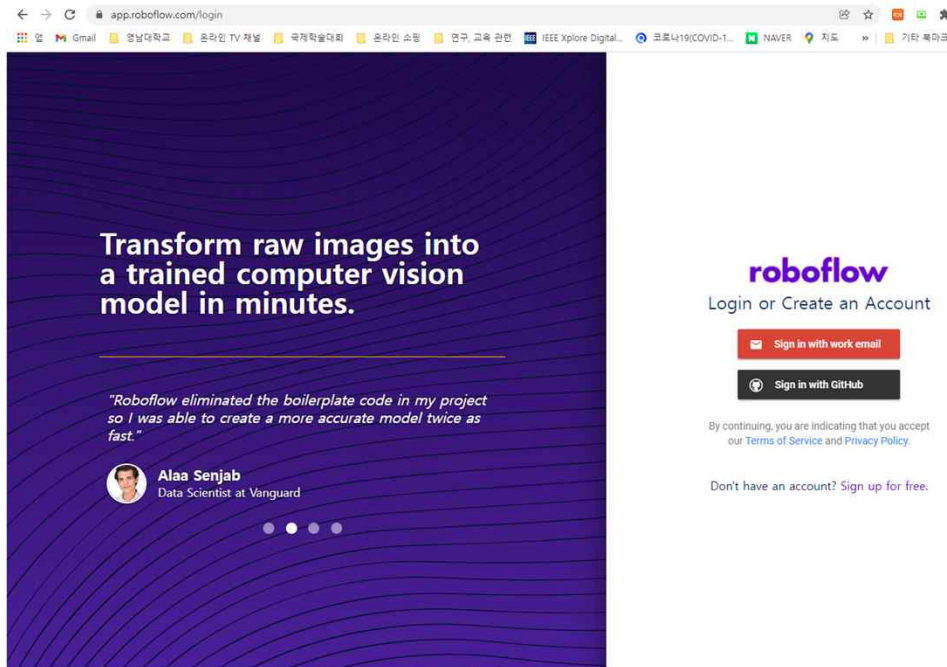




# roboflow를 사용한 custom train data 준비

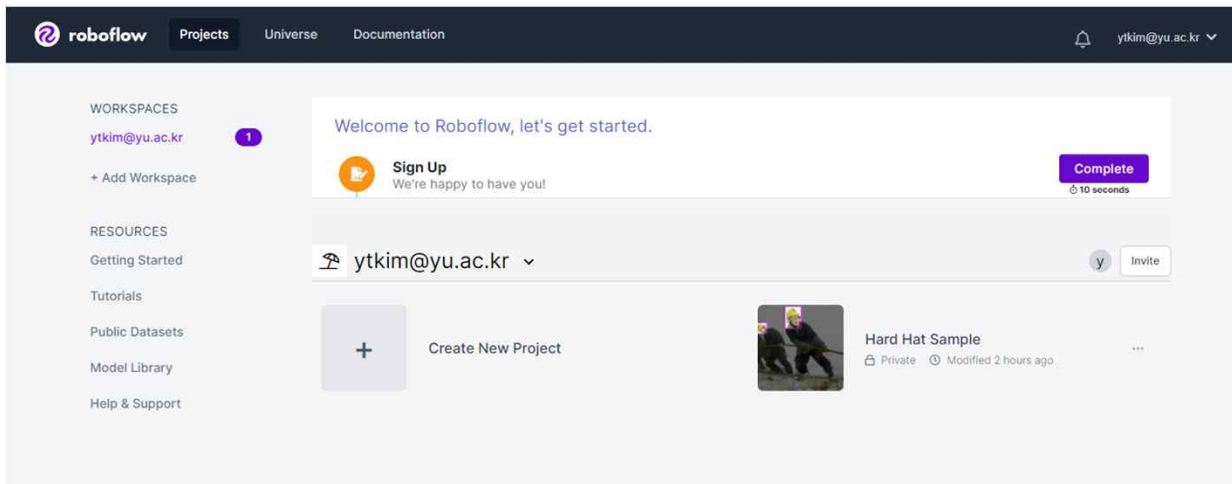
## ◆ roboflow

- <https://roboflow.com/?ref=ultralytics>



# roboflow - Create New Project

## ◆ Create New Project in roboflow



×

Create Project

ytkim@yu.ac.kr / 🔒 New Private Project

Project Name

traffic signs

Project Type

Object Detection (Bounding Box)

Annotation Group ?

letters

Cancel

Create Private Project

# yaml 파일 준비

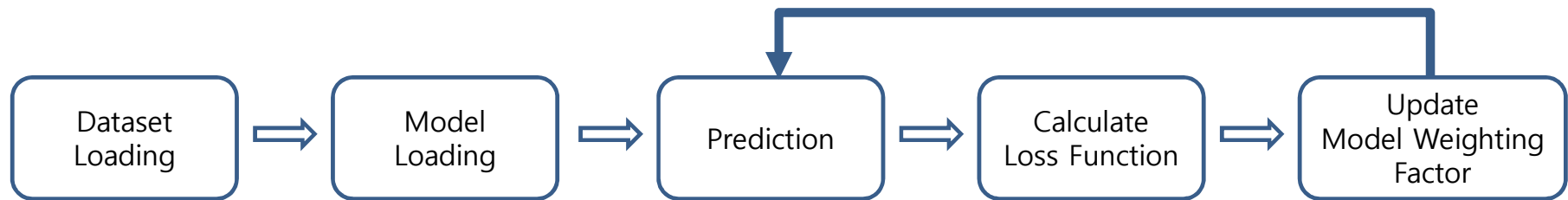
## ◆ dataset.yaml

- yaml 파일: 학습데이터의 경로, 클래스 개수 및 종류 설정
- train : 학습데이터의 경로
- val : 검증 데이터 경로
- nc : 학습할 클래스 개수
- names : 학습할 클래스 이름 들

```
C: > YOLOv5 > yolov5-master > yolov5-master > data > Vehicles > ! data.yaml
1  train: ../train/images
2  val: ../valid/images
3
4  nc: 5
5  names: ['Ambulance', 'Bus', 'Car', 'Motorcycle', 'Truck']
6
```

# 모델 학습

## ◆ 모델 학습 순서



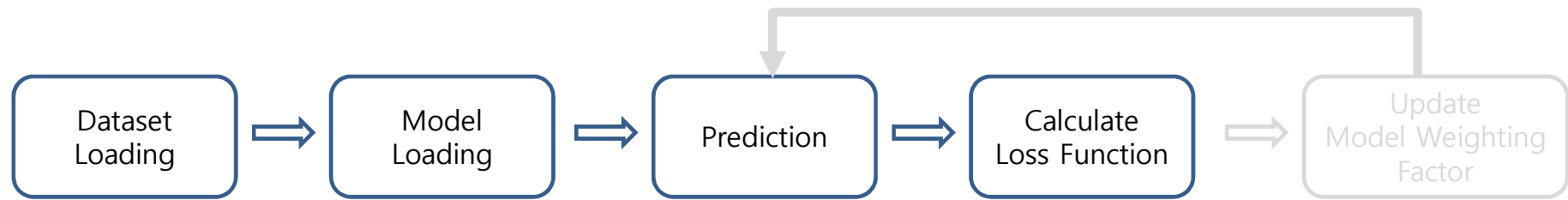
## ◆ train.py

```
> python train.py --data "data/dataset.yaml" --epochs 100
```

## 학습된 모델의 검증

### ◆ 학습된 모델의 검증 절차

- 모델 학습 절차에서 모델 가중치 업데이터터 과정 생략

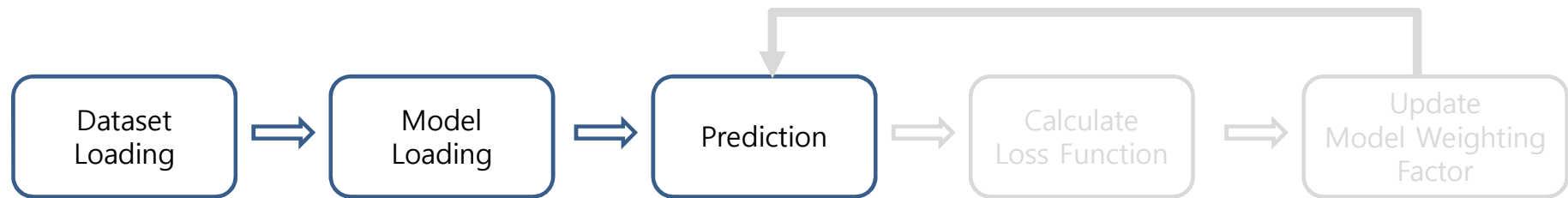


### ◆ val.py

```
> python val.py --data "data/dataset.yaml"
  --weights "C:\YOLOv5\yolov5-master\yolov5-master
    \run\train\exp\weights\best.pt"
```

## 학습된 모델로 예측하기

### ◆ 학습된 모델을 사용한 예측과정



### ◆ detect.py

> python detect.py --source 0

## **Jetson Nano에 YOLOv5의 설치 및 객체 탐지**

# Install YOLO V5 on Jetpack(1)

## ◆ gdm3 제거 및 lightdm 설치

- Jetson Nano는 desktop manager 환경에서 YOLOv5를 실행시키면 램 사용량 증가
- RAM 사용량을 최대한 줄이기 위해서 desktop manager를 lightdm으로 사용
- sudo apt-get install lightdm
- sudo apt-get purge gdm3

## ◆ Shell 파일을 사용한 OpenCV 4.5.4 with CUDA 설치

- wget <https://github.com/Qengineering/Install-OpenCV-Jetson-Nano/raw/main/OpenCV-4-5-4.sh>
- sudo chmod 755 ./OpenCV-4-5-4.sh
- ./OpenCV-4-5-4.sh





# Install YOLO V5 on Jetpack(2)

## ◆ PyTorch 1.8 + torchvision v0.9.0

# PyTorch 1.8.0 다운로드 및 dependencies 설치

- wget <https://nvidia.box.com/shared/static/p57jwntv436lfrd78inwl7iml6p13fzh.whl> -O torch-1.8.0-cp36-cp36m-linux\_aarch64.whl

- sudo apt-get install python3-pip libopenblas-base libopenmpi-dev

# Cython, numpy, pytorch 설치

- pip3 install Cython
- pip3 install numpy torch-1.8.0-cp36-cp36m-linux\_aarch64.whl

# torchvision dependencies 설치

- sudo apt-get install libjpeg-dev zlib1g-dev libpython3-dev libavcodec-dev libavformat-dev libswscale-dev
- git clone --branch v0.9.0 https://github.com/pytorch/vision torchvision
- cd torchvision
- export BUILD\_VERSION=0.9.0
- python3 setup.py install --user
- cd ../ # attempting to load torchvision from build dir will result in import error

# Install YOLO V5 on Jetpack(3)

## ◆ Install YOLO V5 on Jetpack

- `cd ~`
- `git clone https://github.com/ultralytics/yolov5`
- `cd yolov5`

# yolov5s.pt weight 다운로드

- `wget https://github.com/ultralytics/yolov5/releases/download/v6.0/yolov5s.pt`

# requirements.txt에서 아래 내용 제거

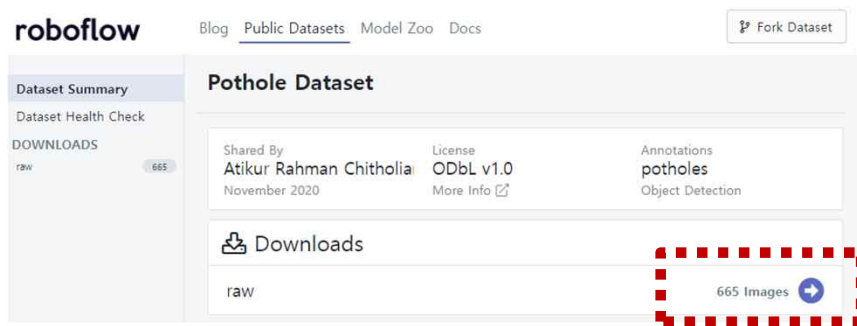
```
numpy>=1.18.5  
opencv-python>=4.1.2  
torch>=1.7.0  
torchvision>=0.8.1
```

- `python3 -m pip install --upgrade pip`
- `python3 -m pip install -r requirements.txt`

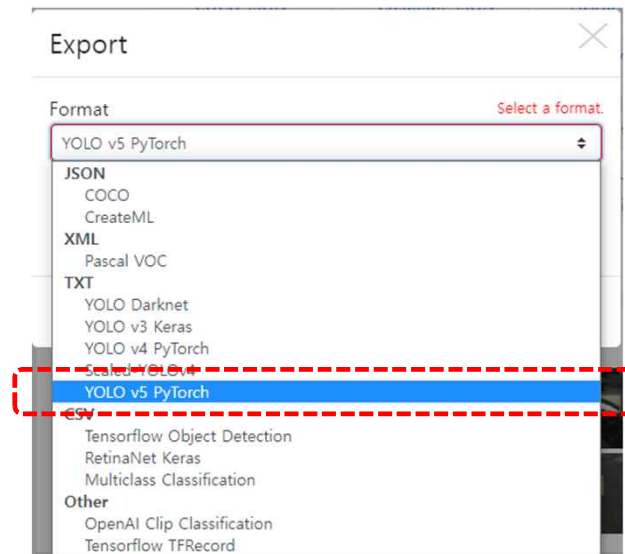
# Generate Learning Model about Detecting Pothole

## ◆ Pothole DataSet

참고: <https://public.roboflow.com/object-detection/pothole>



(a) Pothole Dataset



(b) Export Dataset  
in YOLOv5 PyTorch Format

# Generate Learning Model about Detecting Pothole(1)

## ◆ Pothole DataSet

Your Download Code

Jupyter Terminal Raw URL

Paste this snippet into [a notebook from our model library](#) to download and unzip your dataset:

```
!curl -L "https://public.roboflow.com/ds/g6xhatFbYB?key=RCc6KT5G2o" > roboflow.zip;
unzip roboflow.zip; rm roboflow.zip
```

**Warning:** Do not share this link beyond your team, it contains a private key that is tied to your Roboflow account. Acceptable use policy applies.

Done Choose a Model

colab을 이용해 학습 환경을 만든 후 코드 삽입

+ 코드 + 텍스트

```
1 !curl -L "https://public.roboflow.com/ds/g6xhatFbYB?key=RCc6KT5G2o" > roboflow.zip; unzip roboflow.zip; rm roboflow.zip
2
3 extracting: valid/labels/img-457.jpg.rf.11ec4d1f8537d717069f83c677b00c1114.txt
4 extracting: valid/labels/img-468.jpg.rf.2b4c5e14c8c1bb5f760bb238992e3929.txt
5 extracting: valid/labels/img-469.jpg.rf.a0f424db1106cf40f7cbbf56696ab5ba.txt
6 extracting: valid/labels/img-46.jpg.rf.b479d9e82ab3f93117826289011004a9.txt
7 extracting: valid/labels/img-474.jpg.rf.087e72d05f12bbc01d250db1e2e8343b.txt
8 extracting: valid/labels/img-485.jpg.rf.7b6a34e38d89caf1f8d97f24298e67b2.txt
9 extracting: valid/labels/img-490.jpg.rf.32377cf7b8ee68f6b7faf955aa30ab7b.txt
10 extracting: valid/labels/img-491.jpg.rf.b6979774778537b494872c0a743070d6.txt
11 extracting: valid/labels/img-492.jpg.rf.9e7685d521951604f29fe49175c0040e.txt
12 extracting: valid/labels/img-493.jpg.rf.bb58a09287078393b132cd2b68fe6f29.txt
13 extracting: valid/labels/img-495.jpg.rf.3f61bcb802633686f9ce58fe78841711.txt
```

# Generate Learning Model about Detecting Pothole(2)

## ◆ Pothole DataSet

※ **colab** 환경에서 해당 코드 삽입

### (1) YOLO V5 cnn 모델

```
%cd /content  
!git clone https://github.com/ultralytics/yolov5.git
```

### (2) requirement.txt install

```
%cd /content/yolov5  
!pip install -r requirements.txt
```

### (3) train data set, validation data set list 작성

```
with open('/content/dataset/train.txt', 'w') as f:  
    f.write('\n'.join(train_img_list) + '\n')  
  
with open('/content/dataset/val.txt', 'w') as f:  
    f.write('\n'.join(val_img_list) + '\n')
```

# Generate Learning Model about Detecting Pothole(3)

## ◆ Pothole DataSet

※colab 환경에서 해당 코드 삽입

### (4) yaml 파일에 image 삽입

```
import yaml

with open('/content/dataset/data.yaml', 'r') as f:
    data = yaml.load(f)
print(data)
data['train'] = '/content/dataset/train.txt'
data['val'] = '/content/dataset/val.txt'
with open('/content/dataset/data.yaml', 'w') as f:
    yaml.dump(data, f)
print(data)
```

### (5) image 학습

```
%cd /content/yolov5/

!python train.py --img 416 --batch 16 --epochs 50 --data /content/dataset/data.yaml
--cfg ./models/yolov5s.yaml --weights yolov5s.pt --name gun_yolov5s_results
```



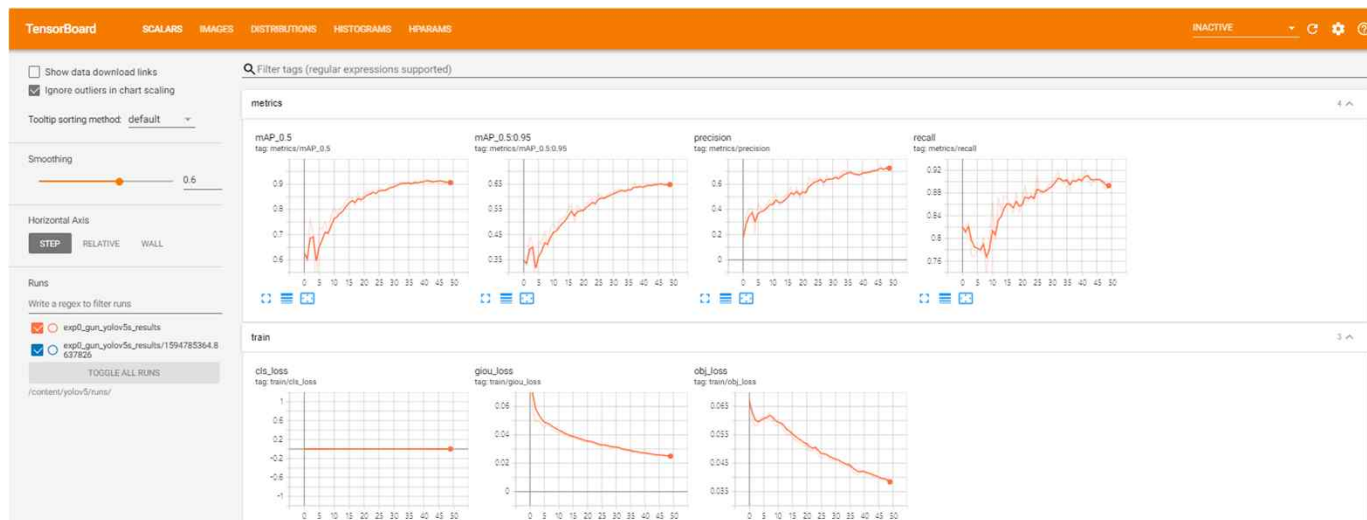
# Generate Learning Model about Detecting Pothole(4)

## ◆ Pothole DataSet

※colab 환경에서 해당 코드 삽입

(6) 학습 결과 확인

```
%load_ext tensorboard  
%tensorboard --logdir /content/yolov5/runs/
```



# Detecting Pothole on Jetson(1)

## ◆ 앞서 학습한 모델을 Jetson으로 이동

- ✓ 생성된 모델을 옮겨온 모습

```
antl@antl-desktop:~/yolov5/iot$ ls  
best.pt  README.md  video
```

- ✓ YOLO V5를 install하면 detect.py 생성됨

```
antl@antl-desktop:~/yolov5$ ls  
'=1.18.5'  detect.org.py  export.py  LICENSE  requirements.txt  setup.cfg  utils  
CONTRIBUTING.md  detect.py  hubconf.py  models  result  train.py  val.py  
data  Dockerfile  iot  README.md  runs  tutorial.ipynb  yolov5s.pt
```

- ✓ 학습된 모델을 이용해 detect.py를 실행
  - sudo python3 detect.py --source 0 --weights iot/best.pt

--source: image, video, webcam 중에서 detect에 사용할 방법 선택  
(0: webcam, image나 video를 사용하기 위해서는 해당 파일명 입력)

--weights: detect에 사용할 cnn model 선택



## Detecting Pothole on Jetson(2)

### ◆ 실행 시 발생할 수 있는 오류

- ✓ 같은 **process**에서 두 가지 **version**의 **GTK**를 동시에 사용하여 발생하는 문제

```
(detect_org.py:10112): Gtk-ERROR **: 21:14:26.767: GTK+ 2.x symbols detected. Using GTK+ 2.x and GTK+ 3 in the same process is not supported
Trace/breakpoint trap
antl@antl-desktop:~/yolov5$
```

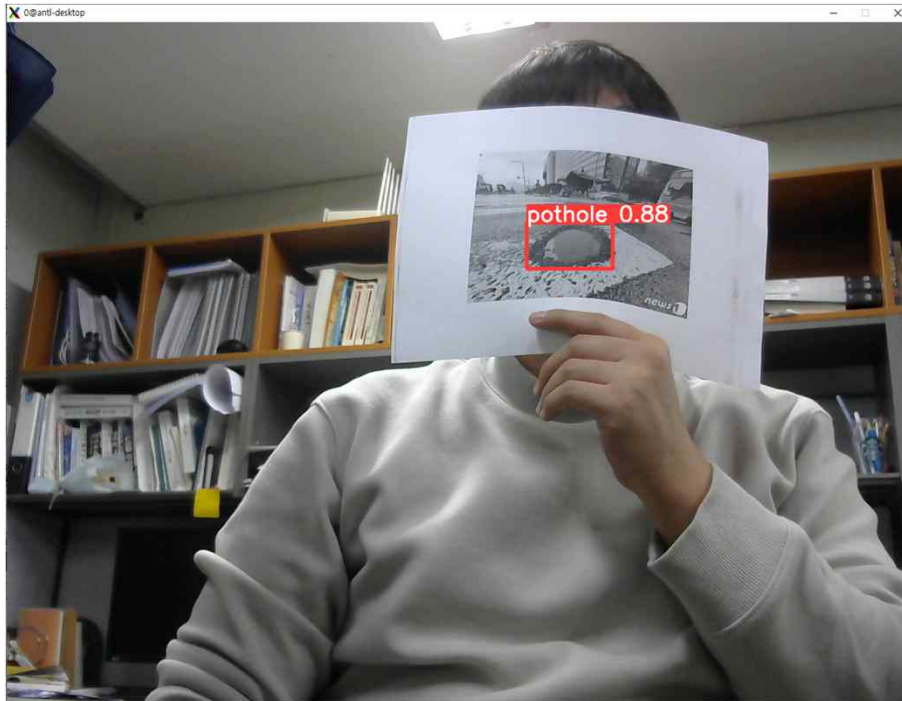
- ✓ **detect** 코드 내에서 하나의 **GTK**를 사용하도록 **2.0 version** 선언

```
import argparse
import os
import sys
from pathlib import Path
import cv2
import torch
import torch.backends.cudnn as cudnn
import gi

gi.require_version('Gtk', '2.0')
FILE = Path(__file__).resolve()
ROOT = FILE.parents[0] # YOLOv5 root directory
if str(ROOT) not in sys.path:
    sys.path.append(str(ROOT)) # add ROOT to PATH
ROOT = Path(os.path.relpath(ROOT, Path.cwd())) # relative
```

## Detecting Pothole on Jetson(3)

### ◆ 실행 결과



## **Homework 18**

# Homework 18

## 18.1 Desktop PC 또는 Notebook에 YOLOv5를 설치 및 Webcam 영상 인식 기능 실험

- 강의 자료를 참고하여 desktop PC 또는 notebook에 YOLOv5를 설치하라
- 설치된 YOLOv5를 사용하여 Webcam에 입력되는 동영상을 실시간으로 인식하는 기능을 구성하고, 실행 결과를 제출하라.

## 18.2 YOLOv5를 사용하여 자동차 번호판 인식 기능 구현

- 설치된 YOLOv5를 사용하여 자동차 번호판의 사진을 자동으로 인식하는 기능을 구현하고, 실행 결과를 제출하라.

## 18.3 YOLOv5를 사용하여 도로 교통 표지판 인식 기능 구현

- 설치된 YOLOv5를 사용하여 동영상에 포함된 도로 교통 표지판 인식 기능을 구현하고, 실행 결과를 제출하라.



# References

## <Deep learning, 필기체 인식>

- [1] [https://www.tensorflow.org/guide/keras/sequential\\_model?hl=ko](https://www.tensorflow.org/guide/keras/sequential_model?hl=ko)
- [2] <https://qkqhxa1.tistory.com/987>.
- [3] What is neural network, <https://www.youtube.com/watch?v=aircAruvnKk>.

## <YOLO>

- [1] You Only Look Once: Unified, Real-Time Object Detection, <https://www.youtube.com/watch?v=NM6lrxy0bxs>.
- [2] 객체 탐지 Object Detection - YOLO, <https://www.youtube.com/watch?v=5ev0MMBzY3E>.
- [3] YOLO 논문 리뷰, <https://www.youtube.com/watch?v=O78V3kwBRBk>.
- [4] YOLOv5 - training & test, <https://bigdata-analyst.tistory.com/195>
- [5] Jetson Nano에서 YOLOv5 사용 전 준비할 것들, <https://whiteknight3672.tistory.com/313>.
- [6] CUDA VERSION확인, <https://ghostweb.tistory.com/829>
- [7] OpenCV VERSION 확인, <https://jstar0525.tistory.com/145?category=1018133>
- [8] cuDNN VERSION 확인, <https://junk-research-note.tistory.com/5>
- [9] YOLO V5, <https://whiteknight3672.tistory.com/313>

