2-1. YOLO

주제	
0. Introduction	강의 커리큘럼 소개
1. Face Recognition	1-1. Face Recognition 이론 소개
	1-2. Face Detection - 대표 모델 및 코드 소개
	1-3. [실습1] Dlib 및 Retina Face 코드 구현
	1-4. Face Alignment - 대표 모델 및 코드 소개
	1-5. [실습2] 황금비율 계산
	1-6. Face Recognition - 대표 모델 및 코드 소개
	1-7. [실습3] 그룹 가수 사진에서 각각 멤버 인식하기
2. Object Detection	2-1. Object Detection 이론 소개
	2-2. 대표 모델 – Yolov8 소개
	2-3. [실습1] 마스크 착용 유무 프로젝트
	2-4. [실습2] Tensor-RT 기반의 Yolov8, 표지판 신호등 검출
	2-5. 대표 모델 - Complex-Yolov4
	2-6. [실습3] Lidar Data 기반의 차량 Detection

YOLOv8

ultralytics
YOLOv8

DOWNLOAD THE APP

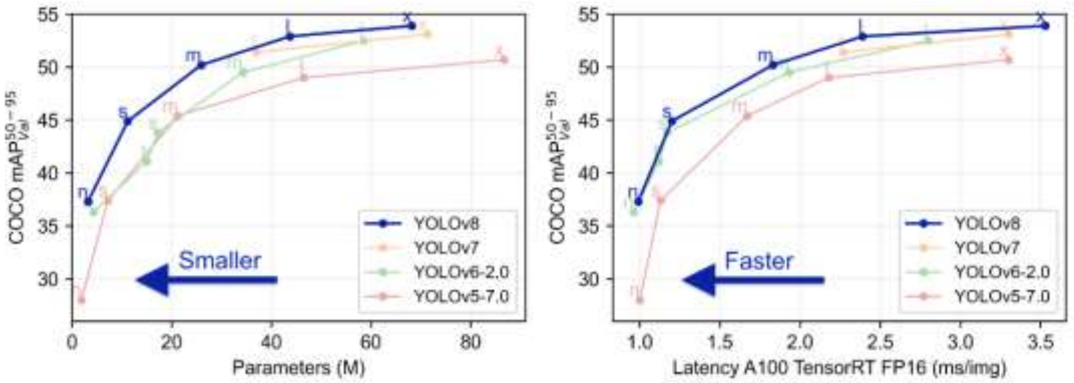
English | 简体中文

Ultralytics CI passing | codecov 87% | DOI 10.5281/zenodo.7347926 | docker pulls 23k
Run on Gradient | Open in Colab | Open in Kaggle

Ultralytics YOLOv8 is a cutting-edge, state-of-the-art (SOTA) model that builds upon the success of previous YOLO versions and introduces new features and improvements to further boost performance and flexibility. YOLOv8 is designed to be fast, accurate, and easy to use, making it an excellent choice for a wide range of object detection and tracking, instance segmentation, image classification and pose estimation tasks.

We hope that the resources here will help you get the most out of YOLOv8. Please browse the YOLOv8 Docs for details, raise an issue on GitHub for support, and join our Discord community for questions and discussions!

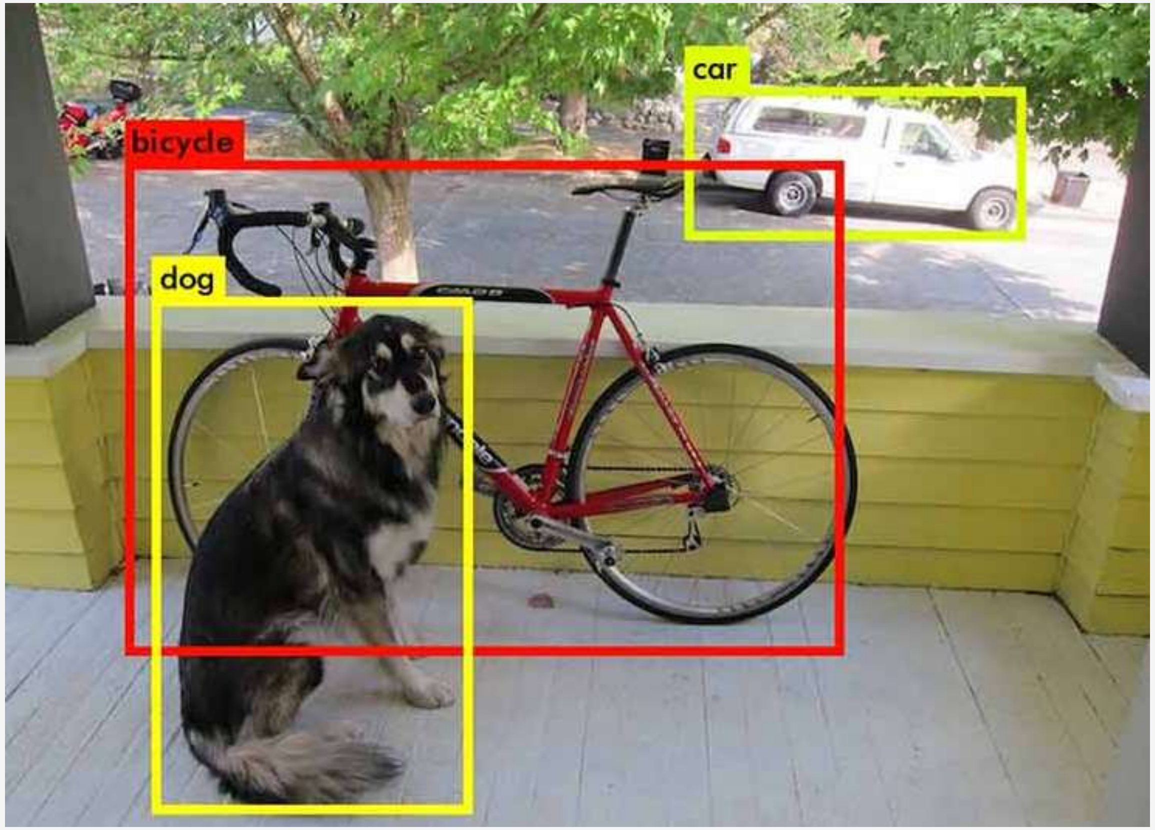
To request an Enterprise License please complete the form at Ultralytics Licensing.



References
https://github.com/ultralytics/ultralytics

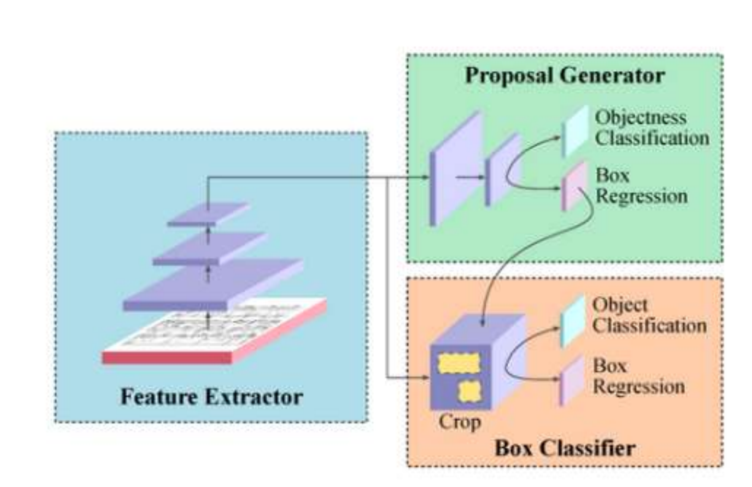
Object Detection이란?

이미지 내의 모든 Object에 대하여 Classification와 Localization을 수행

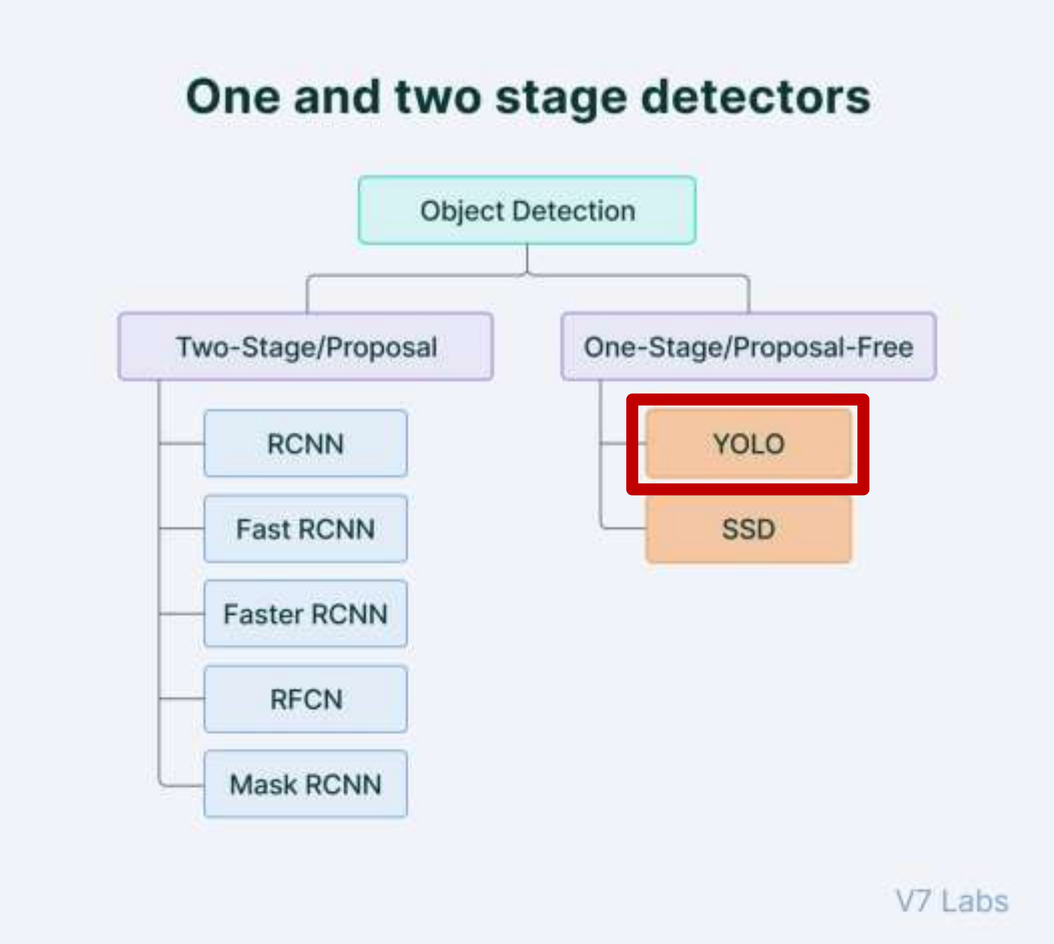


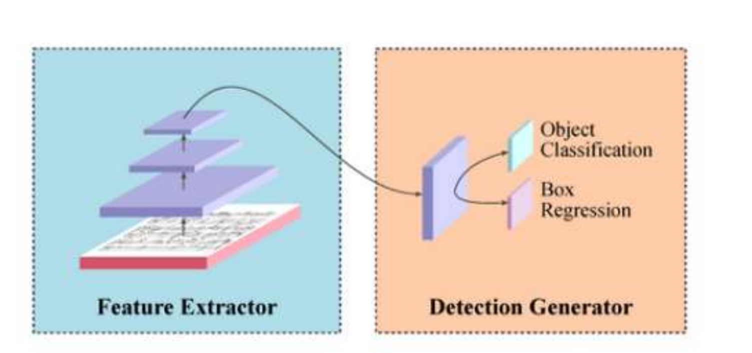
References
https://machinethink.net/blog/object-detection-with-yolo/

One-Stage Detector VS Two-Stage Detector



(b) Basic architecture of a two-stage detector.





(a) Basic architecture of a one-stage detector.

References
(Middle) https://www.v7labs.com/blog/yolo-object-detection
(Left, Right)https://gaussian37.github.io/vision-detection-table

CONTENT

01

YOLO

02

YOLOv5

03

YOLOv8

04

Experimental Results

05

Conclusion

YOLO

YOLO (You Only Look Once)

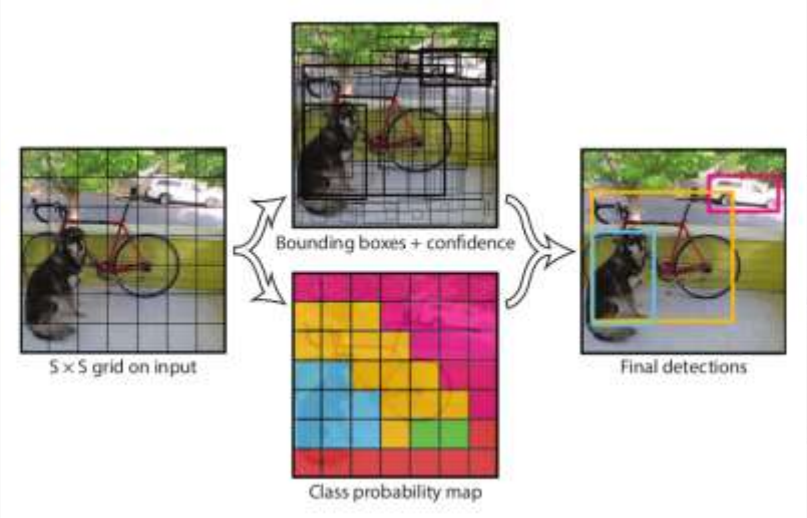
Central **real-time** object detection system for robotics, driverless cars, and video monitoring applications

End-to-End Network / One-stage object detection

Object Detection 문제를 **regression문제**로 정의하는 것을 통해 bounding box 좌표 및 각 클래스일 확률을 계산

YOLO의 장점

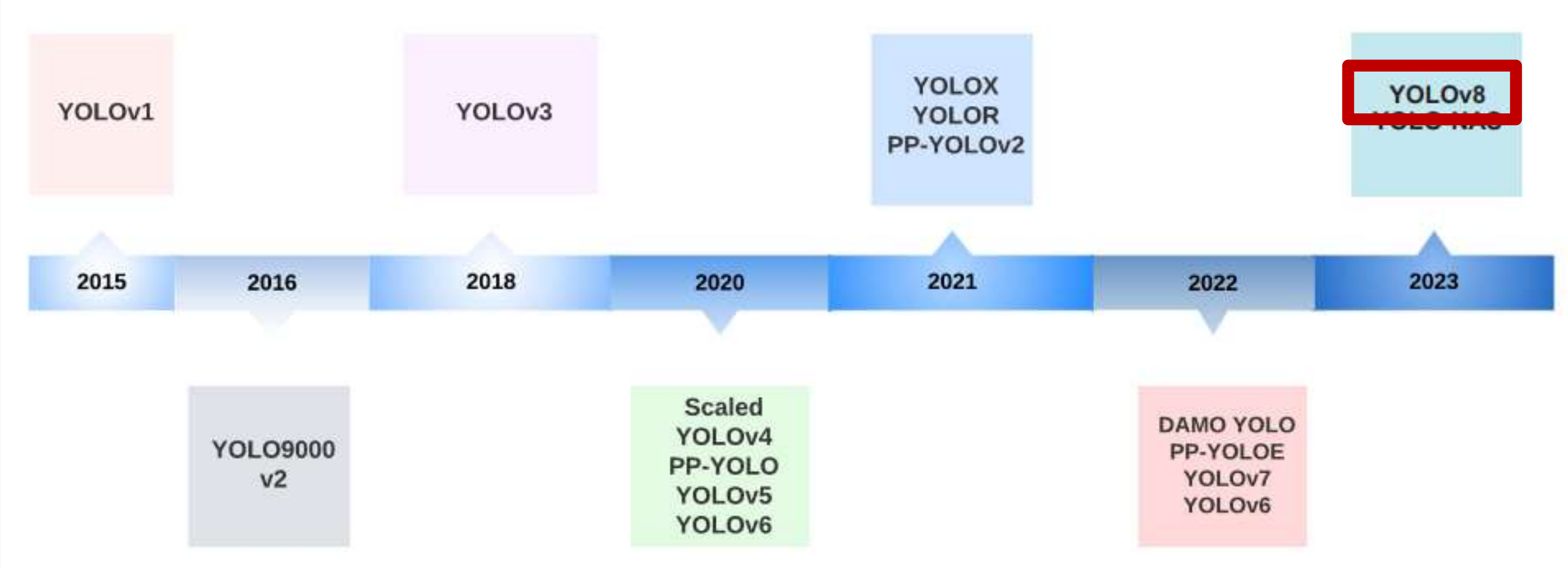
- Sliding Window 방식이 아닌 CNN을 사용하여 이미지 전역의 **Contextual information**을 얻어 학습 성능을 높임
- 일반적인 Object의 표현을 학습하기에 **Domain이 달라**도 높은 성능을 보임



References
(Left) https://pjreddie.com/darknet/yolo/
(Right) https://arxiv.org/pdf/2304.00501.pdf

History of YOLO

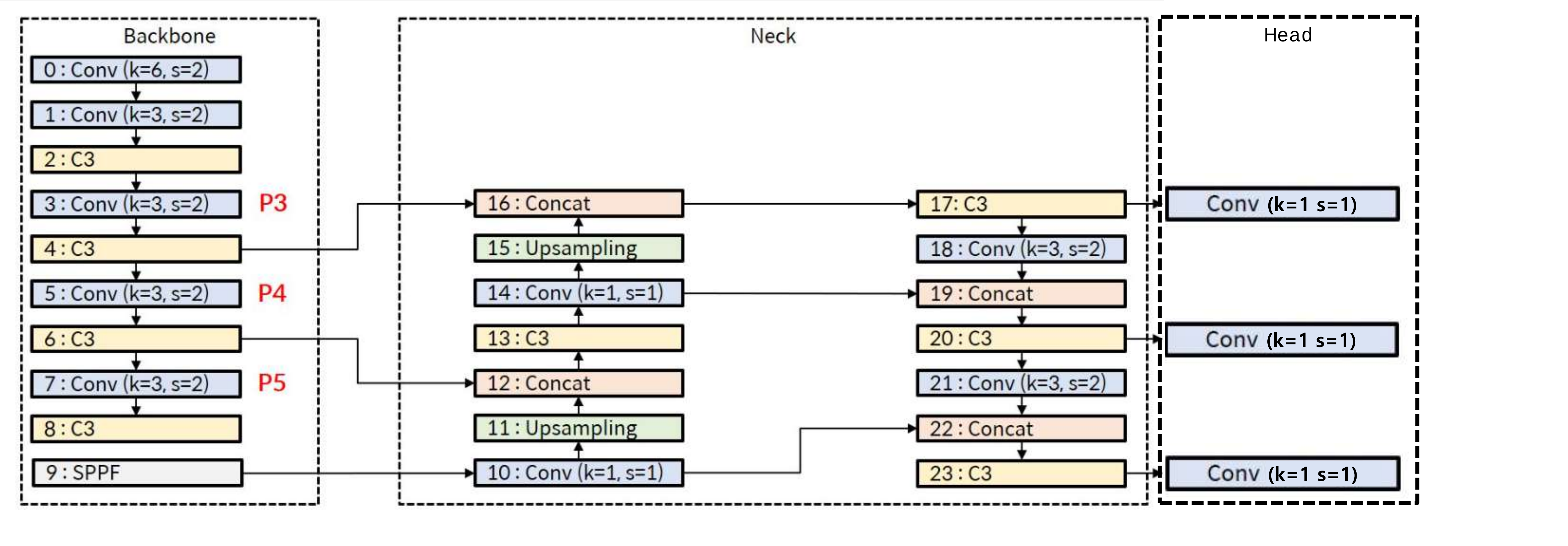
- **YOLOv1 :** 24 CNN + 2FC / leaky ReLU
- **YOLOv2 :** Darknet-19, Batch Normalization , Anchor boxes, Multi-scale training
- **YOLOv3 :** Efficient backbone, Spatial pyramid pooling
- **YOLOv4 :** Mosaic data augmentation, anchor-free detection head
- **YOLOv5 :** Modified CSPDarknet53 backbone, SPPF, Several augmentations, Five scaled versions, SiLU
- **YOLOv6 :** RepVGG backbone, Self-distillation, VariFocal & SIoU & GIoU, Quantization-scheme
- **YOLOv7 :** Without pre-trained backbones, Additional task (Pose estimation)



References
https://arxiv.org/pdf/2304.00501.pdf

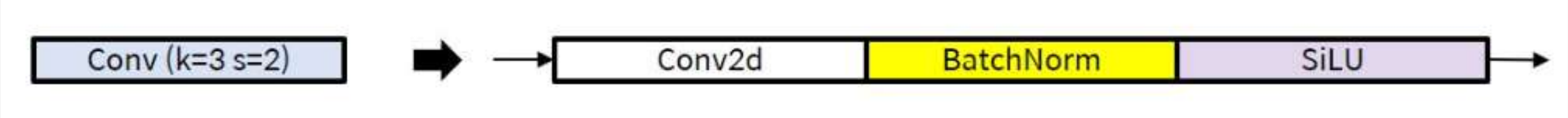
YOLOv5

Architecture



References
https://epozen-dt.github.io/Yolov5/

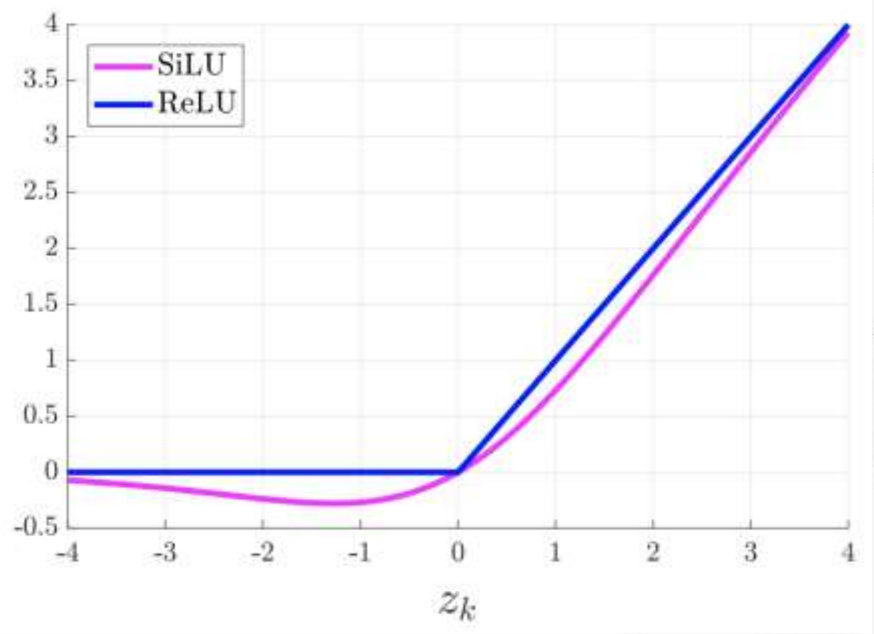
Convolution Block



SiLU (Sigmoid Linear Unit) : Swish activation function

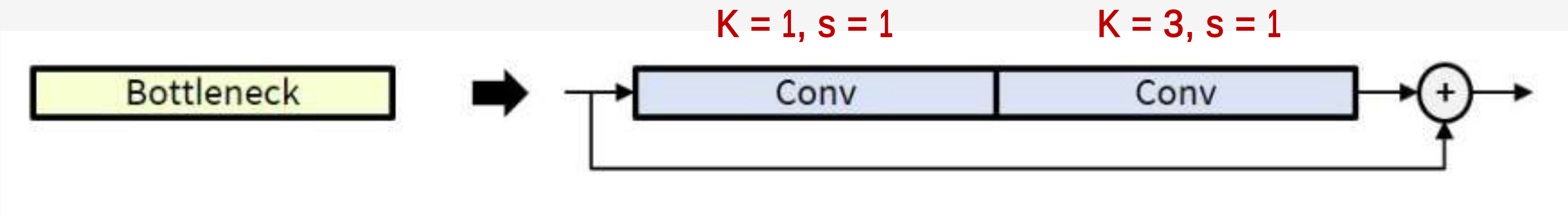
- Unbounded above where x>= 0
- Bounded below where x<0
- Non monotonicity
- Smooth figure

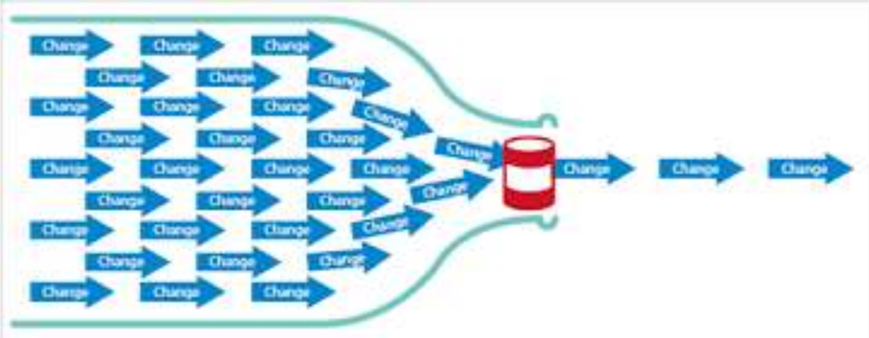
$$SiLU(x) = x\left(\frac{1}{1+e^{-x}}\right)$$

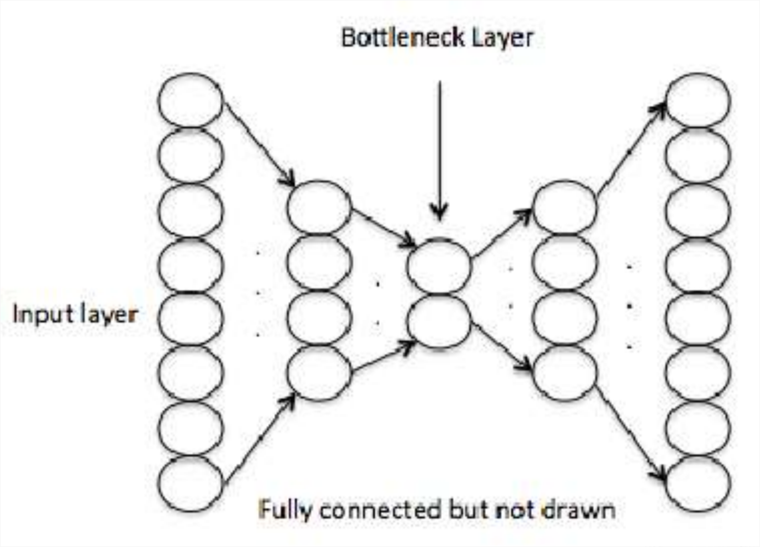


References
(Top) https://epozen-dt.github.io/Yolov5/
(Bottom) https://paperswithcode.com/method/silu

Bottleneck

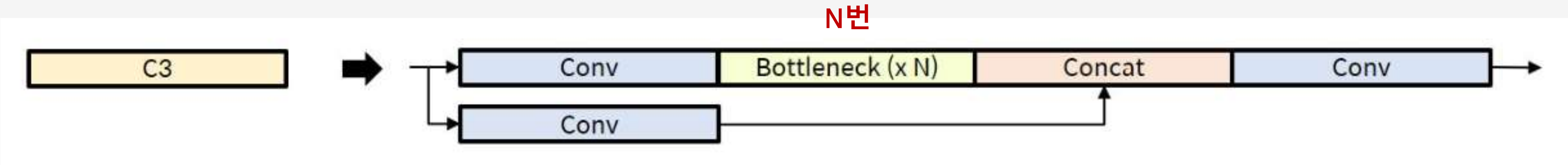






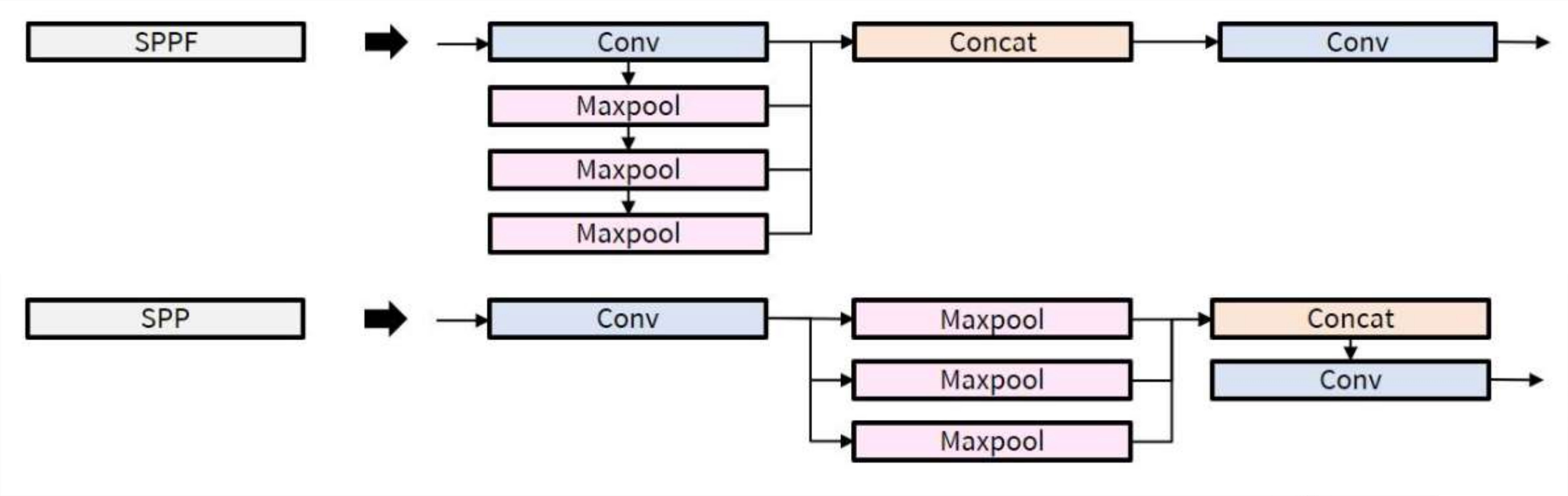
References
(Top) https://epozen-dt.github.io/Yolov5/
(Bottom-Left) https://nearhome.tistory.com/129
(Bottom-Right) https://www.researchgate.net/figure/Visualization-of-a-bottleneck-architecture_fig1_282859516

C3



References
https://epozen-dt.github.io/Yolov5/

SPPF (Spatial Pyramid Pooling – Fast) vs SPP



References
https://epozen-dt.github.io/Yolov5/

SPP (Spatial Pyramid Pooling)

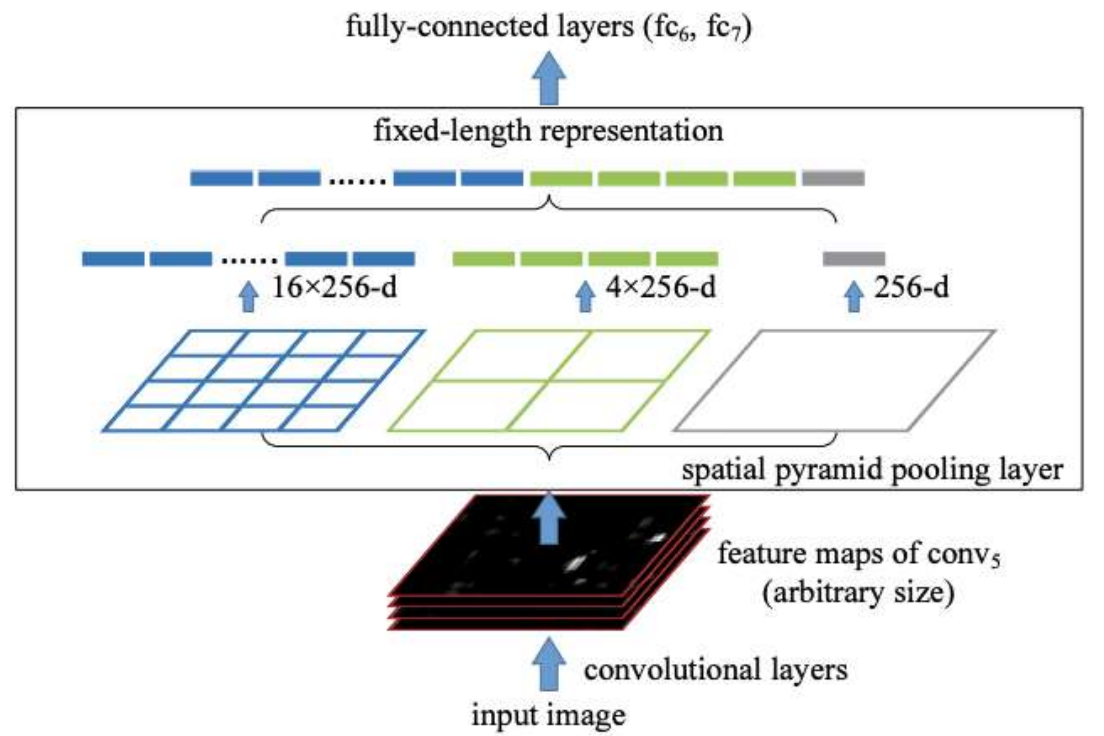
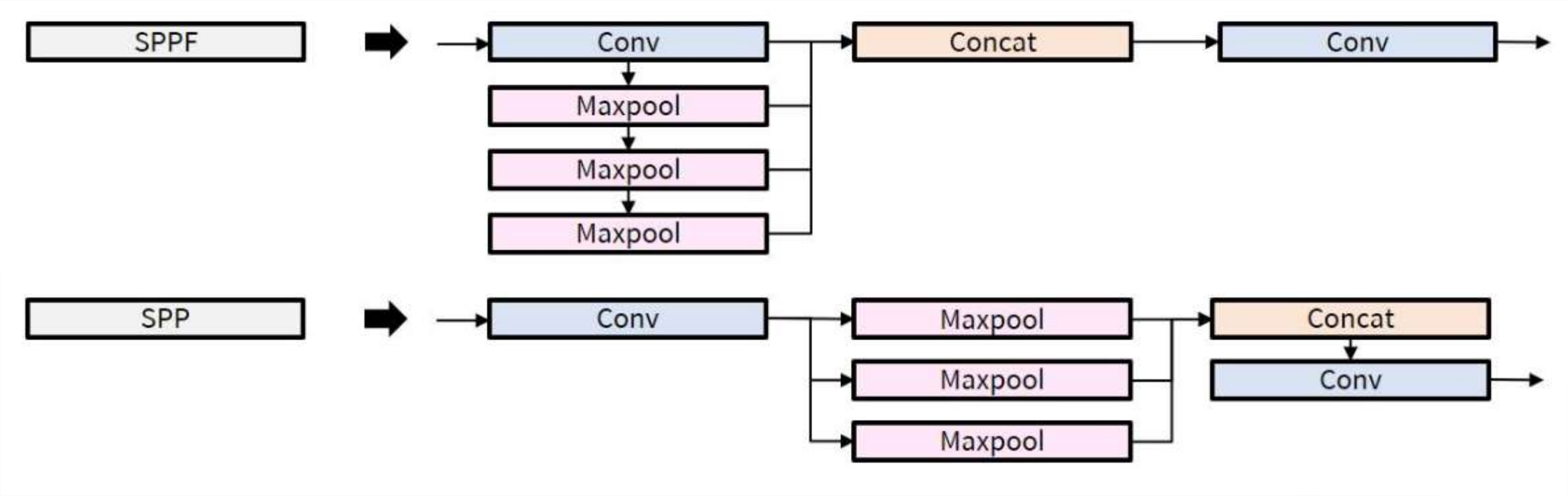


Figure 3: A network structure with a **spatial pyramid pooling layer**. Here 256 is the filter number of the $conv_5$ layer, and $conv_5$ is the last convolutional layer.

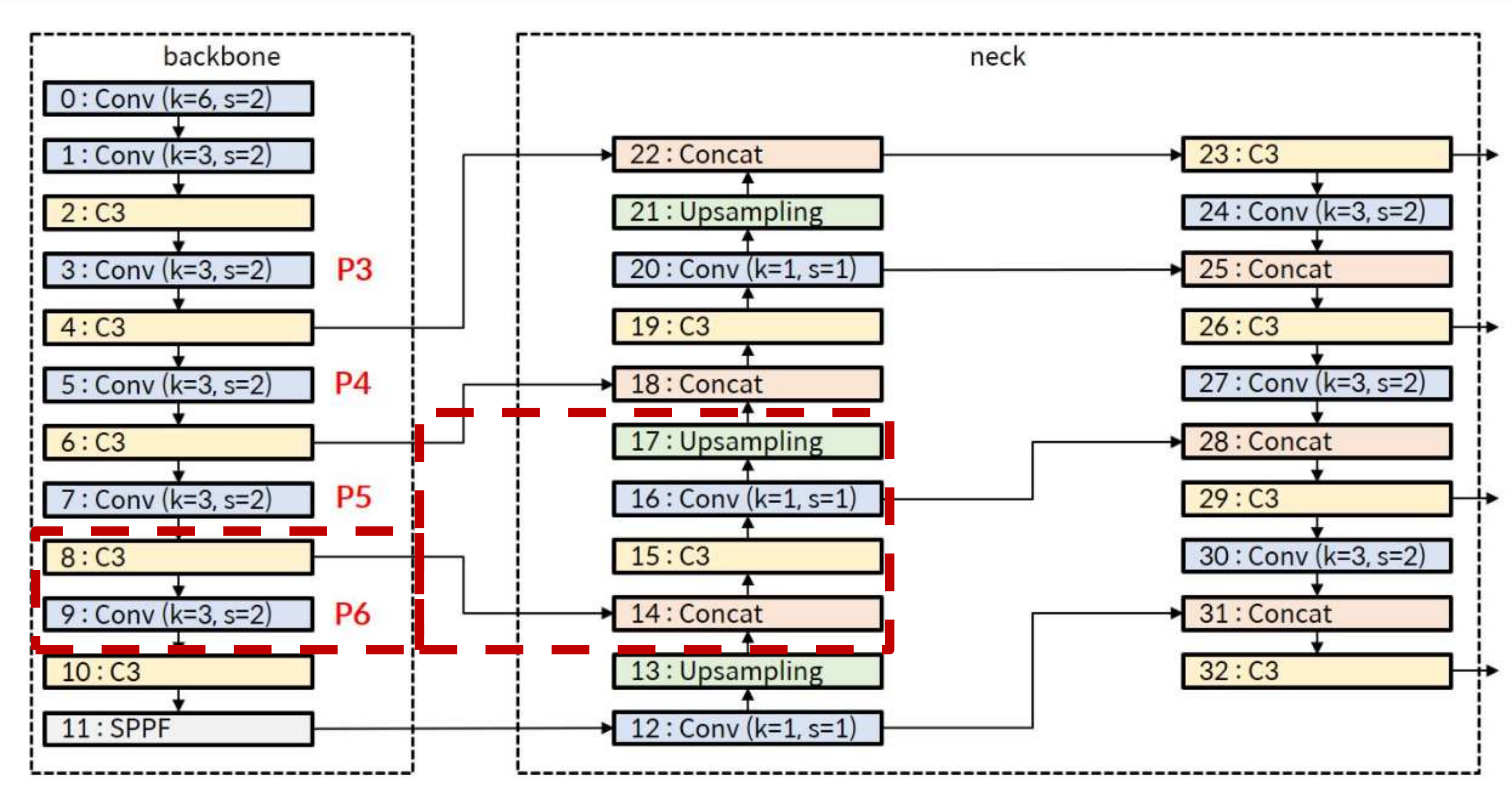
References
https://epozen-dt.github.io/Yolov5/
https://paperswithcode.com/method/spatial-pyramid-pooling

SPPF (Spatial Pyramid Pooling – Fast) vs SPP



References
https://epozen-dt.github.io/Yolov5/

Architecture (YOLOv5x6)



References
https://epozen-dt.github.io/Yolov5/

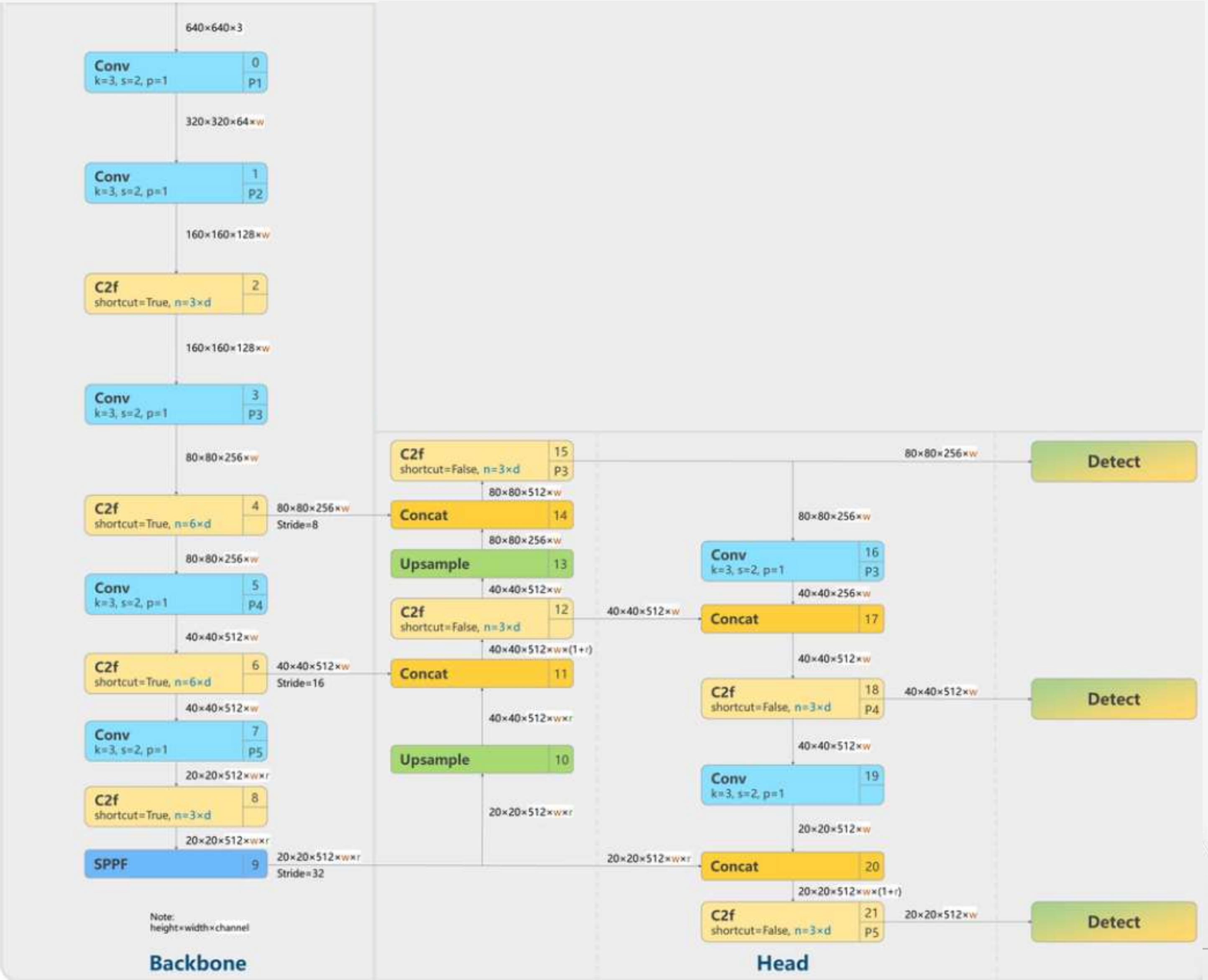
YOLOv8

YOLOv8 vs YOLOv5

- Replace the **C3** module with the **C2f** module
- Replace the first 6x6 Conv with 3x3 Conv in the Backbone
- **Delete two Convs** (No.10 and No.14 in the YOLOv5 config)
- Replace the first **1x1 Conv** with **3x3 Conv** in the Bottleneck
- Use **decoupled head** and delete the objectness branch
- Anchor-free model
- Modified Mosaic Augmentation

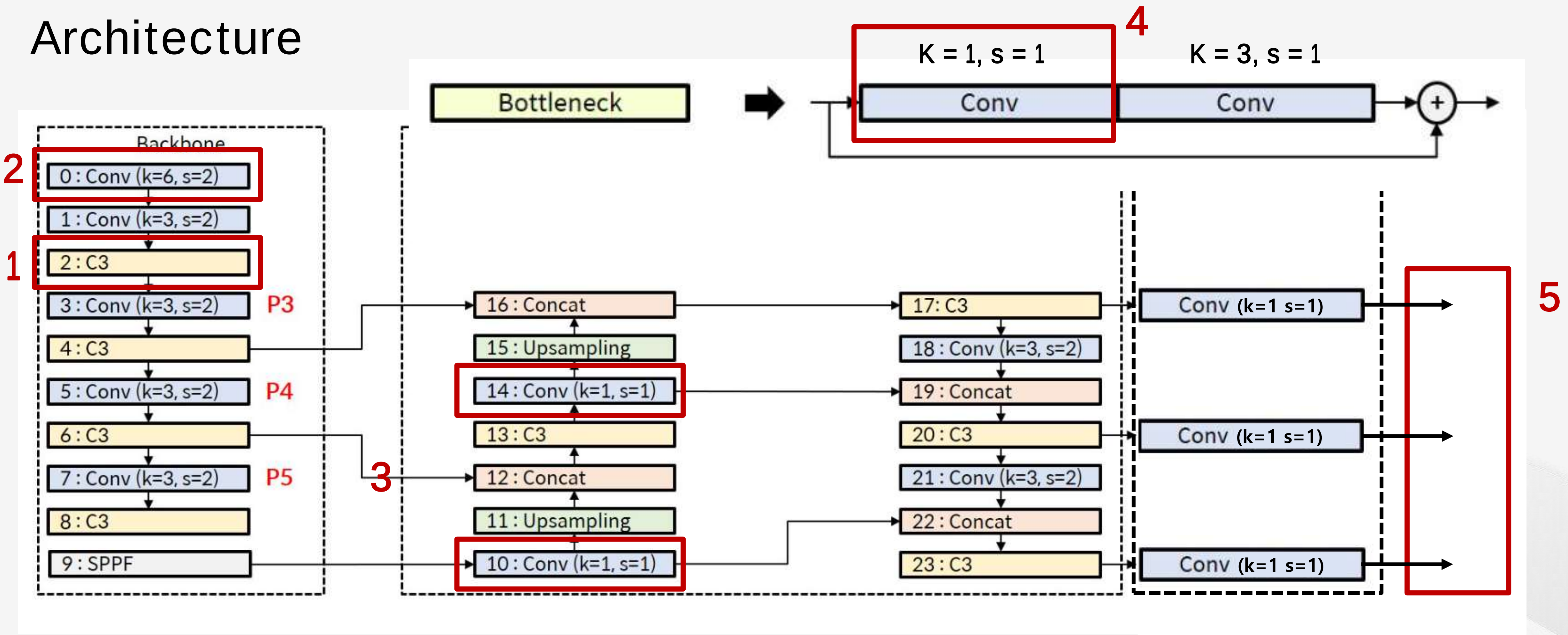
References
https://m.blog.naver.com/PostView.naver?blogId=skfnsid123&logNo=223000302805&categoryNo=21&proxyReferer=

Architecture



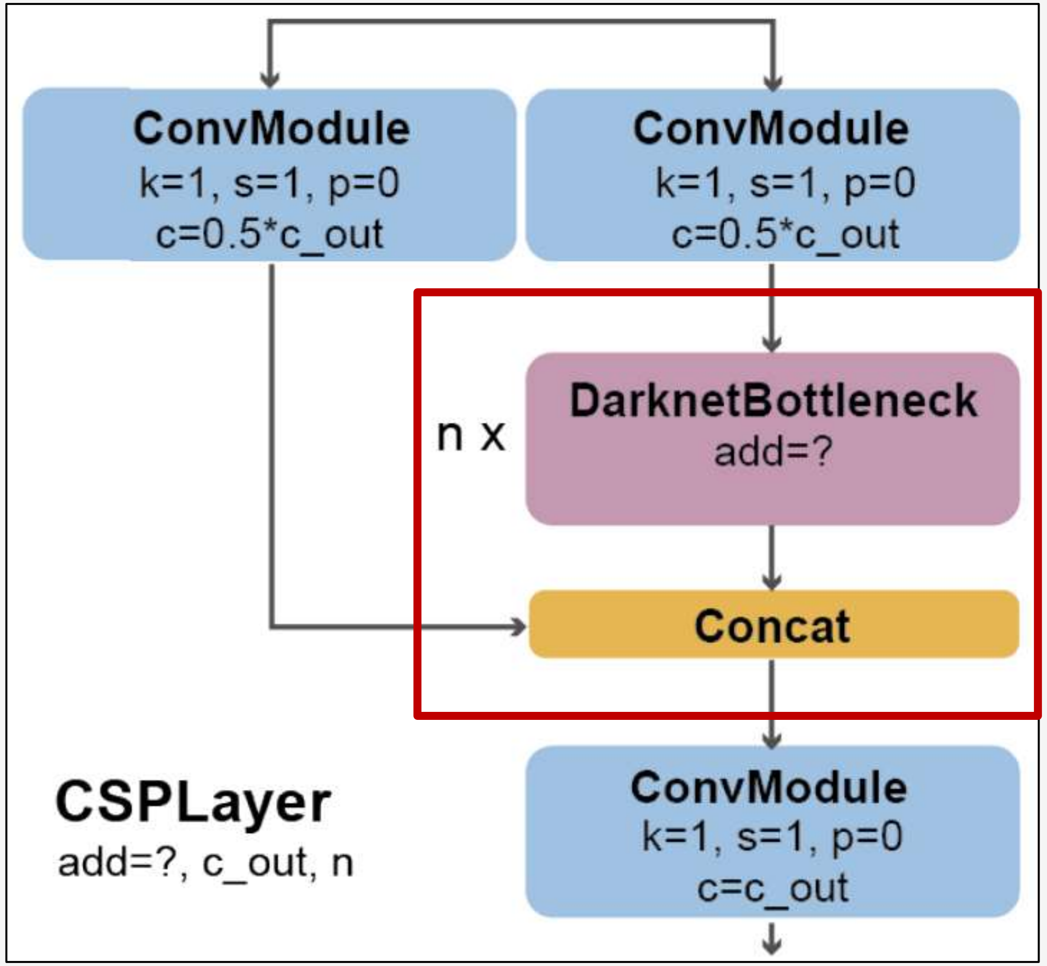
References
https://blog.roboflow.com/whats-new-in-yolov8/

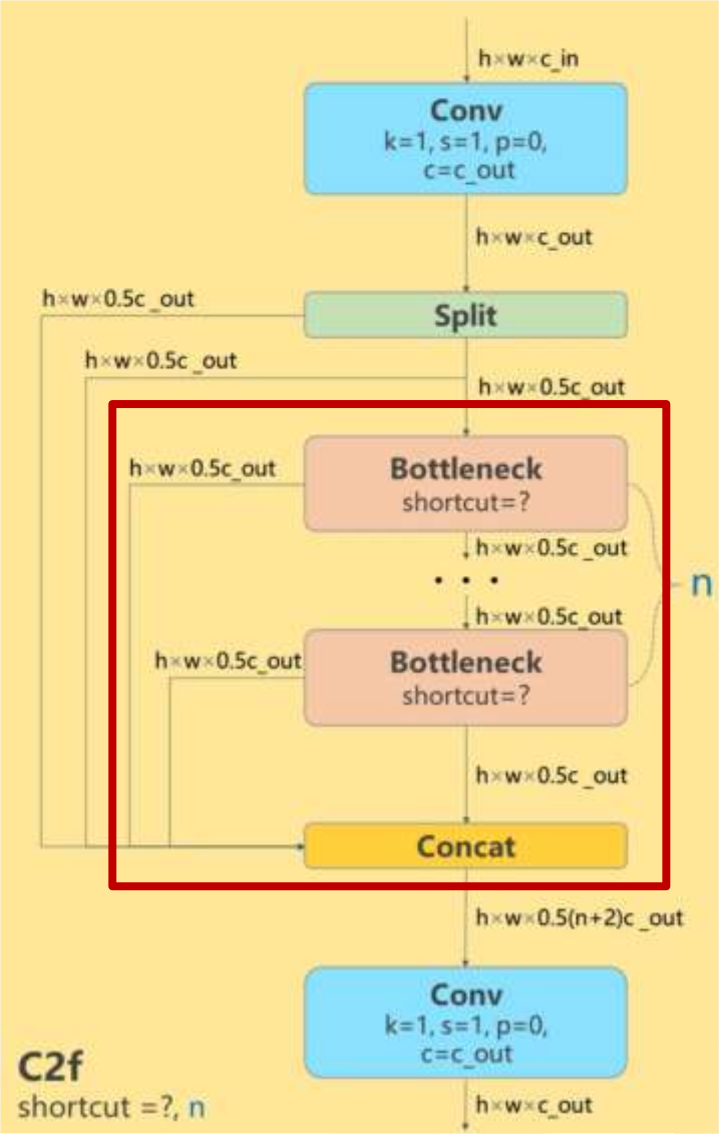
Architecture



References
https://epozen-dt.github.io/Yolov5/

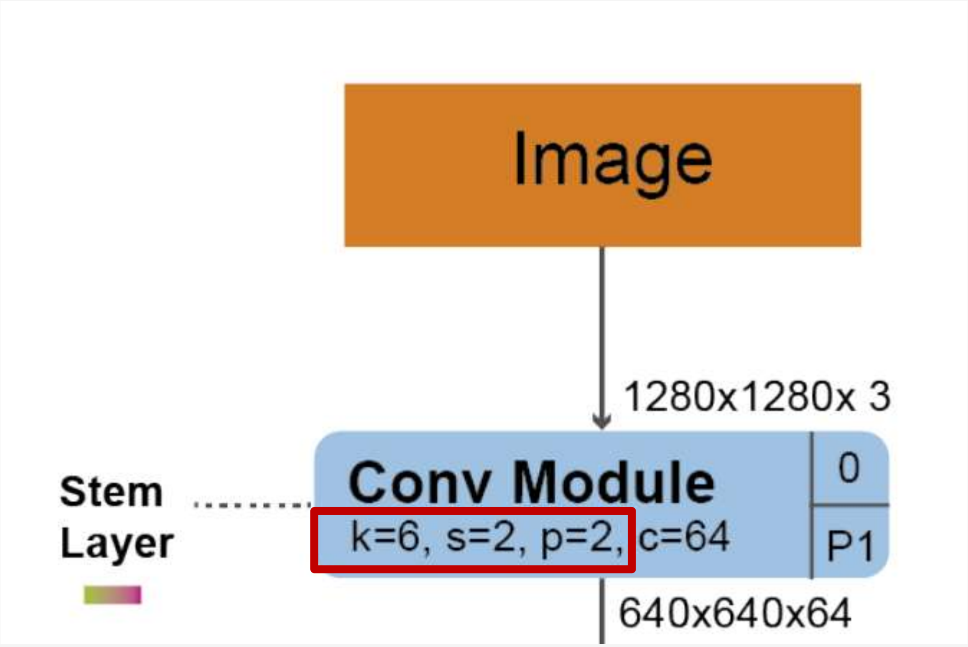
C2f Block

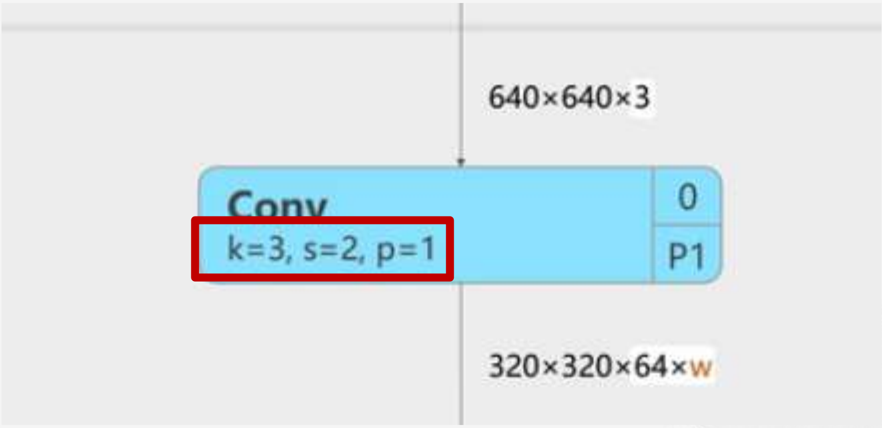




References
(Left) https://blog.roboflow.com/whats-new-in-yolov8/
(Right) https://arxiv.org/pdf/2304.00501.pdf

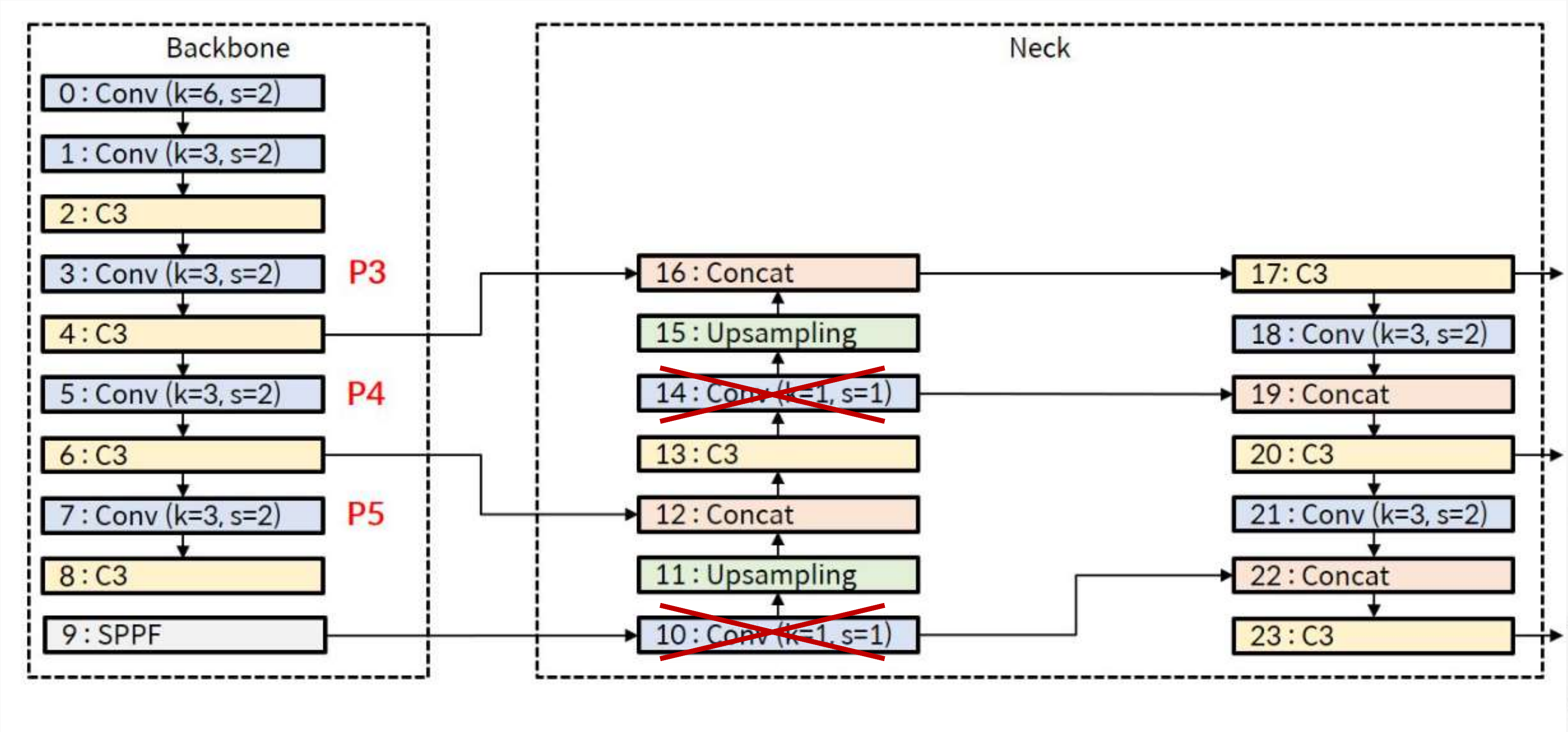
Replace the first 6x6 Conv with 3x3 Conv in the Backbone





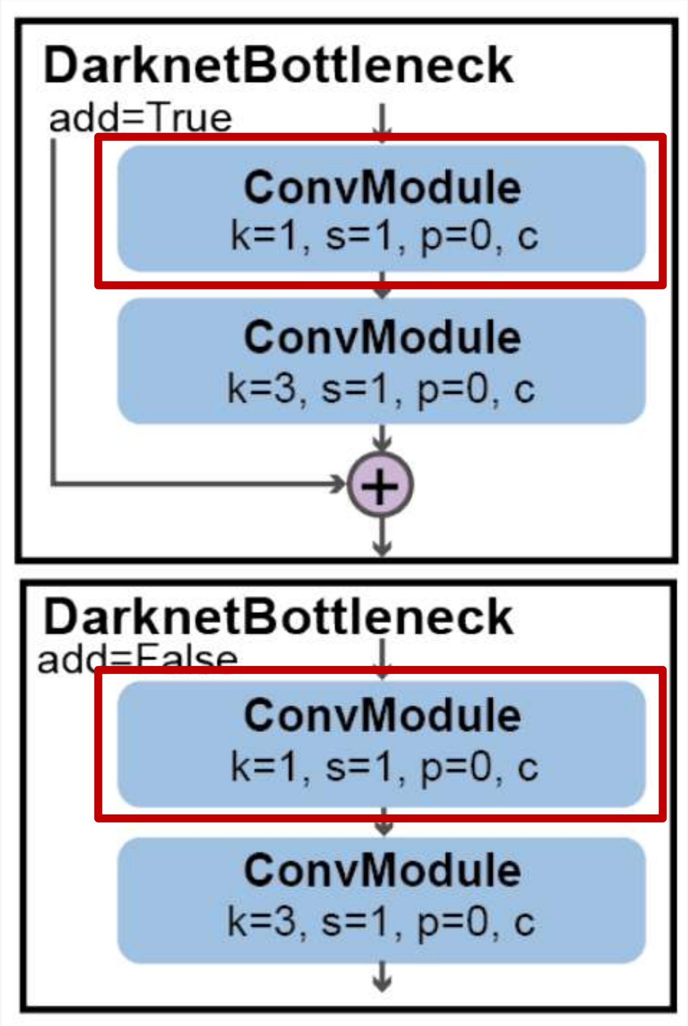
References
(Left) https://blog.roboflow.com/whats-new-in-yolov8/
(Right) https://arxiv.org/pdf/2304.00501.pdf

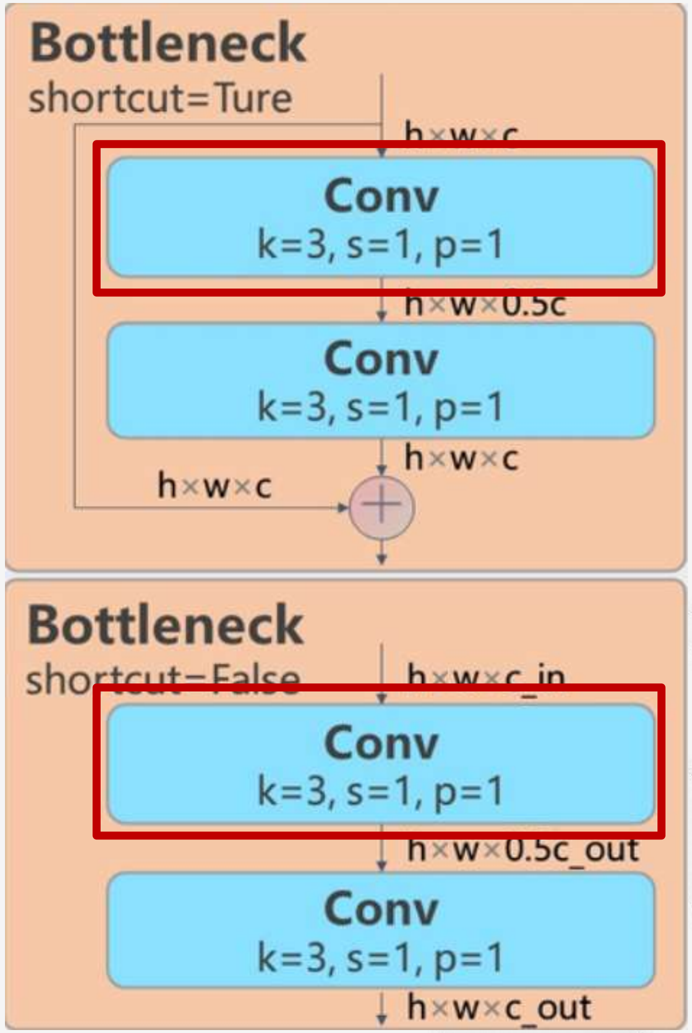
Delete Two Convs



References
https://epozen-dt.github.io/Yolov5/

Replace the First 1x1 Conv with 3x3 Conv in the Bottleneck



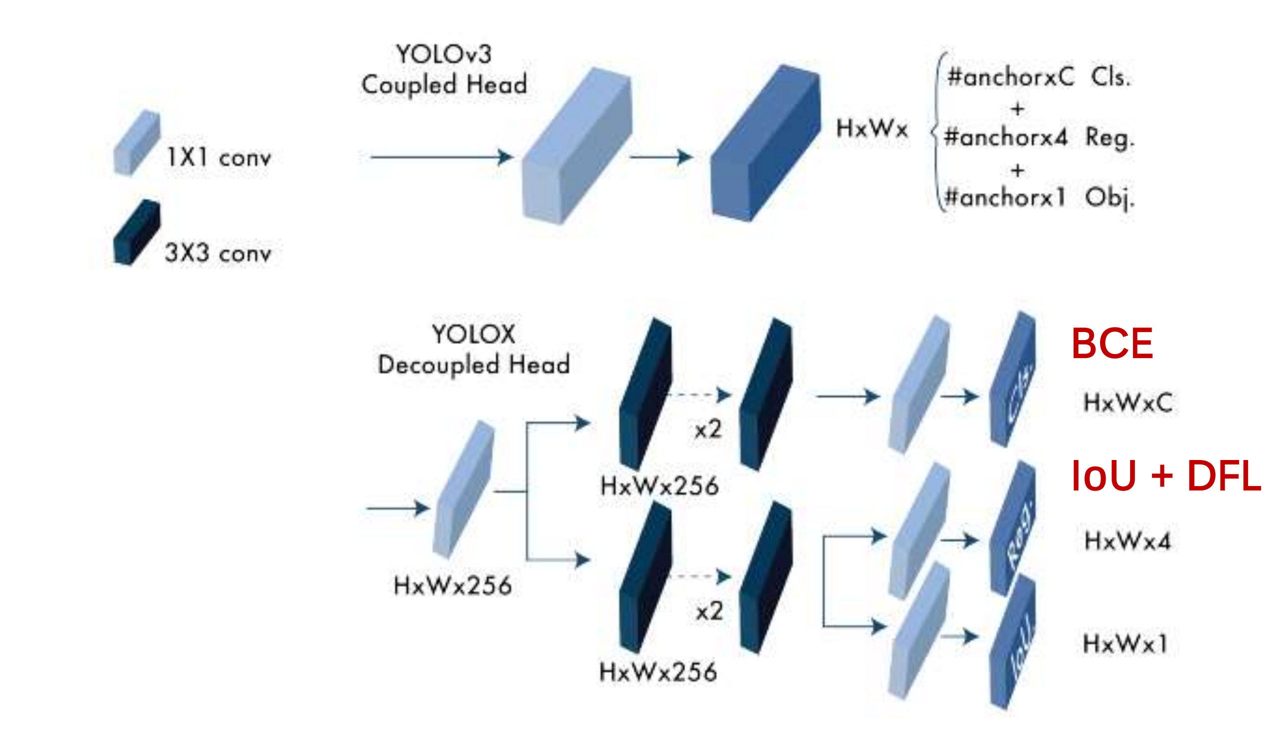


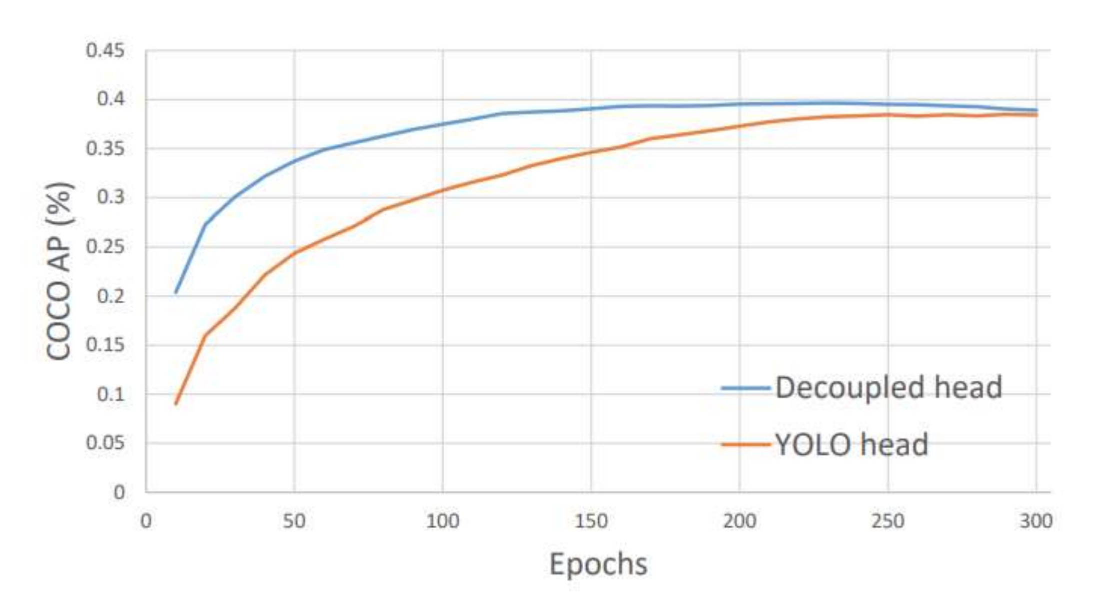
References
(Left) https://blog.roboflow.com/whats-new-in-yolov8/
(Right) https://arxiv.org/pdf/2304.00501.pdf

Decoupled Head

One-head에 비해 성능이 좋음 (속도, AP)

Anchor Free model





References
https://arxiv.org/pdf/2304.00501.pdf

Loss function

$$L_{Total} = \lambda_{bbox} \cdot L_{bbox} + \lambda_{cls} \cdot L_{cls} + \lambda_{dfl} \cdot L_{dfl}$$

L_{bbox} (Bounding box loss)

IoU 기반으로 측정

L_{cls} (Class loss)

Binary Cross Entropy

L_{dfl}(Bounding box loss)

옵션으로, 더 정확한 위치 측정을 위해 사용됨

Five Scaled Version

model	d (depth_multiple)	w (width_multiple)	r (ratio)
n	0.33	0.25	2.0
s	0.33	0.50	2.0
m	0.67	0.75	1.5
l	1.00	1.00	1.0
x	1.00	1.25	1.0

References
https://blog.roboflow.com/whats-new-in-yolov8/

Experimental Results

Performances

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

References
https://github.com/ultralytics/ultralytics

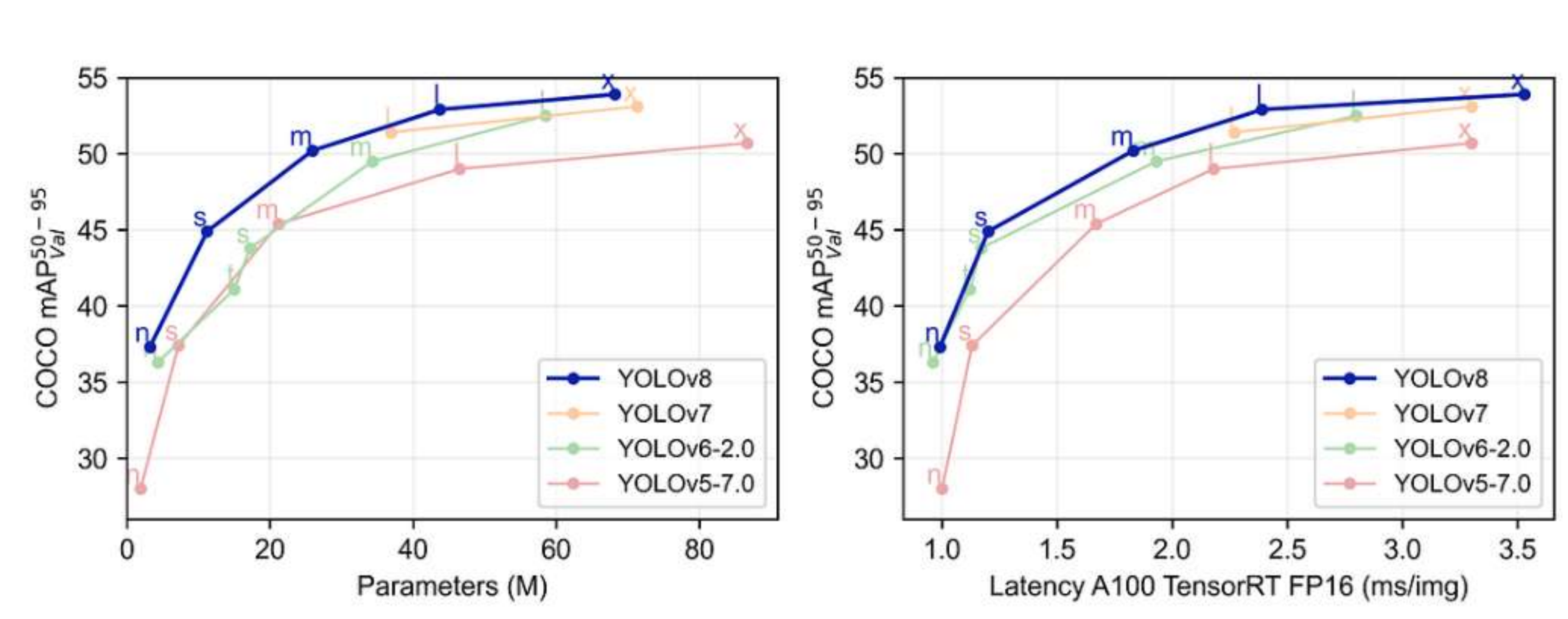
YOLOv8 vs YOLOv5

Model Size	Detection#	Segmentation#	Classification*
Nano	+33.21%	+32.97%	+3.10%
Small	+20.05%	+18.62%	+1.12%
Medium	+10.57%	+10.89%	+0.66%
Large	+7.96%	+6.73%	0.00%
Xtra Large	+6.31%	+5.33%	-0.76%

#Image Size = 640 *Image Size = 224

References
https://the-decoder.com/yolov8-shows-the-enormous-possibilities-of-computer-vision/

Performance of YOLO



References
https://docs.ultralytics.com/

Conclusion

Conclusion

- *A **new state-of-the-art (SOTA)** model is proposed, featuring an object detection model for P5 640 and P6 1280 resolutions, as well as a YOLACT-based instance segmentation model. The model also includes **different size options** with N/S/M/L/X scales, similar to YOLOv5, to cater to various scenarios.*

- *The backbone network and neck module are based on the **YOLOv7 ELAN design concept**, replacing the C3 module of YOLOv5 with the **C2f module**. However, there are a lot of operations such as Split and Concat in this C2f module that are not as deployment-friendly as before.*

- *The Head module has been updated to the current mainstream **decoupled structure**, separating the classification and detection heads, and switching from Anchor-Based to **Anchor-Free**.*

- *The loss calculation adopts the TaskAlignedAssigner in TOOD and introduces the **Distribution Focal Loss** to the regression loss.*

- *In the data augmentation part, **Mosaic is closed** in the last 10 training epoch, which is the same as YOLOX training part.*

References
https://openmmlab.medium.com/dive-into-yolov8-how-does-this-state-of-the-art-model-work-10f18f74bab1

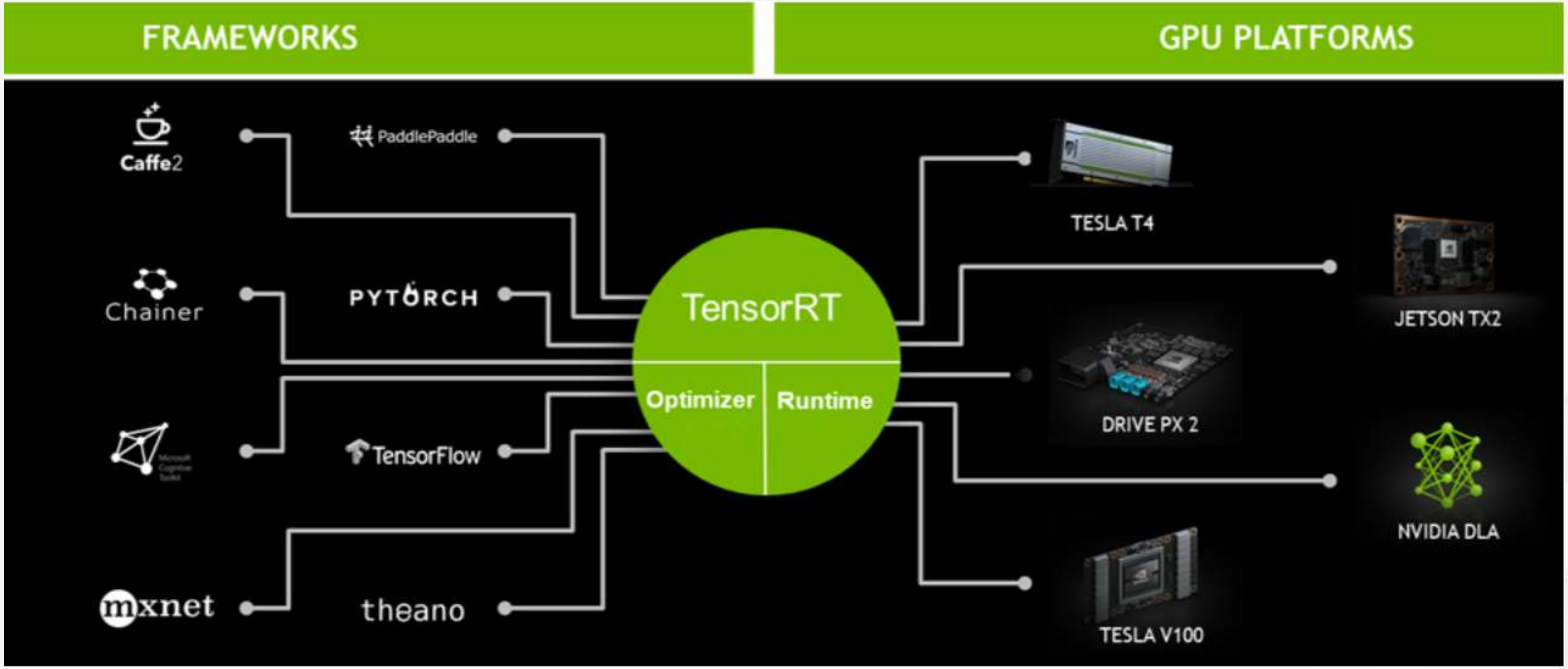
Tensor-RT

CONTENT

TensorRT

TensorRT

NVIDIA에서 만든 프레임워크로써, NVIDIA GPU에서 최적화 된 기술

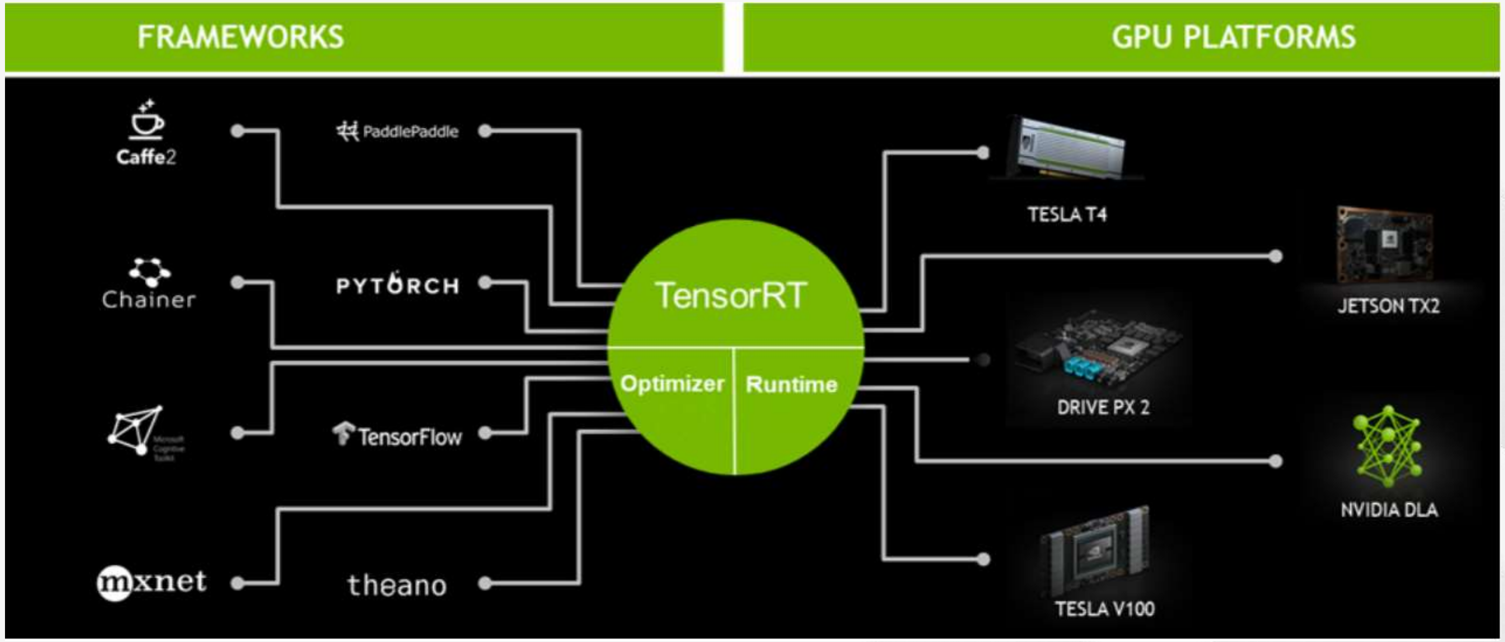


References
https://thecho7.tistory.com/entry/PyTorch-20-vs-ONNX-vs-TensorRT-%EB%B9%84%EA%B5%90
https://developer.nvidia.com/ko-kr/blog/nvidia-tensorrt-inference-%EC%B5%9C%EC%A0%81%ED%99%94-%EB%B0%8F-%EA%B0%80%EC%86%8D%ED%99%94%EB%A5%BC-%EC%9C%84%ED%95%9C-nvidia%EC%9D%98-toolkit/

TensorRT

- GPU가 지원하는 활용 가능한 최적의 연산 자원을 자동으로 사용할 수 있도록 Runtime binary 를 빌드함
- Latency와 Throughput을 향상시킴
- Deep Learning 응용 프로그램 및 서비스의 효율적인 실행이 가능

- Latency – 시간 단위 : 작업을 처리하는데 걸리는 시간
- Throughput – 일 단위 : 단위시간 (초)당 처리하는 작업의 수



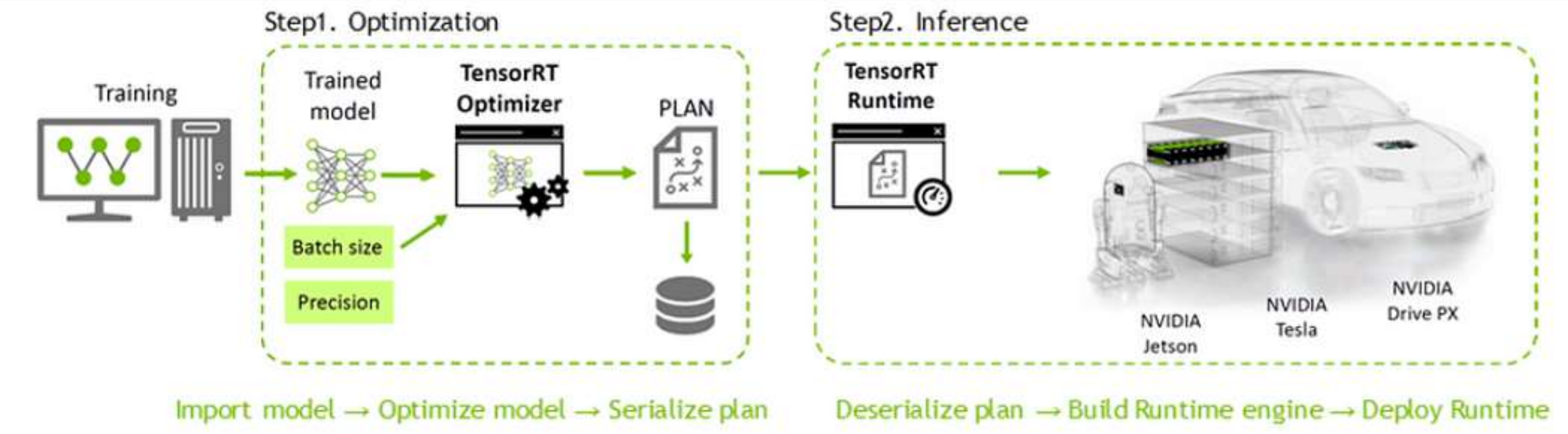
References
https://blog.kubwa.co.kr/inference-tensorrt-c1a97404eb0c

TensorRT의 구성

TensorRT Workflow

TensorRT는 C++ 및 Python 모두를 API 레벨에서 지원

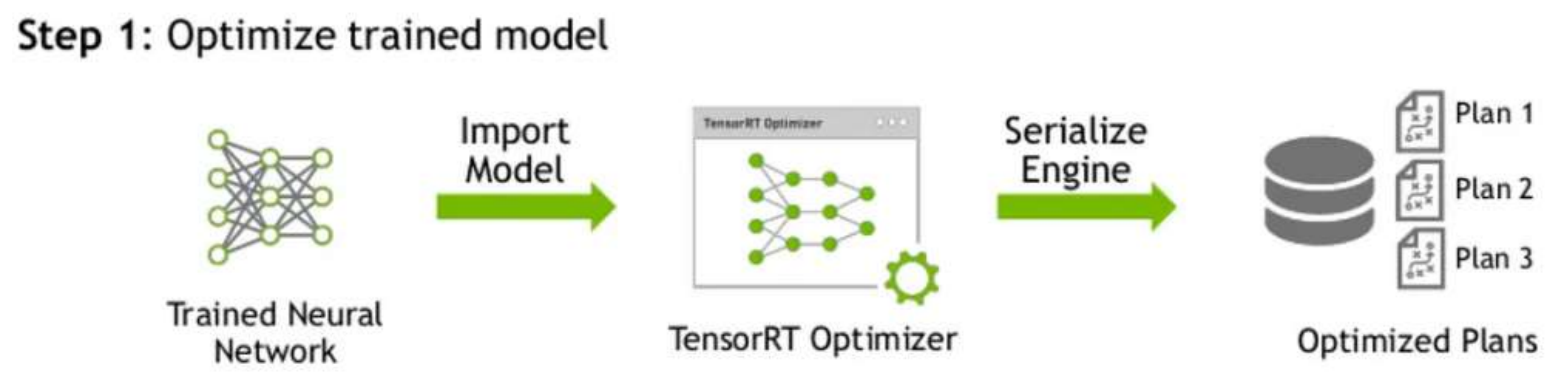
GPU programming language인 CUDA 지식이 별도로 없더라도 Deep Learning 분야의 **개발자들이 쉽게 사용**



References
https://blog.kubwa.co.kr/inference-tensorrt-c1a97404eb0c

Optimizer

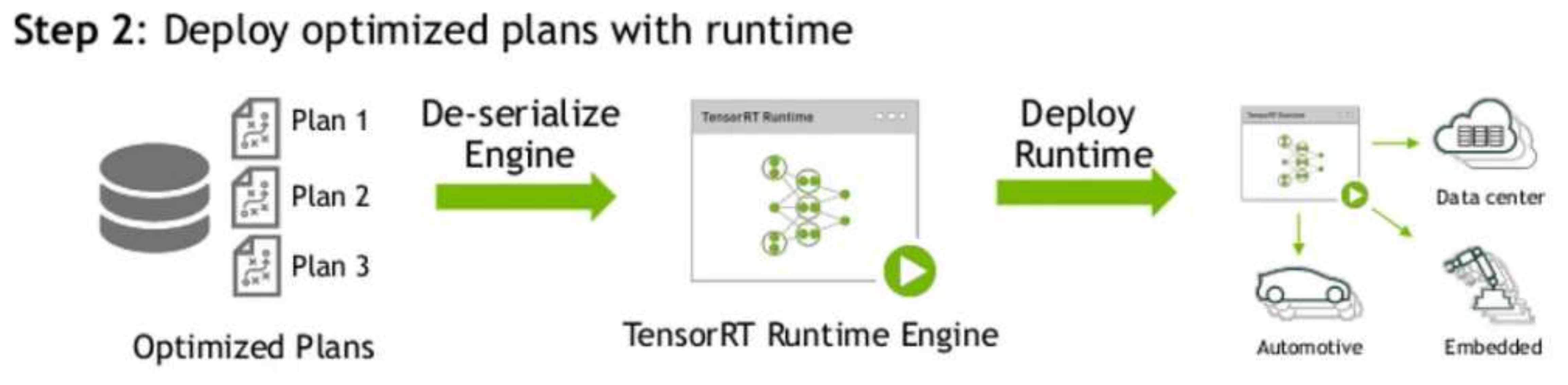
NVIDIA GPU 연산에 적합한 최적화 기법들을 사용해 훈련된 딥러닝 모델을 최적화하는 역할



References
https://blog.kubwa.co.kr/inference-tensorrt-c1a97404eb0c

Engine

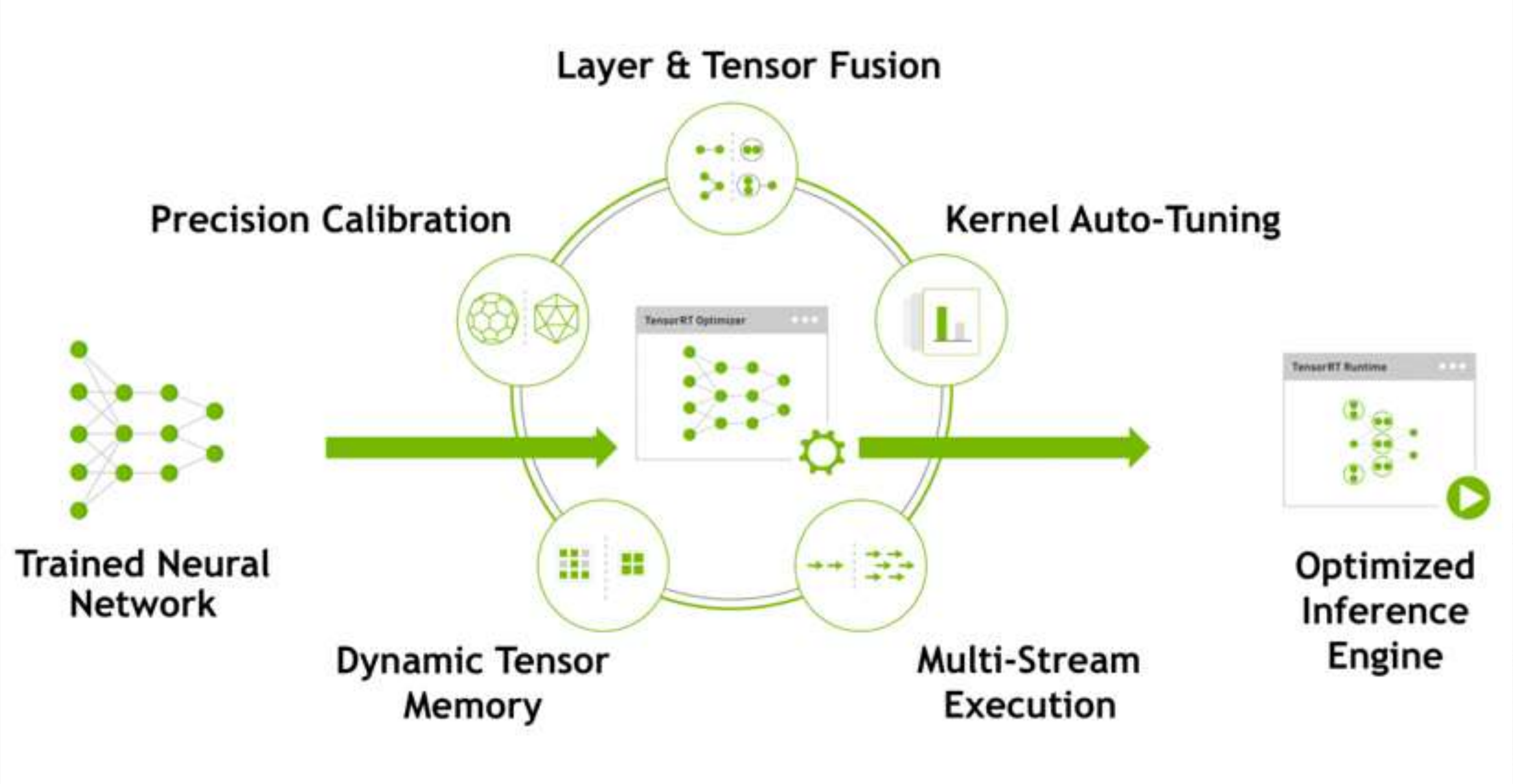
배포할 NVIDIA GPU 에 따라 최적의 연산을 수행할 수 있도록 도와주는 역할



References
https://blog.kubwa.co.kr/inference-tensorrt-c1a97404eb0c

딥러닝 가속화 방법

딥러닝 가속화 방법



References
https://developer.nvidia.com/ko-kr/blog/nvidia-tensorrt-inference-%EC%B5%9C%EC%A0%81%ED%99%94-%EB%B0%8F-%EA%B0%80%EC%86%8D%ED%99%94%EB%A5%BC-%EC%9C%84%ED%95%9C-nvidia%EC%9D%98-toolkit/

Quantization & Precision Calibration

양자화 및 정밀도 캘리브레이션

일상 생활에서 흔히 사용되는 소형 edge device 는 메모리, 성능, 저장공간 등 환경이 제한적이기 때문에 모델을 탑재하기에는 적합하지 않음

모델을 가볍게 만들어야 하는 경량화가 필요

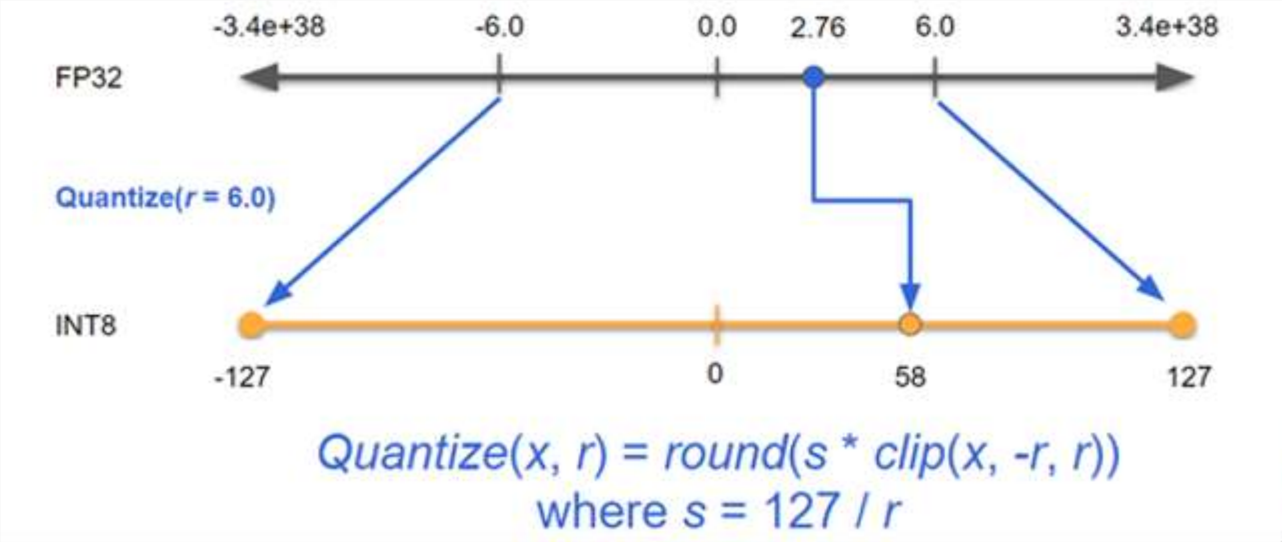
낮은 Precision의 Network일 수록 data의 크기 및 weight들의 bit수가 작기 때문에 더 빠르고 효율적인 연산이 가능

- Quantization
- Precision Calibration

References
https://developer.nvidia.com/ko-kr/blog/nvidia-tensorrt-inference-%EC%B5%9C%EC%A0%81%ED%99%94-%EB%B0%8F-%EA%B0%80%EC%86%8D%ED%99%94%EB%A5%BC-%EC%9C%84%ED%95%9C-nvidia%EC%9D%98-toolkit/

Quantization

- Neural Network 모델 내부의 대부분은 weight와 activation ouput으로 구성
- weight 와 activation output 은 모델의 정확도를 높이기 위해, 32bit floating point (FP32) 로 표현
- 리소스가 제한된 환경에서 모든 weight와 activation output을 32 bit floating point로 표현한 모델은 추론에 사용하기 어려움
- Symmetric Linear Quantization을 사용하여 양자화 진행



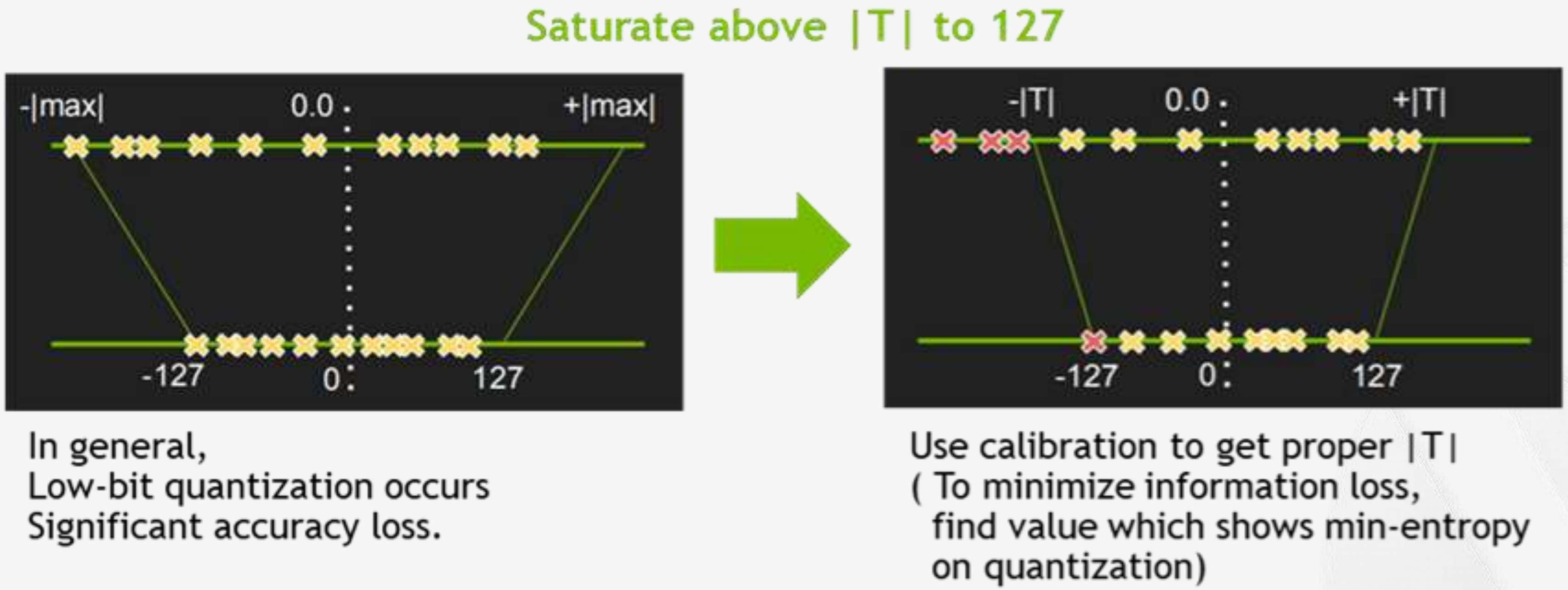
Symmetric linear quantization

x : Input
r : Floating point range
s : Scaling factor

References
https://developer.nvidia.com/ko-kr/blog/nvidia-tensorrt-inference-%EC%B5%9C%EC%A0%81%ED%99%94-%EB%B0%8F-%EA%B0%80%EC%86%8D%ED%99%94%EB%A5%BC-%EC%9C%84%ED%95%9C-nvidia%EC%9D%98-toolkit/

Precision Calibration

- FP16으로의 precision down-scale은 Network의 accuracy drop에 큰 영향을 주지는 않지만, INT8로의 down-scale은 accuracy drop을 보이는 몇 부류의 Network이 존재
- Calibration 작업을 활용하여 Quantization시 가중치 및 intermediate tensor 들의 정보 손실을 최소화
- EntronpyCalibrator, EntropyCalibrator2 그리고 MinMaxCalibrator를 지원

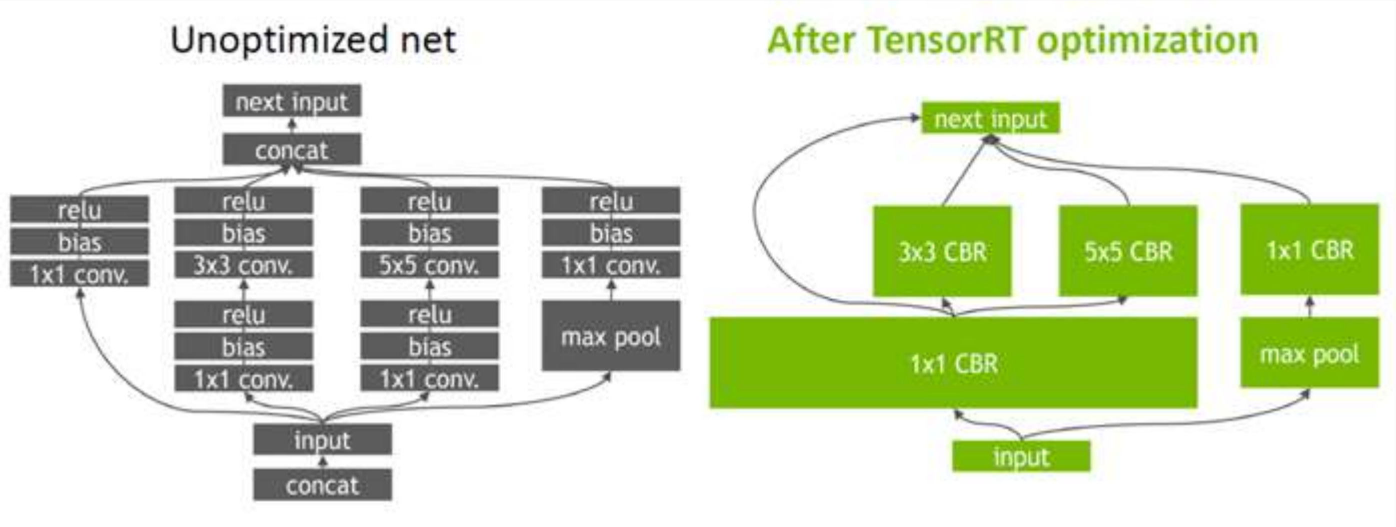


References
https://developer.nvidia.com/ko-kr/blog/nvidia-tensorrt-inference-%EC%B5%9C%EC%A0%81%ED%99%94-%EB%B0%8F-%EA%B0%80%EC%86%8D%ED%99%94%EB%A5%BC-%EC%9C%84%ED%95%9C-nvidia%EC%9D%98-toolkit/

Graph Optimization

- 일반적으로 Graph Optimization은 Deep Learning Network에서 사용되는 primitive 연산 형태, compound 연산 형태의 graph node들을 각 platform에 최적화된 code를 구성하기 위하여 사용
- TensorRT에서는 이를 기반으로 Layer Fusion 방식과 Tensor Fusion 방식을 동시에 적용하여 그래프를 단순화 시켜 모델의 Layer 수가 크게 감소

- Layer Fusion : 딥러닝 네트워크에서 이루어진 여러 Layer들을 하나의 Layer로 합치는 작업
- Tensor Fusion : 감소될 준비가 된 모든 텐서를 하나의 감소 연산으로 결합하려고 시도하는 작업



Networks	Number of layers (Before)	Number of layers (After)
VGG19	43	27
Inception v3	309	113
ResNet-152	670	159

References
https://blog.kubwa.co.kr/inference-tensorrt-c1a97404eb0c

Kernel Auto-tuning

- TensorRT는 NVIDIA의 다양한 platform 및 architecture에 맞는 Runtime 생성을 도움
- CUDA engine 갯수, architecture, memory 그리고 serialized engine 포함 여부에 따라 최적화된 kernel(커널)을 찾아 선택적으로 engine을 생성
- TensorRT Runtime engine build 시에 시행하여 최종적으로 최적의 engine binary 생성을 도움

References
https://blog.kubwa.co.kr/inference-tensorrt-c1a97404eb0c

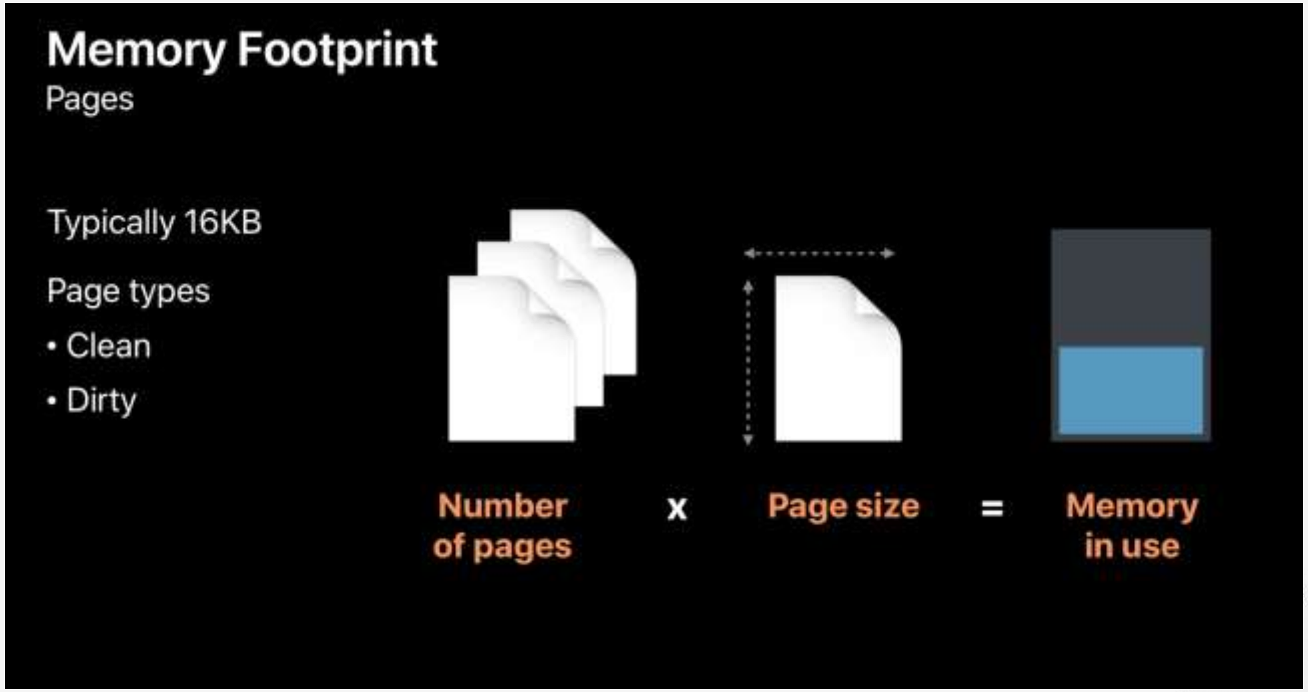
Dynamic Tensor Memory & Multi-Stream Execution

- Dynamic tensor memory

  Memory management를 통하여 footprint를 줄여 재사용을 할 수 있도록 도움

- Multi-stream execution

  CUDA stream 기술을 이용하여 multiple input stream의 scheduling을 통해 병렬 효율을 극대화



References
https://developer.nvidia.com/ko-kr/blog/nvidia-tensorrt-inference-%EC%B5%9C%EC%A0%81%ED%99%94-%EB%B0%8F-%EA%B0%80%EC%86%8D%ED%99%94%EB%A5%BC-%EC%9C%84%ED%95%9C-nvidia%EC%9D%98-toolkit/

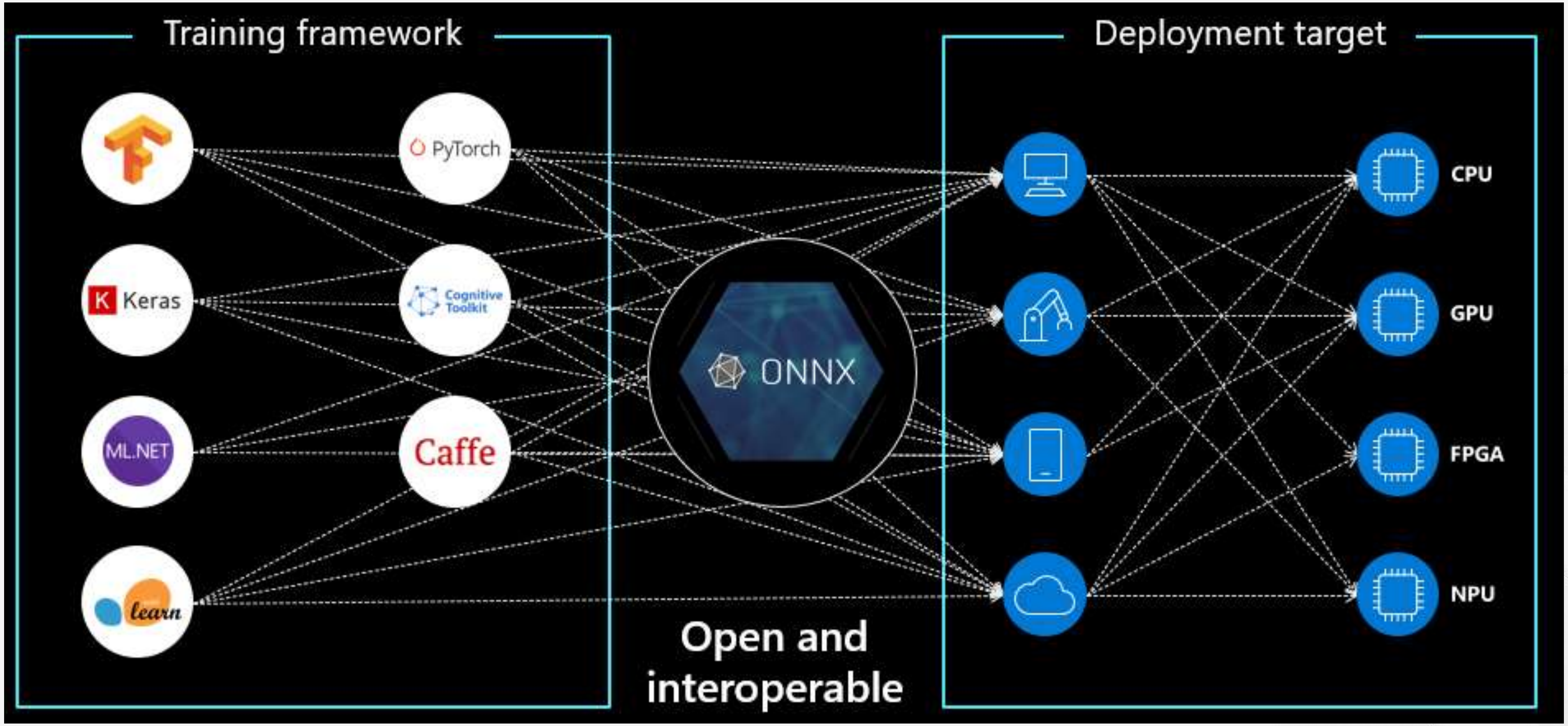
Advantages of TensorRT

Advantages of TensorRT

- C++과 python을 API 레벨에서 지원하므로 CUDA를 잘 모르는 Deep Learning 개발자들도 쉽게 사용할 수 있음
- latency 및 throuput을 쉽게 향상
- 다양한 layer 및 연산에 대해 customization할 수 있는 방법론을 제공

ONNX

- Open Neural Network Exchange 오픈 소스 프로젝트
- ONNX는 인공지능(AI) 모델을 표준 형식으로 표현하고 **서로 다른 딥러닝 프레임워크** 간에 모델의 변환 및 공유를 지원



References
https://thecho7.tistory.com/entry/PyTorch-20-vs-ONNX-vs-TensorRT-%EB%B9%84%EA%B5%90

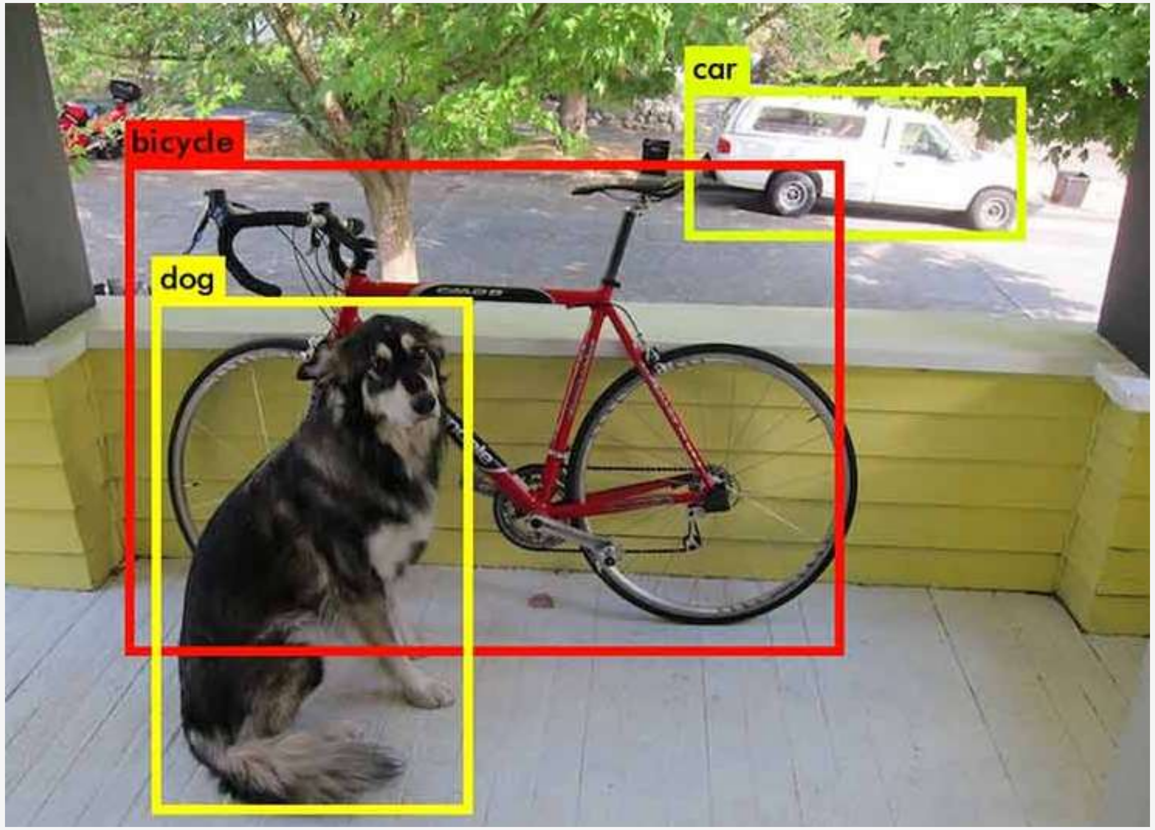
[Practice 1] 마스크 착용 유무 프로젝트

CONTENT

실습 소개

Object Detection이란?

이미지 내의 모든 Object에 대하여 Classification와 Localization을 수행



References
https://machinethink.net/blog/object-detection-with-yolo/

[실습1] 마스크 착용 유무 프로젝트

References
https://www.lgcns.com/pr/news/12676/

데이터셋

데이터셋

References
https://github.com/VictorLin000/YOLOv3_mask_detect

데이터셋 소개

- 데이터셋 다운로드 링크 : https://drive.google.com/drive/folders/1aAXDTl5kMPKAHE08WKGP2PifIdc21-ZG
- 678 Images
- 3 Classes (착용, 미착용, 잘못된 착용)
- Bounding box annotations are provided in the PASCAL YOLO format

*Mask_180.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
2 0.11375 0.379375 0.0375 0.04625
2 0.213125 0.386875 0.04625 0.04625
0 0.305625 0.38125 0.04375 0.0525
0 0.415 0.4375 0.0425 0.05
2 0.57125 0.408125 0.06 0.05375
2 0.740625 0.42375 0.06875 0.0575
```

References
https://github.com/VictorLin000/YOLOv3_mask_detect

데이터셋 구조

- mask_yolo
  - Mask_1.jpg
  - Mask_1.txt
  - Mask_10.jpg
  - Mask_10.txt
  - Mask_100.jpg
  - Mask_100.txt
  - Mask_101.jpg
  - Mask_101.txt
  - Mask_102.jpg
  - Mask_102.txt
  - Mask_103.jpg
  - Mask_103.txt
  - Mask_104.jpg
  - Mask_104.txt

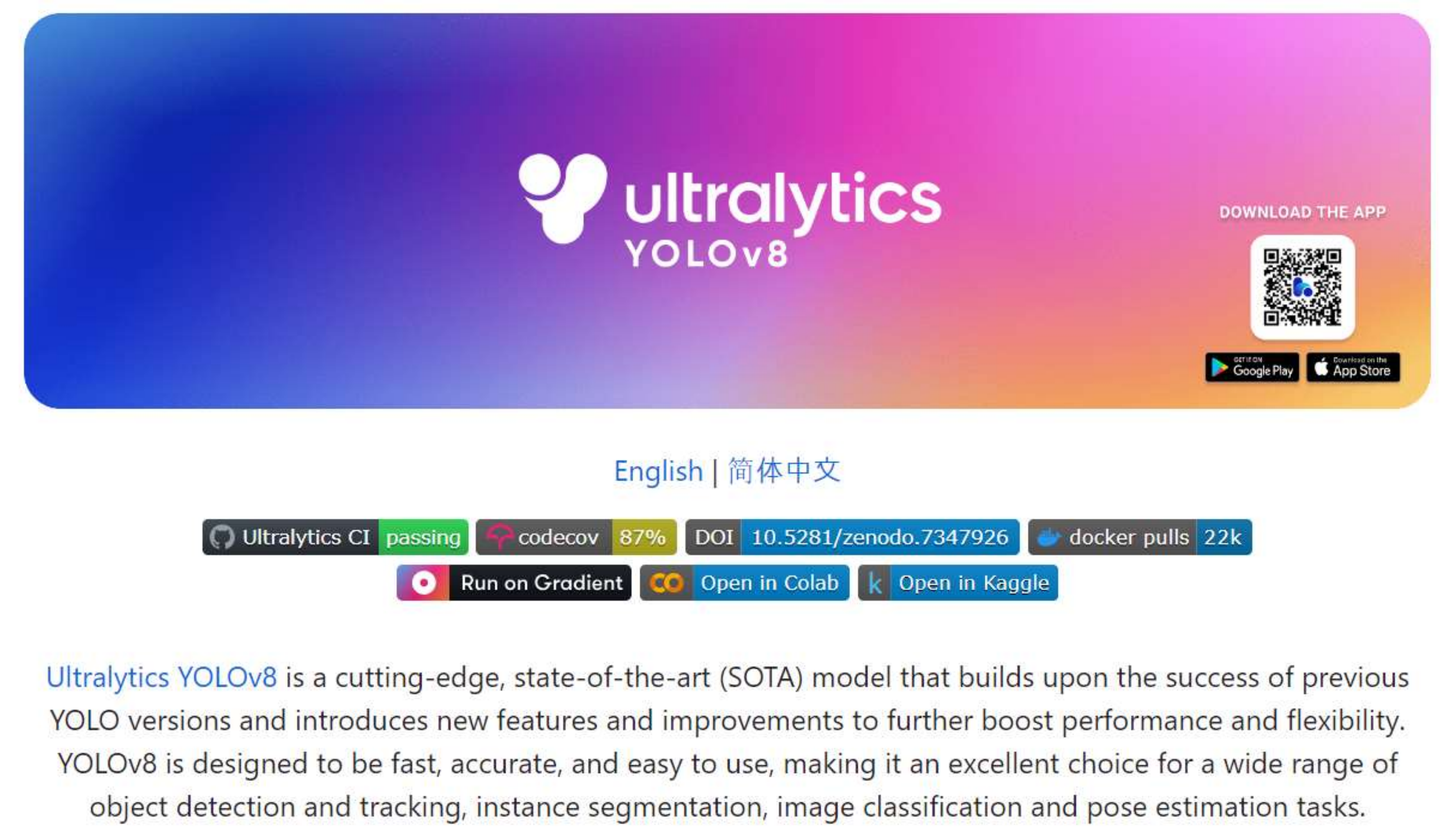
```
2 0.11375 0.379375 0.0375 0.04625
2 0.213125 0.386875 0.04625 0.04625
0 0.305625 0.38125 0.04375 0.0525
0 0.415 0.4375 0.0425 0.05
2 0.57125 0.408125 0.06 0.05375
2 0.740625 0.42375 0.06875 0.0575
```

실습 튜토리얼

YOLOv8 - ultralytics

공식 Github : https://github.com/ultralytics/ultralytics

공식 Documents : https://docs.ultralytics.com/



실습 환경 구축

- 실습 환경 구축

  pip install ultralytics

  or

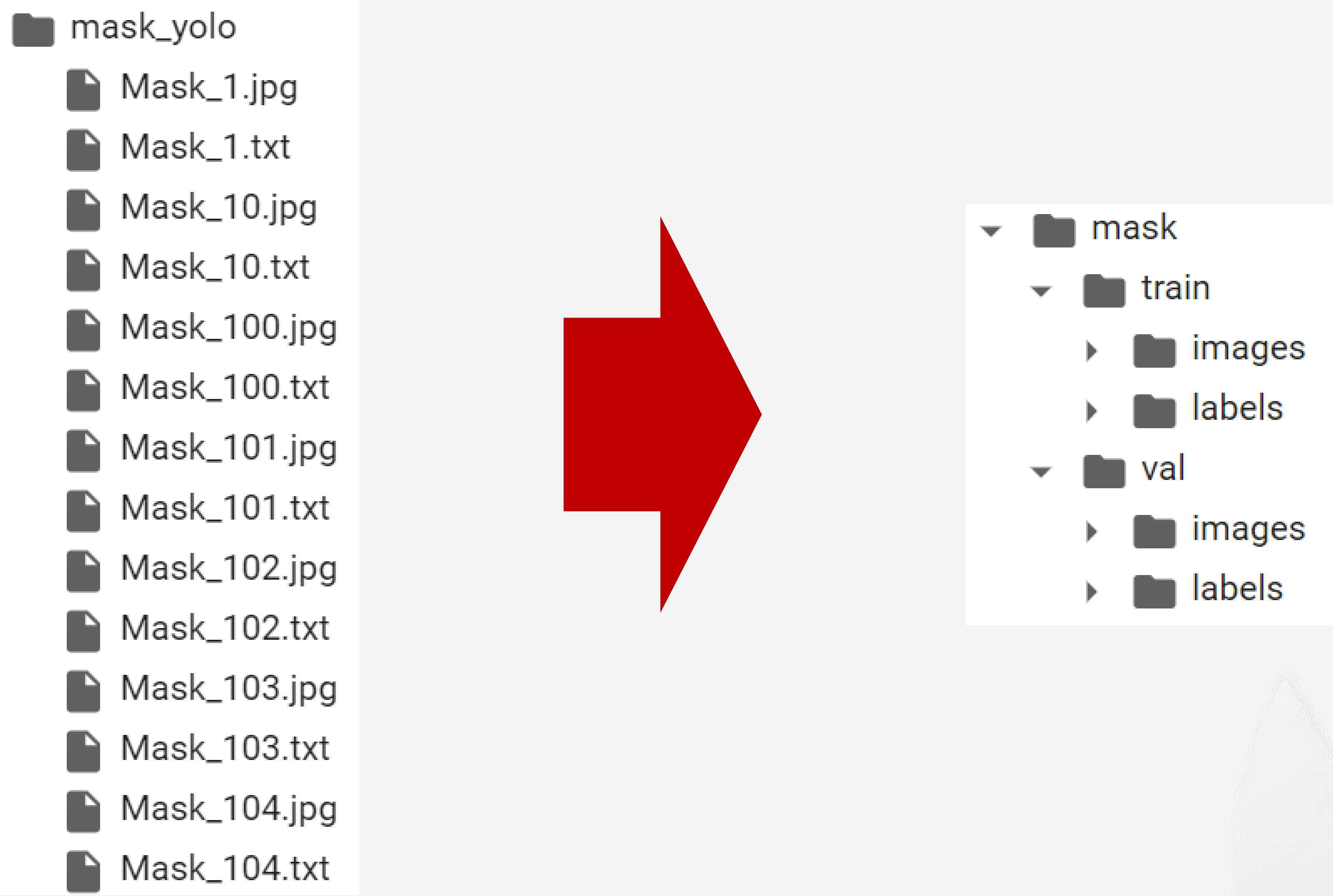
  pip install git+https://github.com/ultralytics/ultralytics.git@main

- 정상 설치 확인

```
import ultralytics
ultralytics.checks()
```

```
Ultralytics YOLOv8.0.157 🚀 Python-3.10.12 torch-2.0.1+cu118 CUDA:0 (Tesla T4, 15102MiB)
Setup complete ✅ (2 CPUs, 12.7 GB RAM, 26.3/166.8 GB disk)
```

데이터셋 전처리


mask_yolo
Mask_1.jpg
Mask_1.txt
Mask_10.jpg
Mask_10.txt
Mask_100.jpg
Mask_100.txt
Mask_101.jpg
Mask_101.txt
Mask_102.jpg
Mask_102.txt
Mask_103.jpg
Mask_103.txt
Mask_104.jpg
Mask_104.txt
mask
train
images
labels
val
images
labels

데이터셋 전처리

- Train/Test Split

```
random.shuffle(file_list)

test_ratio = 0.1
test_list = file_list[:int(len(file_list)*test_ratio)]
train_list = file_list[int(len(file_list)*test_ratio):]

for i in test_list:
  f_name = os.path.splitext(i)[0]
  copyfile(os.path.join(road_sign_path, 'images', (f_name+img_)), os.path.join(mask_path,
'val/images', (f_name+img_)))
  copyfile(os.path.join(road_sign_path, 'labels', (f_name+label_)), os.path.join(mask_path,
'val/labels', (f_name+label_)))
for i in train_list:
  f_name = os.path.splitext(i)[0]
  copyfile(os.path.join(road_sign_path, 'images', (f_name+img_)), os.path.join(mask_path,
'train/images', (f_name+img_)))
  copyfile(os.path.join(road_sign_path, 'labels', (f_name+label_)), os.path.join(mask_path,
'train/labels', (f_name+label_)))
```

Config 파일 생성

```
import yaml
data =dict()

data['train'] = '/content/drive/MyDrive/dataset/mask/train'
data['val'] = '/content/drive/MyDrive/dataset/mask/val'
data['test'] = '/content/drive/MyDrive/dataset/mask/val'

data['nc'] = 3
data['names'] =['OK','improperly', 'NO']

with open('mask_detection.yaml', 'w') as f:
    yaml.dump(data, f)
```

Train

튜토리얼 링크 : https://docs.ultralytics.com/modes/train/

```
from ultralytics import YOLO

# Load a model
model = YOLO('yolov8n.yaml')  # build a new model from YAML
model = YOLO('yolov8n.pt')  # load a pretrained model (recommended for training)
model = YOLO('yolov8n.yaml').load('yolov8n.pt')  # build from YAML and transfer weights

# Train the model
results = model.train(data='coco128.yaml', epochs=100, imgsz=640)
```

```
yolo train data=coco.yaml
```

Train

Train Arguments

Key	Value	Description
model	None	path to model file, i.e. yolov8n.pt, yolov8n.yaml
data	None	path to data file, i.e. coco128.yaml
epochs	100	number of epochs to train for
patience	50	epochs to wait for no observable improvement for early stopping of training
batch	16	number of images per batch (-1 for AutoBatch)
imgsz	640	size of input images as integer
save	True	save train checkpoints and predict results
save_period	-1	Save checkpoint every x epochs (disabled if < 1)
cache	False	True/ram, disk or False. Use cache for data loading
device	None	device to run on, i.e. cuda device=0 or device=0,1,2,3 or device=cpu
workers	8	number of worker threads for data loading (per RANK if DDP)
project	None	project name
name	None	experiment name
exist_ok	False	whether to overwrite existing experiment
pretrained	False	whether to use a pretrained model
optimizer	'auto'	optimizer to use, choices=[SGD, Adam, Adamax, AdamW, NAdam, RAdam, RMSProp, auto]
verbose	False	whether to print verbose output
seed	0	random seed for reproducibility
deterministic	True	whether to enable deterministic mode
single_cls	False	train multi-class data as single-class
rect	False	rectangular training with each batch collated for minimum padding
cos_lr	False	use cosine learning rate scheduler

Train

Train Arguments

Key	Value	Description
cos_lr	False	use cosine learning rate scheduler
close_mosaic	10	(int) disable mosaic augmentation for final epochs (0 to disable)
resume	False	resume training from last checkpoint
amp	True	Automatic Mixed Precision (AMP) training, choices=[True, False]
fraction	1.0	dataset fraction to train on (default is 1.0, all images in train set)
profile	False	profile ONNX and TensorRT speeds during training for loggers
freeze	None	(int or list, optional) freeze first n layers, or freeze list of layer indices during training
lr0	0.01	initial learning rate (i.e. SGD=1E-2, Adam=1E-3)
lrf	0.01	final learning rate (lr0 * lrf)
momentum	0.937	SGD momentum/Adam beta1
weight_decay	0.0005	optimizer weight decay 5e-4
warmup_epochs	3.0	warmup epochs (fractions ok)
warmup_momentum	0.8	warmup initial momentum
warmup_bias_lr	0.1	warmup initial bias lr
box	7.5	box loss gain
cls	0.5	cls loss gain (scale with pixels)
dfl	1.5	dfl loss gain
pose	12.0	pose loss gain (pose-only)
kobj	2.0	keypoint obj loss gain (pose-only)
label_smoothing	0.0	label smoothing (fraction)
nbs	64	nominal batch size
overlap_mask	True	masks should overlap during training (segment train only)
mask_ratio	4	mask downsample ratio (segment train only)
dropout	0.0	use dropout regularization (classify train only)
val	True	validate/test during training

Validation

튜토리얼 링크 : https://docs.ultralytics.com/modes/val/

```
from ultralytics import YOLO

# Load a model
model = YOLO('yolov8n.pt')  # load an official model
model = YOLO('path/to/best.pt')  # load a custom model

# Validate the model
metrics = model.val()  # no arguments needed, dataset and settings remembered
metrics.box.map    # map50-95
metrics.box.map50  # map50
metrics.box.map75  # map75
metrics.box.maps   # a list contains map50-95 of each category
```

```
yolo val model=yolov8n.pt
or
model('yolov8n.pt').val()
```

Validation

Validation Arguments

Key	Value	Description
data	None	path to data file, i.e. coco128.yaml
imgsz	640	size of input images as integer
batch	16	number of images per batch (-1 for AutoBatch)
save_json	False	save results to JSON file
save_hybrid	False	save hybrid version of labels (labels + additional predictions)
conf	0.001	object confidence threshold for detection
iou	0.6	intersection over union (IoU) threshold for NMS
max_det	300	maximum number of detections per image
half	True	use half precision (FP16)
device	None	device to run on, i.e. cuda device=0/1/2/3 or device=cpu
dnn	False	use OpenCV DNN for ONNX inference
plots	False	show plots during training
rect	False	rectangular val with each batch collated for minimum padding
split	val	dataset split to use for validation, i.e. 'val', 'test' or 'train'

Inference

튜토리얼 링크 : https://docs.ultralytics.com/modes/predict/#inference-sources

```python
from ultralytics import YOLO

# Load a pretrained YOLOv8n model
model = YOLO('yolov8n.pt')

# Define path to the image file
source = 'path/to/image.jpg'

# Run inference on the source
results = model(source)  # list of Results objects
```

```python
from ultralytics import YOLO

# Load a pretrained YOLOv8n model
model = YOLO('yolov8n.pt')

# Run inference on 'bus.jpg' with arguments
model.predict('bus.jpg', save=True, imgsz=320, conf=0.5)
```

Inference

Attributes of Results

Attribute	Type	Description
orig_img	numpy.ndarray	The original image as a numpy array.
orig_shape	tuple	The original image shape in (height, width) format.
boxes	**Boxes, optional**	**A Boxes object containing the detection bounding boxes.**
masks	Masks, optional	A Masks object containing the detection masks.
probs	Probs, optional	A Probs object containing probabilities of each class for classification task.
keypoints	Keypoints, optional	A Keypoints object containing detected keypoints for each object.
speed	dict	A dictionary of preprocess, inference, and postprocess speeds in milliseconds per image.
names	dict	A dictionary of class names.
path	str	The path to the image file.

Inference

튜토리얼 링크 : https://docs.ultralytics.com/modes/predict/#inference-sources

```
from ultralytics import YOLO

# Load a pretrained YOLOv8n model
model = YOLO('yolov8n.pt')

# Run inference on an image
results = model('bus.jpg')  # results list

# View results
for r in results:
    print(r.boxes)  # print the Boxes object containing the detection bounding boxes
```

Inference

```
from ultralytics import YOLO
import cv2
import os
from ultralytics.yolo.utils.plotting import Annotator
import matplotlib.pyplot as plt
import numpy as np

model = YOLO('./best_mask.pt')

root_folder = '../dataset/mask/val/images'
result_folder = '../dataset/mask/result'
test_img_list = os.listdir(root_folder)
device = 'cpu'
color_dict = [(0, 255, 0), (255, 0, 0), (0, 0, 255)]
```

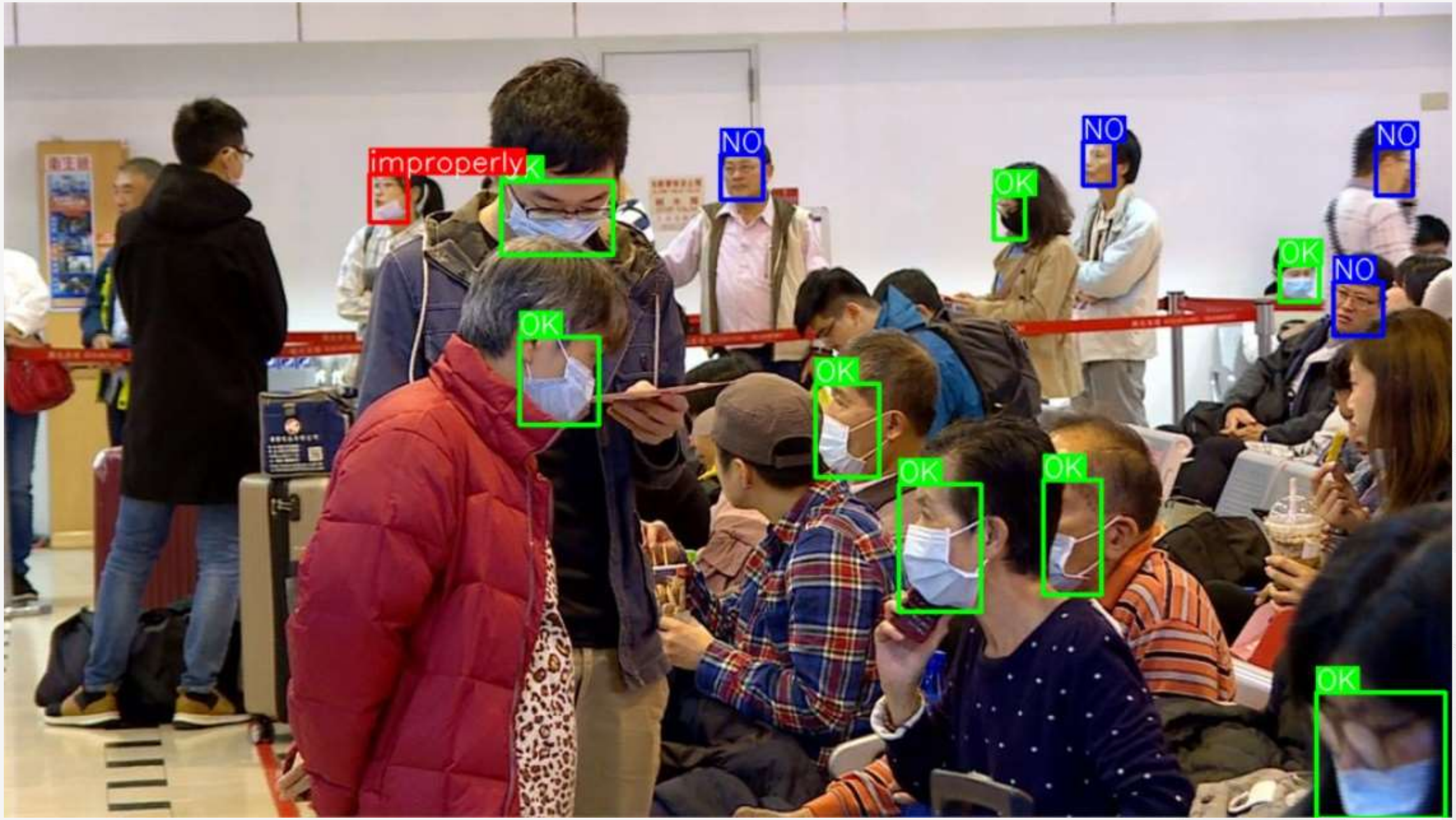
Inference

```
for idx , file in enumerate(test_img_list):
    black = np.zeros(shape = (640, 1280,3), dtype = np.uint8)
    test_img = cv2.imread(os.path.join(root_folder, file))
    img_src = cv2.cvtColor(test_img, cv2.COLOR_BGR2RGB)
    results = model(test_img)

    for result in results:
        annotator = Annotator(img_src)
        boxes = result.boxes
        for box in boxes:
            b = box.xyxy[0]  # get box coordinates in (top, left, bottom, right) format
            cls = box.cls
            annotator.box_label(b, model.names[int(cls)], color_dict[int(cls)])
    test_img = annotator.result()
    h,w,_ = test_img.shape
    if h <w:
        r = min(640/h, 1280/w)
        test_img = cv2.resize(test_img, (0,0), fx=r, fy=r)
        black[:test_img.shape[0],:test_img.shape[1],:] = test_img[:,:,:]
        cv2.imwrite(os.path.join(result_folder, file), cv2.cvtColor(black, cv2.COLOR_RGB2BGR))
```

실습 결과

실습 결과



[Practice 2] Tensor-RT 기반의 Yolov8, 표지판 신호등 검출 프로젝트

CONTENT

01

실습 소개

02

데이터셋

03

실습 튜토리얼

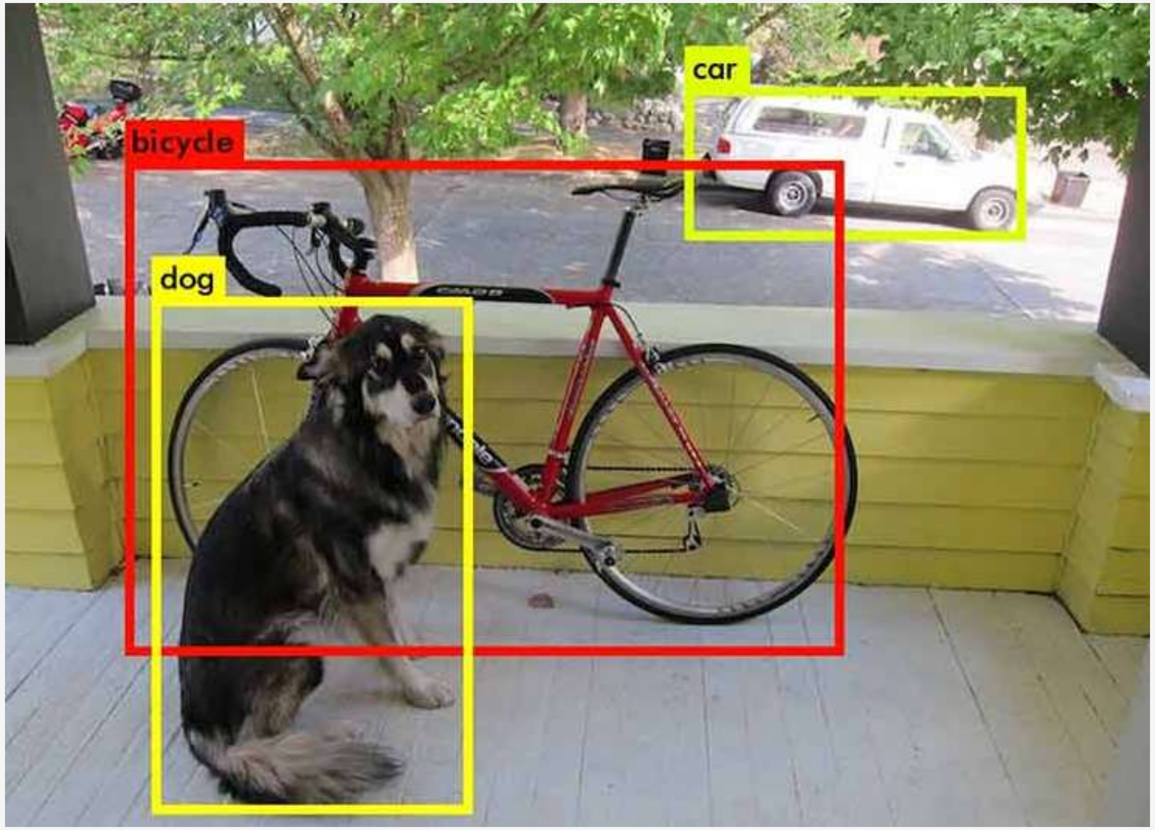
04

실습 결과

실습 소개

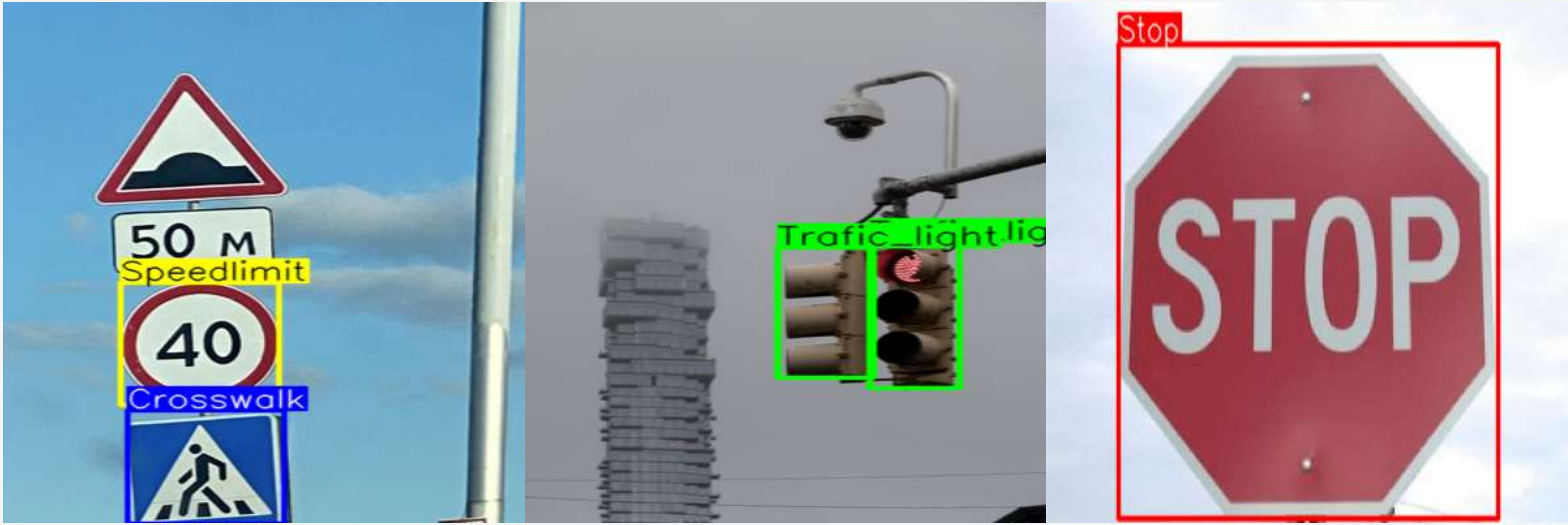
Object Detection이란?

이미지 내의 모든 Object에 대하여 Classification와 Localization을 수행



References
https://machinethink.net/blog/object-detection-with-yolo/

[실습2] Road Sign Detection



데이터셋

데이터셋 – (Road Sign Detection –Kaggle)

Road Sign Detection

877 images belonging to 4 classes.

Data Card Code (16) Discussion (0)

About Dataset

Usability
8.75

License
CC0: Public Domain

Expected update frequency
Never

Tags

References
https://www.kaggle.com/datasets/andrewmvd/road-sign-detection

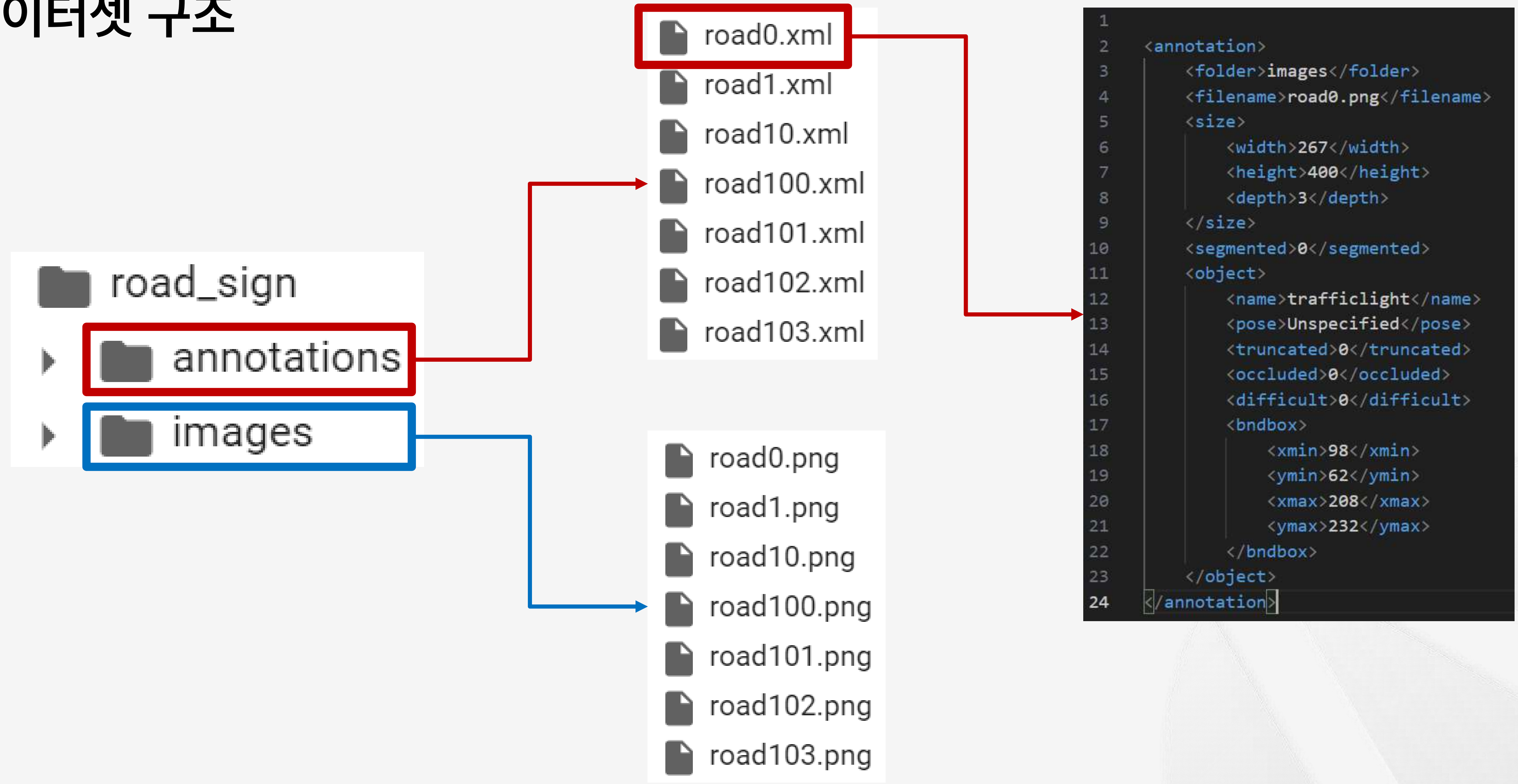
데이터셋 소개

- 데이터셋 소개 페이지 : https://www.kaggle.com/datasets/andrewmvd/road-sign-detection
- 877 Image
- Class 4개 : Trafic Light, Stop, Speedlimit, Crosswalk
- Bounding box annotations are provided in the PASCAL VOC format


```xml

<annotation>
    <folder>images</folder>
    <filename>road0.png</filename>
    <size>
        <width>267</width>
        <height>400</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>trafficlight</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <occluded>0</occluded>
        <difficult>0</difficult>
        <bndbox>
            <xmin>98</xmin>
            <ymin>62</ymin>
            <xmax>208</xmax>
            <ymax>232</ymax>
        </bndbox>
    </object>
</annotation>
```

데이터셋 구조

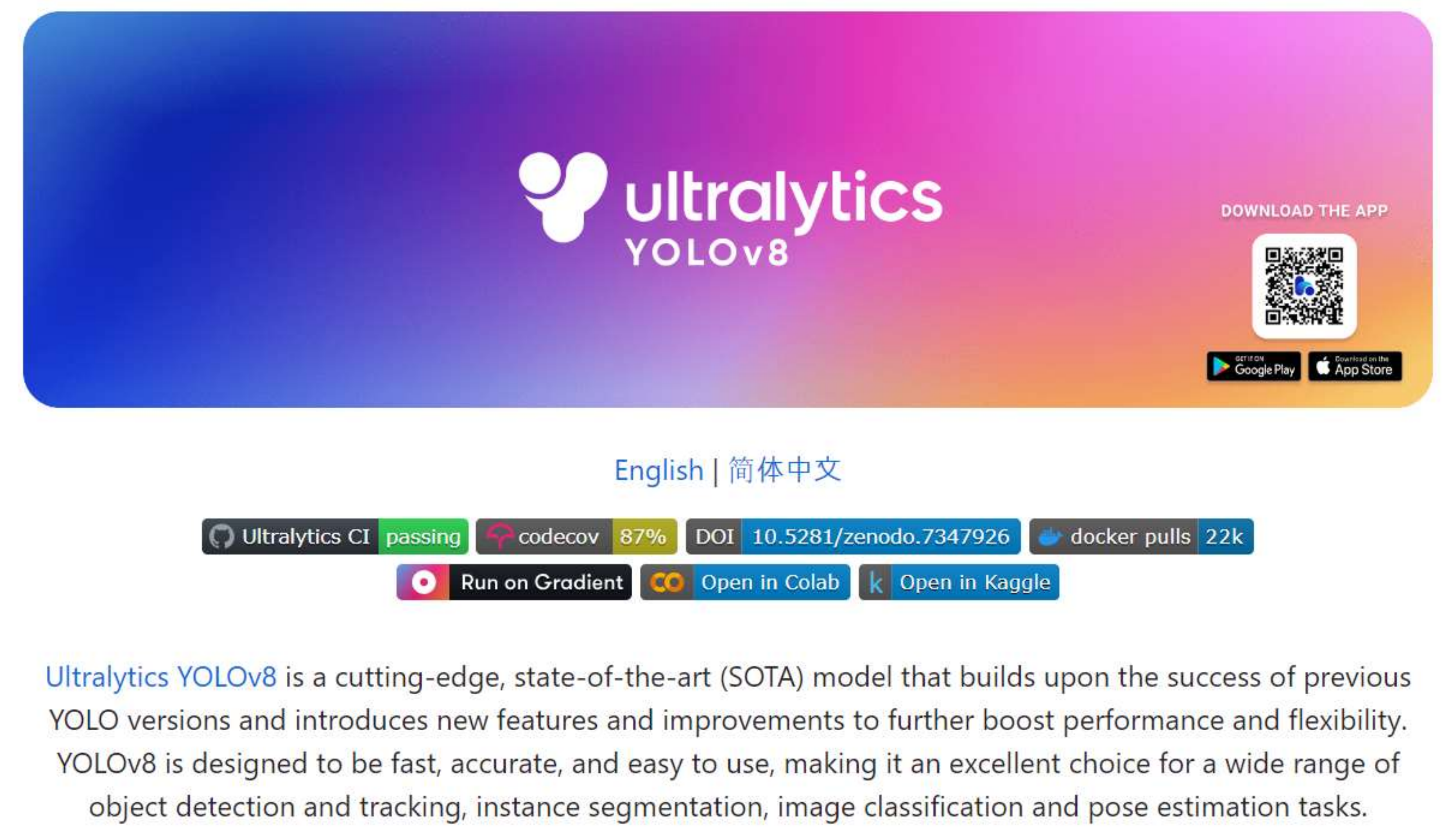

road_sign
annotations
images
road0.xml
road1.xml
road10.xml
road100.xml
road101.xml
road102.xml
road103.xml
road0.png
road1.png
road10.png
road100.png
road101.png
road102.png
road103.png
<annotation>
<folder>images</folder>
<filename>road0.png</filename>
<size>
<width>267</width>
<height>400</height>
<depth>3</depth>
</size>
<segmented>0</segmented>
<object>
<name>trafficlight</name>
<pose>Unspecified</pose>
<truncated>0</truncated>
<occluded>0</occluded>
<difficult>0</difficult>
<bndbox>
<xmin>98</xmin>
<ymin>62</ymin>
<xmax>208</xmax>
<ymax>232</ymax>
</bndbox>
</object>
</annotation>

실습 튜토리얼

YOLOv8 - ultralytics

공식 Github : https://github.com/ultralytics/ultralytics

공식 Documents : https://docs.ultralytics.com/



실습 환경 구축

- 실습 환경 구축

  pip install ultralytics

  or

  pip install git+https://github.com/ultralytics/ultralytics.git@main

- 정상 설치 확인

```
import ultralytics
ultralytics.checks()
```

```
Ultralytics YOLOv8.0.157 🚀 Python-3.10.12 torch-2.0.1+cu118 CUDA:0 (Tesla T4, 15102MiB)
Setup complete ✅ (2 CPUs, 12.7 GB RAM, 26.3/166.8 GB disk)
```

데이터셋 전처리

- Annotation convert

  Pascal VOC to Yolo

```
def xml_to_yolo_bbox(bbox, w, h):
    # xmin, ymin, xmax, ymax
    x_center = ((bbox[2] + bbox[0]) / 2) / w
    y_center = ((bbox[3] + bbox[1]) / 2) / h
    width = (bbox[2] - bbox[0]) / w
    height = (bbox[3] - bbox[1]) / h
    return [x_center, y_center, width, height]
```

+ Normalize

[xmin, ymin, xmax, ymax] → [x_center, y_center, width, height]

데이터셋 전처리

- Annotation convert

```
for fil in tqdm(files):
    basename = os.path.basename(fil)
    filename = os.path.splitext(basename)[0]
    result = []
    tree = ET.parse(fil)
    root = tree.getroot()
    width = int(root.find("size").find("width").text)
    height = int(root.find("size").find("height").text)
    for obj in root.findall('object'):
        label = obj.find("name").text
        if label not in classes:
            classes.append(label)
        index = classes.index(label)
        pil_bbox = [int(x.text) for x in obj.find("bndbox")]
        yolo_bbox = xml_to_yolo_bbox(pil_bbox, width, height)
        bbox_string = " ".join([str(x) for x in yolo_bbox])
        result.append(f"{index} {bbox_string}")
    if result:
        with open(os.path.join(label_path, f"{filename}.txt"), "w", encoding="utf-8") as f:
            f.write("\n".join(result))
```

데이터셋 전처리

- Annotation convert

  Pascal VOC to Yolo

```
<object>
    <name>Text</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <occluded>0</occluded>
    <difficult>0</difficult>
    <bndbox>
        <xmin>379</xmin>
        <ymin>1014</ymin>
        <xmax>702</xmax>
        <ymax>1077</ymax>
    </bndbox>
</object>
```

→

```
5 0.49953703703703706 0.5822185061315496 0.2990740740740741 0.03511705685618729
```

Yolo Format
Class(1) + coordinates of bounding box (4)
Class Cx Cy w h

References
https://www.researchgate.net/figure/An-example-of-conversion-from-Pascal-VOC-XML-to-YOLO-TXT_fig2_362694426

데이터셋 전처리

- Train/Test Split

```
random.shuffle(file_list)

test_ratio = 0.1
test_list = file_list[:int(len(file_list)*test_ratio)]
train_list = file_list[int(len(file_list)*test_ratio):]

for i in test_list:
  f_name = os.path.splitext(i)[0]
  copyfile(os.path.join(road_sign_path, 'images', (f_name+img_)), os.path.join(road_sign_path, 
'val/images', (f_name+img_)))
  copyfile(os.path.join(road_sign_path, 'labels', (f_name+label_)), os.path.join(road_sign_path, 
'val/labels', (f_name+label_)))
for i in train_list:
  f_name = os.path.splitext(i)[0]
  copyfile(os.path.join(road_sign_path, 'images', (f_name+img_)), os.path.join(road_sign_path, 
'train/images', (f_name+img_)))
  copyfile(os.path.join(road_sign_path, 'labels', (f_name+label_)), os.path.join(road_sign_path, 
'train/labels', (f_name+label_)))
```

Config 파일 생성

```
import yaml
data =dict()

data['train'] = '/content/drive/MyDrive/dataset/road_sign/train'
data['val'] = '/content/drive/MyDrive/dataset/road_sign/val'
data['test'] = '/content/drive/MyDrive/dataset/road_sign/val'

data['nc'] = 4
data['names'] =['Trafic_light','Speedlimit', 'Crosswalk','Stop']

with open('road_sign.yaml', 'w') as f:
    yaml.dump(data, f)
```

Train

튜토리얼 링크 : https://docs.ultralytics.com/modes/train/

```python
from ultralytics import YOLO

# Load a model
model = YOLO('yolov8n.yaml')  # build a new model from YAML
model = YOLO('yolov8n.pt')  # load a pretrained model (recommended for training)
model = YOLO('yolov8n.yaml').load('yolov8n.pt')  # build from YAML and transfer weights

# Train the model
results = model.train(data='coco128.yaml', epochs=100, imgsz=640)
```

```
yolo train data=coco.yaml
```

Train

Train Arguments

Key	Value	Description
model	None	path to model file, i.e. yolov8n.pt, yolov8n.yaml
data	None	path to data file, i.e. coco128.yaml
epochs	100	number of epochs to train for
patience	50	epochs to wait for no observable improvement for early stopping of training
batch	16	number of images per batch (-1 for AutoBatch)
imgsz	640	size of input images as integer
save	True	save train checkpoints and predict results
save_period	-1	Save checkpoint every x epochs (disabled if < 1)
cache	False	True/ram, disk or False. Use cache for data loading
device	None	device to run on, i.e. cuda device=0 or device=0,1,2,3 or device=cpu
workers	8	number of worker threads for data loading (per RANK if DDP)
project	None	project name
name	None	experiment name
exist_ok	False	whether to overwrite existing experiment
pretrained	False	whether to use a pretrained model
optimizer	'auto'	optimizer to use, choices=[SGD, Adam, Adamax, AdamW, NAdam, RAdam, RMSProp, auto]
verbose	False	whether to print verbose output
seed	0	random seed for reproducibility
deterministic	True	whether to enable deterministic mode
single_cls	False	train multi-class data as single-class
rect	False	rectangular training with each batch collated for minimum padding
cos_lr	False	use cosine learning rate scheduler

Train

Train Arguments

Key	Value	Description
cos_lr	False	use cosine learning rate scheduler
close_mosaic	10	(int) disable mosaic augmentation for final epochs (0 to disable)
resume	False	resume training from last checkpoint
amp	True	Automatic Mixed Precision (AMP) training, choices=[True, False]
fraction	1.0	dataset fraction to train on (default is 1.0, all images in train set)
profile	False	profile ONNX and TensorRT speeds during training for loggers
freeze	None	(int or list, optional) freeze first n layers, or freeze list of layer indices during training
lr0	0.01	initial learning rate (i.e. SGD=1E-2, Adam=1E-3)
lrf	0.01	final learning rate (lr0 * lrf)
momentum	0.937	SGD momentum/Adam beta1
weight_decay	0.0005	optimizer weight decay 5e-4
warmup_epochs	3.0	warmup epochs (fractions ok)
warmup_momentum	0.8	warmup initial momentum
warmup_bias_lr	0.1	warmup initial bias lr
box	7.5	box loss gain
cls	0.5	cls loss gain (scale with pixels)
dfl	1.5	dfl loss gain
pose	12.0	pose loss gain (pose-only)
kobj	2.0	keypoint obj loss gain (pose-only)
label_smoothing	0.0	label smoothing (fraction)
nbs	64	nominal batch size
overlap_mask	True	masks should overlap during training (segment train only)
mask_ratio	4	mask downsample ratio (segment train only)
dropout	0.0	use dropout regularization (classify train only)
val	True	validate/test during training

Validation

튜토리얼 링크 : https://docs.ultralytics.com/modes/val/

```
from ultralytics import YOLO

# Load a model
model = YOLO('yolov8n.pt')  # load an official model
model = YOLO('path/to/best.pt')  # load a custom model

# Validate the model
metrics = model.val()  # no arguments needed, dataset and settings remembered
metrics.box.map    # map50-95
metrics.box.map50  # map50
metrics.box.map75  # map75
metrics.box.maps   # a list contains map50-95 of each category
```

```
yolo val model=yolov8n.pt
or
model('yolov8n.pt').val()
```

Validation

Validation Arguments

Key	Value	Description
data	None	path to data file, i.e. coco128.yaml
imgsz	640	size of input images as integer
batch	16	number of images per batch (-1 for AutoBatch)
save_json	False	save results to JSON file
save_hybrid	False	save hybrid version of labels (labels + additional predictions)
conf	0.001	object confidence threshold for detection
iou	0.6	intersection over union (IoU) threshold for NMS
max_det	300	maximum number of detections per image
half	True	use half precision (FP16)
device	None	device to run on, i.e. cuda device=0/1/2/3 or device=cpu
dnn	False	use OpenCV DNN for ONNX inference
plots	False	show plots during training
rect	False	rectangular val with each batch collated for minimum padding
split	val	dataset split to use for validation, i.e. 'val', 'test' or 'train'

Inference

튜토리얼 링크 : https://docs.ultralytics.com/modes/predict/#inference-sources

```
from ultralytics import YOLO

# Load a pretrained YOLOv8n model
model = YOLO('yolov8n.pt')

# Define path to the image file
source = 'path/to/image.jpg'

# Run inference on the source
results = model(source)  # list of Results objects
```

```
from ultralytics import YOLO

# Load a pretrained YOLOv8n model
model = YOLO('yolov8n.pt')

# Run inference on 'bus.jpg' with arguments
model.predict('bus.jpg', save=True, imgsz=320, conf=0.5)
```

Inference

Attributes of Results

Attribute	Type	Description
orig_img	numpy.ndarray	The original image as a numpy array.
orig_shape	tuple	The original image shape in (height, width) format.
boxes	Boxes, optional	A Boxes object containing the detection bounding boxes.
masks	Masks, optional	A Masks object containing the detection masks.
probs	Probs, optional	A Probs object containing probabilities of each class for classification task.
keypoints	Keypoints, optional	A Keypoints object containing detected keypoints for each object.
speed	dict	A dictionary of preprocess, inference, and postprocess speeds in milliseconds per image.
names	dict	A dictionary of class names.
path	str	The path to the image file.

Inference

튜토리얼 링크 : https://docs.ultralytics.com/modes/predict/#inference-sources

```
from ultralytics import YOLO

# Load a pretrained YOLOv8n model
model = YOLO('yolov8n.pt')

# Run inference on an image
results = model('bus.jpg')  # results list

# View results
for r in results:
    print(r.boxes)  # print the Boxes object containing the detection bounding boxes
```

Inference

```
from ultralytics import YOLO
import cv2
import os
from ultralytics.yolo.utils.plotting import Annotator
import matplotlib.pyplot as plt
import numpy as np

model = YOLO('best_road_sign.pt')

root_folder = 'test'
result_folder = 'result'
test_img_list = os.listdir(root_folder)
device = 'cpu'
color_dict = [(0, 255, 0),(255, 255, 0),(0, 0, 255), (255, 0,0)]
color_dict_2 = [(0, 0, 0),(0, 0, 0),(255, 255, 255), (255, 255,255)]
```

Inference

```
for idx , file in enumerate(test_img_list):
    #black = np.zeros(shape = (400, 400,3), dtype = np.uint8)
    test_img = cv2.imread(os.path.join(root_folder, file))
    img_src = cv2.cvtColor(test_img, cv2.COLOR_BGR2RGB)
    results = model(test_img)

    for result in results:
        annotator = Annotator(img_src)
        boxes = result.boxes
        for box in boxes:
            b = box.xyxy[0]  # get box coordinates in (top, left, bottom, right) format
            cls = box.cls
            annotator.box_label(b, model.names[int(cls)], color_dict[int(cls)],color_dict_2[int(cls)])
    img_src = annotator.result()
    img_src = cv2.resize(img_src, (400,400))
    cv2.imwrite(os.path.join(result_folder, file), cv2.cvtColor(img_src, cv2.COLOR_RGB2BGR))
```

Export

- 튜토리얼 링크 : https://docs.ultralytics.com/modes/export/
- Export Arguments

Key	Value	Description
format	'torchscript'	format to export to
imgsz	640	image size as scalar or (h, w) list, i.e. (640, 480)
keras	False	use Keras for TF SavedModel export
optimize	False	TorchScript: optimize for mobile
half	False	FP16 quantization
int8	False	INT8 quantization
dynamic	False	ONNX/TensorRT: dynamic axes
simplify	False	ONNX/TensorRT: simplify model
opset	None	ONNX: opset version (optional, defaults to latest)
workspace	4	TensorRT: workspace size (GB)
nms	False	CoreML: add NMS

Export

- 튜토리얼 링크 : https://docs.ultralytics.com/modes/export/
- Export Format

Format	format Argument	Model	Metadata	Arguments
PyTorch	-	yolov8n.pt	✓	-
TorchScript	torchscript	yolov8n.torchscript	✓	imgsz, optimize
ONNX	onnx	yolov8n.onnx	✓	imgsz, half, dynamic, simplify, opset
OpenVINO	openvino	yolov8n_openvino_model/	✓	imgsz, half
TensorRT	engine	yolov8n.engine	✓	imgsz, half, dynamic, simplify, workspace
CoreML	coreml	yolov8n.mlpackage	✓	imgsz, half, int8, nms
TF SavedModel	saved_model	yolov8n_saved_model/	✓	imgsz, keras
TF GraphDef	pb	yolov8n.pb	✕	imgsz
TF Lite	tflite	yolov8n.tflite	✓	imgsz, half, int8
TF Edge TPU	edgetpu	yolov8n_edgetpu.tflite	✓	imgsz
TF.js	tfjs	yolov8n_web_model/	✓	imgsz
PaddlePaddle	paddle	yolov8n_paddle_model/	✓	imgsz
ncnn	ncnn	yolov8n_ncnn_model/	✓	imgsz, half

Export

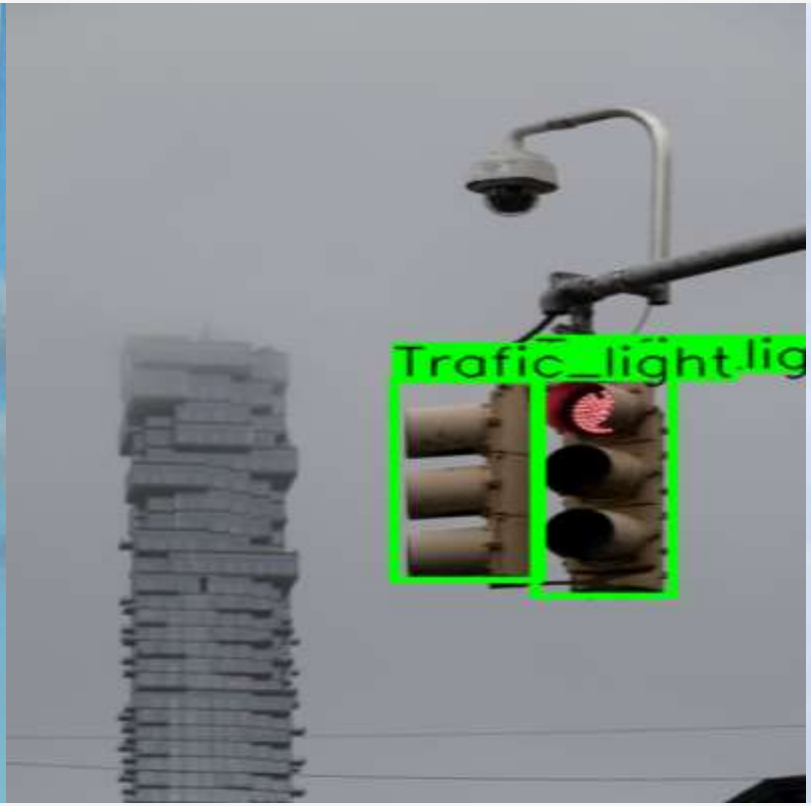
```python
from ultralytics import YOLO
model = YOLO('runs/detect/road_sign_s/weights/best.pt')
model.export(format='engine', device=0, half=False)
```

```python
#float 16
model = YOLO('runs/detect/road_sign_s/weights/best.engine')
#results = model.predict(test_img, imgsz=640, device=0, half=False)
results = model.val(data="road_sign.yaml", batch=1, imgsz=640, plots=False, device=0,
half=False, verbose=False)
metric, speed = results.results_dict['metrics/mAP50-95(B)'], results.speed['inference']
print(metric, speed)
```

```
Ultralytics YOLOv8.0.157 🚀 Python-3.10.12 torch-2.0.1+cu118 CUDA:0 (Tesla T4, 15102MiB)
Loading runs/detect/road_sign_s/weights/best.engine for TensorRT inference...
val: Scanning /content/drive/MyDrive/dataset/road_sign/val/labels.cache... 87 images, 0 backgrounds, 0 corrupt: 100%|██████████| 87/87 [00:00<?, ?it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 87/87 [00:01<00:00, 44.23it/s]
                   all         87        112      0.919      0.927      0.936      0.781
Speed: 0.3ms preprocess, 5.4ms inference, 0.0ms loss, 1.0ms postprocess per image
0.7808557837857013 5.435370850837094
```

실습 결과

실습 결과



실습 결과

Torch / TensorRT (float16)/ TensorRT (flat32)

```
Ultralytics YOLOv8.0.157 🚀 Python-3.10.12 torch-2.0.1+cu118 CUDA:0 (Tesla T4, 15102MiB)
YOLOv8s summary (fused): 168 layers, 11127132 parameters, 0 gradients
Downloading https://ultralytics.com/assets/Arial.ttf to '/root/.config/Ultralytics/Arial.ttf'...
100%|██████████| 755k/755k [00:00<00:00, 17.2MB/s]
val: Scanning /content/drive/MyDrive/dataset/road_sign/val/labels.cache... 87 images, 0 backgrounds, 0 corrupt: 100%|██████████| 87/87 [00:00<?, ?it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 87/87 [00:29<00:00,  2.90it/s]
                   all         87        112      0.918      0.917       0.94      0.791
Speed: 0.4ms preprocess, 20.1ms inference, 0.0ms loss, 1.5ms postprocess per image
```

```
Ultralytics YOLOv8.0.157 🚀 Python-3.10.12 torch-2.0.1+cu118 CUDA:0 (Tesla T4, 15102MiB)
Loading runs/detect/road_sign_s/weights/best.engine for TensorRT inference...
val: Scanning /content/drive/MyDrive/dataset/road_sign/val/labels.cache... 87 images, 0 backgrounds, 0 corrupt: 100%|██████████| 87/87 [00:00<?, ?it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 87/87 [00:01<00:00, 44.23it/s]
                   all         87        112      0.919      0.927      0.936      0.781
Speed: 0.3ms preprocess, 5.4ms inference, 0.0ms loss, 1.0ms postprocess per image
0.7808557837857013 5.435370850837094
```

```
Ultralytics YOLOv8.0.157 🚀 Python-3.10.12 torch-2.0.1+cu118 CUDA:0 (Tesla T4, 15102MiB)
Loading runs/detect/road_sign_s/weights/best.engine for TensorRT inference...
val: Scanning /content/drive/MyDrive/dataset/road_sign/val/labels.cache... 87 images, 0 backgrounds, 0 corrupt: 100%|██████████| 87/87 [00:00<?, ?it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 87/87 [00:03<00:00, 23.09it/s]
                   all         87        112      0.919      0.927      0.936      0.788
Speed: 0.4ms preprocess, 13.1ms inference, 0.0ms loss, 1.9ms postprocess per image
0.7881495053534824 13.130434628190667
```

Thank You.