BIGDATA & AI ANALYTICS
EXPERT COMPANY

One-Stage Detection
YOLO

BIGDATA & AI ANALYTICS
EXPERT COMPANY

YOLO 개요

YOLO 개요

- You Only Look Once
- One-stage object detection 딥러닝 기법
- Darknet 프레임워크 기반
- YOLO v4, v5는 오리지널 YOLO의 저자와 다른 이가 연구개발
- YOLO v5 논문은 아직 공개되지 않음
- YOLO v4는 2020년 4월에 Alexey Bochkovskiy가 발표
- YOLO v5는 2020년 6월에 Glenn Jocherk가 발표 (5월에 SOTA를 찍음)

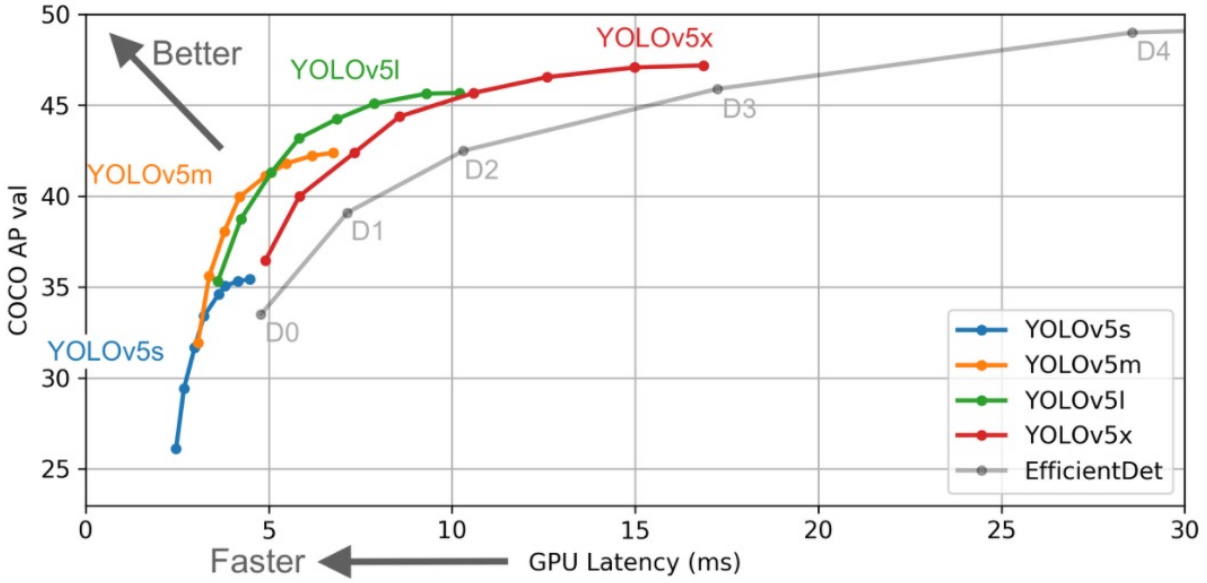
kubwa®

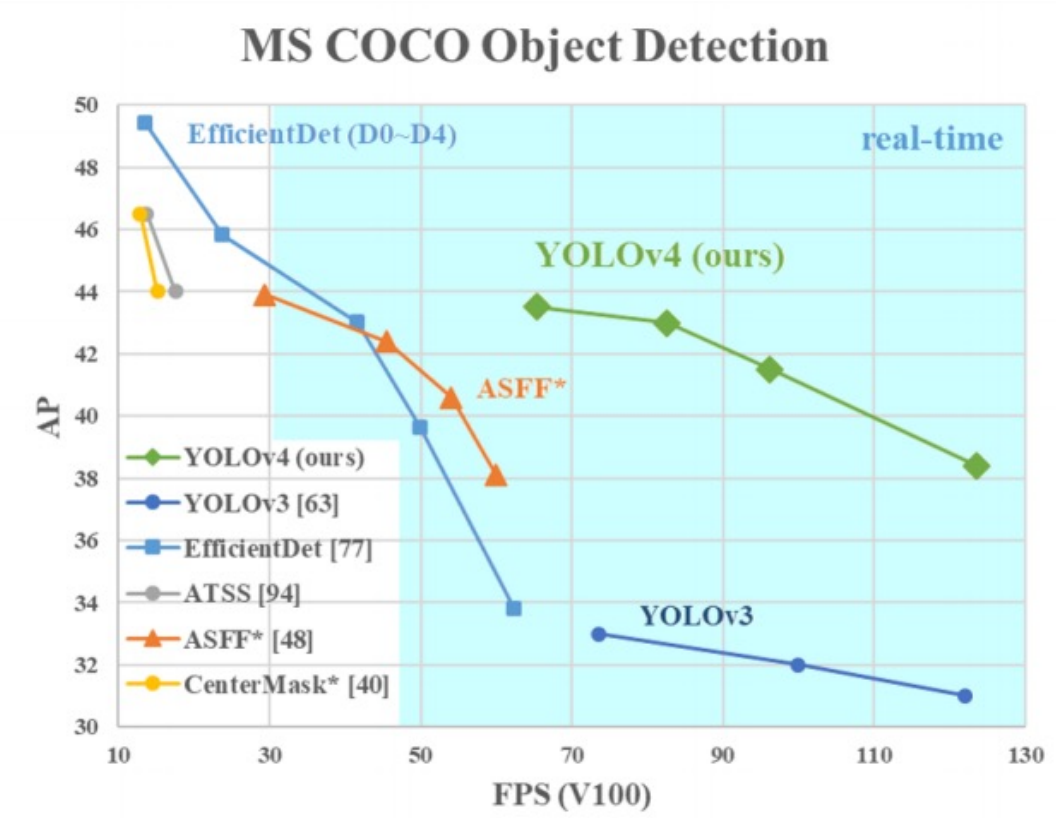
BIGDATA & AI ANALYTICS
EXPERT COMPANY

YOLO v5

YOLO v5 특징

- 정확도 측면에서는 EfficientDet이나 다른 방법에 비해 상대적으로 안좋음
- 하지만, 속도 측면에서는 압도적으로 좋음
- 기존의 YOLO가 가진 색을 많이 잃어버렸다는 평도 있음





YOLO v5 4가지 모델

- YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x
- small, medium, large, xlarge로 이름이어짐
- backbone, head는 모두 동일하지만, depth_multiple(model depth multiple)과 width_multiple(layer channel multiple)이 다름 (large가 1.0으로 기준이 되는 크기로 생각하면 좋음)
- YOLOv5s는 (0.33, 0.50), YOLOv5m은 (0.67, 0.75), YOLOv5l은 (1.0, 1.0), YOLOv5x는 (1.33, 1.25)의 비율임

Model	AP^{val}	AP^{test}	AP_{50}	$Speed_{GPU}$	FPS_{GPU}		params	FLOPs
YOLOv5-s (ckpt)	35.5	35.5	55.0	**2.1ms**	**476**		7.1M	12.6B
YOLOv5-m (ckpt)	42.7	42.7	62.4	3.2ms	312		22.0M	39.0B
YOLOv5-l (ckpt)	45.7	45.9	65.1	4.1ms	243		50.3M	89.0B
YOLOv5-x (ckpt)	**47.2**	**47.3**	**66.6**	6.5ms	153		95.9M	170.3B
YOLOv3-SPP (ckpt)	45.6	45.5	65.2	4.8ms	208		63.0M	118.0B

성능 비교

BIGDATA & AI ANALYTICS
EXPERT COMPANY

YOLO v5 파인튜닝

YOLOv5x 네트워크 기준 파인튜닝

kubwa

- Github에서 프로젝트를 clone함
- Docker로 환경 세팅하기

```
sudo docker build --tag yolo:v5 .
```

- 빌드한 이미지로 컨테이너 실행

```
sudo docker run -it --gpus all --ipc=host --name yolov5 -p 8888:8888 yolo:v5
```

- requirements로 환경 세팅하기

```
pip install -qr requirements.txt
```

YOLOv5x 네트워크 기준 파인튜닝

kubwa

- 데이터 준비하기
  - 학습을 위해 이미지와 레이블이 필요
  - 같은 이름으로 이미지 파일과 txt파일을 준비
  - 레이블 파일은 멀티클래스 형태로 구성할 수 있으며 여러 클래스 정보가 있을 때는 줄 단위로 구분해서 입력하면 됨
  - 각 줄은 클래스, 중심 x좌표, 중심 y좌표, 너비, 높이 순서로 공백으로 구분해서 정보를 입력
  - 좌표와 너비는 0~1 사이의 범위로 입력

```
ex)
3 0.5 0.5 0.2 0.2  # 3번 클래스에 해당하는 오브젝트 정보 입력
2 0.5 0.5 0.2 0.2  # 2번 클래스에 해당하는 오브젝트 정보 입력
```

YOLOv5x 네트워크 기준 파인튜닝

- 데이터경로는 다음과 같이 이미지와 레이블이 쌍을 이루도록 준비
  - 데이터 이름이 mydb라면, 아래와 같은 형태로 디렉토리 구성
  - train 폴더에 실제 이미지 파일과 레이블 txt 파일이 위치하도록 함

```
- mydb/images/train
- mydb/labels/train
```

- 검증 데이터는 아래와 같이 구성
  - val 폴더에 이미지와 레이블을 넣어주면 됨

```
- mydb_val/images/val
- mydb_val/labels/val
```

YOLOv5x 네트워크 기준 파인튜닝

kubwa

- 데이터 정보를 yaml 파일에 작성
  - Mydb.yaml 파일을 만들고 아래와 같이 코드를 작성
  - dog, cat 두 개의 클래스가 있다고 가정한 예시

```
# 학습, 검증 데이터의 이미지 경로 지정
train: ./mydb/images/train
val: ./mydb_val/images/val

# 클래스의 수
nc: 2

# 클래스 이름
names:['dog', 'cat']
```

- 위 예시와 같이 학습, 검증 세트의 이미지 데이터 경로를 지정
  - 정해진 방식대로 데이터 디렉토리를 구성하면 labels 폴더는 알아서 찾기 때문에 레이블 경로는 지정하지 않아도 됨
  - 전체 클래스의 수와 각 클래스의 이름을 지정

YOLOv5x 네트워크 기준 파인튜닝

- 모델 학습시키기
  - yaml 파일이 준비됐으면 모델을 학습시킬 시간
  - 파인튜닝 할 것이기 때문에 링크에서 pre-trained 모델을 다운로드함
  - 모델 학습을 위한 명령어

```
python train.py --img 640 --project ./result/ --cfg ./models/yolov5x.yaml --weights ./y
```

--img: 이미지 크기 지정

--project: 학습 결과가 저장될 경로 지정

--cfg: yolo 아키텍처 정보를 담은 yaml 파일 경로(여기서는 yolov5x 사용)

--weights: finetuning하는 경우 pre-trained 모델의 가중치 파일 경로

--data: 데이터 세트 정보를 담은 yaml 파일 경로

--epoch: 에포크 수

--batch: 배치 수

YOLOv5x 네트워크 기준 파인튜닝

kubwa

- 학습된 모델로 detection하기
  - detection.py 스크립트를 이용해서 학습된 모델로 Object Detection을 할 수 있음
  - Inference할 이미지는 특별한 규칙없이 한 폴더에 이미지를 준비하면 됨
  - 다음 명령어 한 줄로 Object Detection을 실행할 수 있음

```
python3 detect.py --source 이미지폴더경로 --weights 모델파일경로 --conf 0.4
```

--source: inference할 이미지 폴더 경로를 지정합니다

--weights: 학습된 모델 경로를 지정합니다.

--conf: confidence가 몇 이상인 경우 유효한 object로 간주할지 threshold 값을 설정합니다.

YOLOv5x 네트워크 기준 파인튜닝

- Detection 결과는 ./runs 폴더 하위 경로에 저장
  - 이미지 말고 텍스트로 결과를 저장하고 싶으면 save-txt 플래그를 설정
  - confidence 값도 함께 저장하고 싶으면 save-conf 플래그도 함께 설정
  - 결과 이미지는 아래와 같은 형식으로 저장됨

Yolo v5 Object Detection 결과, 출처: https://github.com/ultralytics/yolov5