주간 보고서-9주차 (2024.04.29~ 2024.05.05)

팀명 : 무계획

회의 참석자 : 서태원, 신재환, 최승렬, 류은환

회의 일자 및 회의 장소 :

2024.04.29(월)

- 16:00 ~ 17:00, 회의 장소 : 7호관 302호
- 17:30 ~ 23:00, 회의 장소 : 7호관 302호

2024.04.30(화)

- 18:00 ~ 23:00, 회의 장소 : 7호관 302호

2024.05.01(수)

- 18:30 ~ 23:00, 회의 장소 : 7호관 302호

2024.05.02(목)

- 15:00 ~ 17:30, 회의 장소 : 7호관 302호
- 18:00 ~ 21:30, 회의 장소 : 7호관 302호

2024.05.03(금)

- 13:00 ~ 16:00, 회의 장소 : 7호관 302호

총 활동 시간 : 24시간

진행 사항 :

현재 역할

서태원

[업무]

류은환

[업무]

- Conv 구조 파헤치기

- yolov9 제공 가중치 분석하기

신재환

[업무]

- .pt 파일 뜯는 법 정리
- onnx quantizaiton

최승렬

[업무]

- Dataset Class filtering 툴 개발
- onnx to ncnn

---

test1모델을 MS COCO dataset을 사용하여 epoch을 300번 실행하면서 mAP를 확인하였고, dataset을 COCO128을 사용한 것과 비교함

test1_COCO_300	test1_128_1000
mAP50: 0.636	mAP50 : 0.95
mAP50-95: 0.465	mAP50-95 : 0.86

test7 mAP 개선 시도

1. test7모델을 MS COCO dataset을 사용하여 epoch 수를 300번과 500번으로 설정하고 mAP를 확인

test7_COCO_300	test7_COCO_500
mAP50: 0.381	mAP50: 0.387
mAP50-95: 0.252	mAP50-95: 0.257

epoch을 200번 더 늘렸지만 mAP50는 0.006, mAP50-95는 0.005 증가한 것을 확인함.

2. test7의 하이퍼 파라미터를 변경 (Optimizer 함수 변경) → test8 모델로 만들어둠

test8

- **test7에서 학습 파라미터를 SGD에서 Adam 용으로 변경**
- train.py 실행 시 --optimizer Adam을 명령어에 추가

```
lr0: 0.001  # initial learning rate (SGD=1E-2, Adam=1E-3)
lrf: 0.001  # final OneCycleLR learning rate (lr0 * lrf)
momentum: 0.9  # 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
cls_pw: 1.0  # cls BCELoss positive_weight
obj: 0.7  # obj loss gain (scale with pixels)
obj_pw: 1.0  # obj BCELoss positive_weight
dfl: 1.5  # dfl loss gain
iou_t: 0.20  # IoU training threshold
anchor_t: 5.0  # anchor-multiple threshold
# anchors: 3  # anchors per output layer (0 to ignore)
fl_gamma: 0.0  # focal loss gamma (efficientDet default gamma=
hsv_h: 0.015  # image HSV-Hue augmentation (fraction)
hsv_s: 0.7  # image HSV-Saturation augmentation (fraction)
hsv_v: 0.4  # image HSV-Value augmentation (fraction)
degrees: 0.0  # image rotation (+/- deg)
translate: 0.1  # image translation (+/- fraction)
scale: 0.9  # image scale (+/- gain)
shear: 0.0  # image shear (+/- deg)
perspective: 0.0  # image perspective (+/- fraction), range 0
flipud: 0.0  # image flip up-down (probability)
fliplr: 0.5  # image flip left-right (probability)
mosaic: 1.0  # image mosaic (probability)
mixup: 0.15  # image mixup (probability)
copy_paste: 0.3  # segment copy-paste (probability)
```

Dataset의 Class를 filtering하는 코드를 작성하고 파일을 생성

처음 data.yaml파일에서 nc(number of class)를 71개에서 1개로 줄이고, class 목록들을 ‘person’만 남긴 상태로 수정하였으나 아래와 같은 에러가 발생

```
assert mlc < nc, f'Label class {mlc} exceeds nc={nc} in {data
AssertionError: Label class 70 exceeds nc=1 in /home/mgh/MGH/
```

그래서 data.yaml에서 데이터 class를 직접 건드리지 않고, 데이터셋의 라벨에서 원하는 라벨만 filtering하도록 바꿔주도록 방향성을 잡음.

```
import os

def filter_labels(directory):
    for filename in os.listdir(directory):
        if filename.endswith(".txt"):
            path = os.path.join(directory, filename)
            with open(path, 'r') as file:
                lines = file.readlines()
            with open(path, 'w') as file:
                for line in lines:
                    if line.strip().split()[0] == "42":  # cl
                        file.write(line)


        LOGGER.info('Filtering labels...')
    filter_labels('/home/mgh/MGH/yolov9-pre/COCO128/train/lab
    filter_labels('/home/mgh/MGH/yolov9-pre/COCO128/valid/lab
    filter_labels('/home/mgh/MGH/yolov9-pre/COCO128/test/labe
    LOGGER.info('Labels filtered.')
```

train과정 전에 원하는 class만 필터링하는 함수를 추가하여 학습시키고자 하는 데이터셋 class만을 filtering함

- filtering_class.py

```
import os
import logging

logging.basicConfig(level=logging.INFO, format='%(asctime)s -

def filter_labels(directory):
```

```
    count_files = 0
    count_lines = 0
    for filename in os.listdir(directory):
        if filename.endswith(".txt"):
            path = os.path.join(directory, filename)
            with open(path, 'r') as file:
                lines = file.readlines()
            with open(path, 'w') as file:
                for line in lines:
                    if line.strip().split()[0] == "42":  # 클래
                        file.write(line)
                        count_lines += 1
            count_files += 1
    return count_files, count_lines

def main():
    train_labels = '/home/mgh/MGH/yolov9-pre/COCO128/train/la
    valid_labels = '/home/mgh/MGH/yolov9-pre/COCO128/valid/la
    test_labels = '/home/mgh/MGH/yolov9-pre/COCO128/test/labe

    logging.info('Filtering training labels...')
    train_files, train_filtered = filter_labels(train_labels)
    logging.info(f'Filtered {train_filtered} lines in {train_

    logging.info('Filtering validation labels...')
    valid_files, valid_filtered = filter_labels(valid_labels)
    logging.info(f'Filtered {valid_filtered} lines in {valid_

    logging.info('Filtering test labels...')
    test_files, test_filtered = filter_labels(test_labels)
    logging.info(f'Filtered {test_filtered} lines in {test_fi


if __name__ == "__main__":
    main()
```

filterling하는 함수 코드를 train, valid, test 코드 안에 넣지 않고, 따로 파일을 하나 만들어서 class를 filtering하도록 함

- filtering_copy_dataset.py

```
import os
import shutil
import logging

# 로깅 설정
logging.basicConfig(level=logging.INFO, format='%(asctime)s -

def create_directory(path):
    if not os.path.exists(path):
        os.makedirs(path)
        logging.info(f"Created directory {path}")

def filter_and_copy_labels(src_directory, dst_directory, clas
    count_files = 0
    count_lines = 0
    valid_files = []

    # 필터링된 라벨 저장 디렉토리 생성
    create_directory(dst_directory)

    for filename in os.listdir(src_directory):
        if filename.endswith(".txt"):
            src_path = os.path.join(src_directory, filename)
            dst_path = os.path.join(dst_directory, filename)
            with open(src_path, 'r') as file:
                lines = file.readlines()
            with open(dst_path, 'w') as file:
                for line in lines:
                    class_id = line.strip().split()[0]
                    if class_id in classes:
                        file.write(line)
                        count_lines += 1

            if count_lines > 0:
                valid_files.append(filename.replace('.txt', '
```

```
            count_files += 1

    return count_files, count_lines, valid_files

def copy_images(src_img_dir, dst_img_dir, valid_files):
    create_directory(dst_img_dir)
    for img_name in valid_files:
        for ext in ['.jpg', '.jpeg', '.png']:
            src_img_path = os.path.join(src_img_dir, img_name
            if os.path.exists(src_img_path):
                shutil.copy(src_img_path, os.path.join(dst_im
                break

def main():
    classes_to_keep = {"42"}
    base_dir = "/home/mgh/MGH/yolov9-pre/COCO128"
    src_dirs = {
        'train': os.path.join(base_dir, 'train'),
        'valid': os.path.join(base_dir, 'valid'),
        'test': os.path.join(base_dir, 'test')
    }
    dst_dirs = {
        'train': os.path.join(base_dir, 'filtered/train'),
        'valid': os.path.join(base_dir, 'filtered/valid'),
        'test': os.path.join(base_dir, 'filtered/test')
    }

    for key in src_dirs:
        logging.info(f"Processing {key} data...")
        label_files, line_count, valid_images = filter_and_co
        logging.info(f"Filtered {line_count} lines in {label_
        copy_images(src_dirs[key], dst_dirs[key], valid_image
        logging.info(f"Copied {len(valid_images)} images.")

if __name__ == "__main__":
    main()
```

class filtering을 거친 후에 원본 데이터셋과 분리되도록 filtering된 데이터셋 폴더를 하나 새로 생성함

Quantization(양자화)

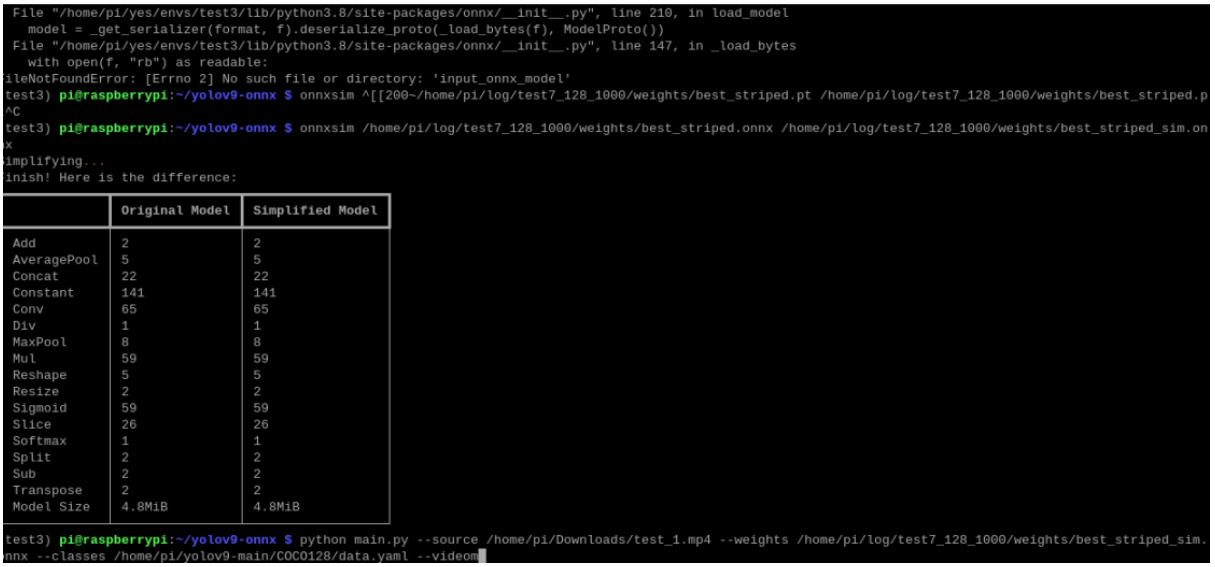
- **동적 vs 정적 양자화**: 정적 양자화는 모델의 모든 가중치를 사전에 고정된 형식으로 변환하지만, 이는 모델의 동적 특성을 제한할 수 있음. 동적 양자화는 실행 시간에 가중치를 양자화하여 더 유연하지만, 계산 오버헤드가 발생할 수 있음.
- **데이터 분포**: 양자화는 입력 데이터의 분포가 양자화 중에 사용된 데이터의 분포와 일치하지 않을 경우, 성능 저하를 일으킬 수 있음. 데이터의 분포가 변하면 모델의 정확도와 성능에 부정적인 영향을 미칠 수 있음.

```
import onnx
from onnxruntime.quantization import quantize_dynamic, QuantT
import onnxruntime as ort


model_fp32 = '/home/pi/log/test3_128_1000/weights/best.onnx'
model_quant = '/home/pi/log/test3_128_1000/weights/best_quant


quantize_dynamic(model_fp32, model_quant, weight_type=QuantTy
```

동적 양자화로 32bit에서 8bit로 변환



test7을 onnxsim 후에 모델을 변경하고자 했으나 실제로 확인하였을 때 변한 것이 없었음.

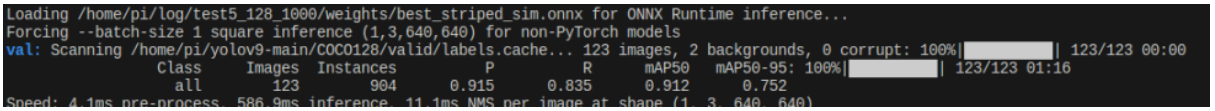
	Original Model	Simplified Model
Add	2	2
AveragePool	5	5
Concat	22	22
Constant	141	141
Conv	65	65
Div	1	1
MaxPool	8	8
Mul	59	59
Reshape	5	5
Resize	2	2
Sigmoid	59	59
Slice	26	26
Softmax	1	1
Split	2	2
Sub	2	2
Transpose	2	2
Model Size	4.8MiB	4.8MiB

```
Loading /home/pi/log/test7_128_1000/weights/best_striped_sim.onnx for ONNX Runtime inference...
Forcing --batch-size 1 square inference (1,3,640,640) for non-PyTorch models
val: Scanning /home/pi/yolov9-main/COCO128/valid/labels.cache... 123 images, 2 backgrounds, 0 corrupt: 100%|██████████| 123/123 00:00
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100%|██████████| 123/123 00:48
                   all        123        904      0.818      0.749      0.823      0.605
Speed: 5.9ms pre-process, 353.0ms inference, 12.9ms NMS per image at shape (1, 3, 640, 640)
```

기존의 test7에 비해서 mAP는 그대로였으나 inference는 오히려 근소하게 증가됨.

하지만 test5로 변경해서 돌려보니 변화가 있는 것을 확인

	Original Model	Simplified Model
Add	15	2
AveragePool	5	5
Concat	22	22
Constant	225	141
Conv	65	65
Div	14	1
Gather	13	0
MaxPool	8	8
Mul	85	59
Pad	5	0
Reshape	5	5
Resize	2	2
Shape	13	0
Sigmoid	59	59
Slice	26	26
Softmax	1	1
Split	2	2
Sub	2	2
Transpose	2	2
Model Size	10.1MiB	10.0MiB



그렇지만 기존의 test5와 비교해서 mAP는 그대로이고, influence는 미미하게 증가됨.

.pt 파일의 구조와 파라미터를 분석하고자 파일을 열어봄

pytorch_open.py

```
import torch

model_path = '/home/pi/yolov9-main/0411_1/weights/best.pt'
model = torch.load(model_path)
print(model)
```

단 yolo 폴더에서 실행해야 함.

터미널에서 출력값을 txt로 저장하는 법

```
python pytorch_open.py > output.txt
```

Colab 실행 코드

```
import torch
import sys

sys.path.append('/content/drive/MyDrive/yolov9-main')
model_path = '/content/drive/MyDrive/yolov9-c.pt'
model = torch.load(model_path)
print(model)
```

```
import torch

#model_path = '/home/pi/yolov9-main/0411_1/weights/best.pt'
#model = torch.load(model_path)
model = torch.load('/home/pi/yolov9-main/0411_1/weights/' + '
model.state_dict()

#model.load_state_dict(torch.load('/home/pi/yolov9-main'+ 'mo

model = torch.hub.load('ultralytics/yolov5', 'custom', path='

for name, param in model.named_parameters():
    print(f"Layer: {name} | Size: {param.size()} | Values : \
```

논문에서 제공된 yolov9.pt 파일을 분석하고, 실제로 돌려본 값들과 비교함

Model	Test Size	AP^{val}	AP_{50}^{val}	AP_{75}^{val}	Param.	FLOPs
YOLOv9-T	640	38.3%	53.1%	41.3%	2.0M	7.7G
YOLOv9-S	640	46.8%	63.4%	50.7%	7.1M	26.4G
YOLOv9-M	640	51.4%	68.1%	56.1%	20.0M	76.3G
YOLOv9-C	640	53.0%	70.2%	57.8%	25.3M	102.1G
YOLOv9-E	640	55.6%	72.8%	60.6%	57.3M	189.0G

논문에서의 결과값

논문에서 제공하는 가중치로 MS COCO dataset 사용하여 mAP를 확인함

YOLOv9-c(Paper DownLoad)	YOLOv9-c (Paper DownLoad) 1 epoch
mAP50: 0.000388	mAP50 : 0.616
mAP50-95: 0.000297	mAP50-95 : 0.452

YOLOv9-c-converted (Paper DownLoad)	YOLOv9-c-converted (Paper DownLoad) 1 epoch
mAP50: 0.000388	mAP50 : 0.0322
mAP50-95: 0.000297	mAP50-95 : 0.0164

논문 제공 파일들을 사용하여 MS COCO에 대해 확인하였을 때 제데로 된 수치가 나오지 않아 전이 학습(1 epoch)을 시킨 뒤 비교함

yolov9-c-converted는 multi-level reversible auxiliary branch가 삭제된 모델

비고

- onnx 형식로 변환할 때 pruning과 quantization을 하면서 export하고자 했으나 실패, 그래서 onnx변환 후에 pruning과 quantization을 하고자 함
- LINC3.0사업단에서 라즈베리파이 5 구매 완료

추후 계획

- 캡스턴 중간 발표 구성 내용과 발표 흐름 정하기
- Conv를 바꾼 모델(test1 및 이후)의 아키텍처 영향을 정리하고, 왜 수정해도 되는지에 대한 분석 내용 정리하기