무[無]계획

서태원, 신재환, 류은환, 최승렬

인천대학교 INCHEON NATIONAL UNIVERSITY | 임베디드시스템공학과

Contents

- 프로젝트 개요
- 프로젝트 목표
- 연구 과정 및 실험 결과
  - 아키텍처 설계
  - 후처리 적용
- 추가 확인 사항
- 기존 계획서와의 비교
- 프로젝트 추진 기록
  - 팀원별 역할
  - 활동 시간
- 연구 기록 리뷰

프로젝트 개요

프로젝트 개요

딥러닝 모델의 최적화를 통한

임베디드보드에서의 추론 속도 향상

프로젝트 개요

일반적으로 고성능 딥러닝 모델은
GPU나 GPU서버 같은 **고사양의 컴퓨팅 자원**을 필요로 함

GPU 자원 X
Network 자원 X

임베디드 보드와 같은 제한된 환경
딥러닝 모델을 구동하기 어려움

프로젝트 개요

온디바이스 **AI** 추세에 따라
딥러닝 모델도 **Edge device** 에서 구동하는 것이 트렌드

고성능 GPU 설치, Network 사용
→ 비효율적, 보안 문제

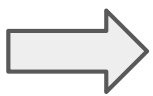
따라서, **딥러닝 모델을 경량화** 하는 방법을 탐색하고,
이를 바탕으로 **모델 구조 설계** 및 **추론 속도 향상**을 목표

프로젝트 개요

경량화 대상은 Object Detection 모델로 결정

영상/이미지를 활용하기에 많은 컴퓨팅 자원이 필요한 분야

임베디드 보드 환경
구동 어려움



다양한 응용 작업에
적용 가능

연구) 제한된 환경에서의 모델 경량화 방법에 대한 방향 탐색 및 실험
응용) 세부 활용 분야에 적용하여 **사회적 가치 창출을 위한 토대** 마련

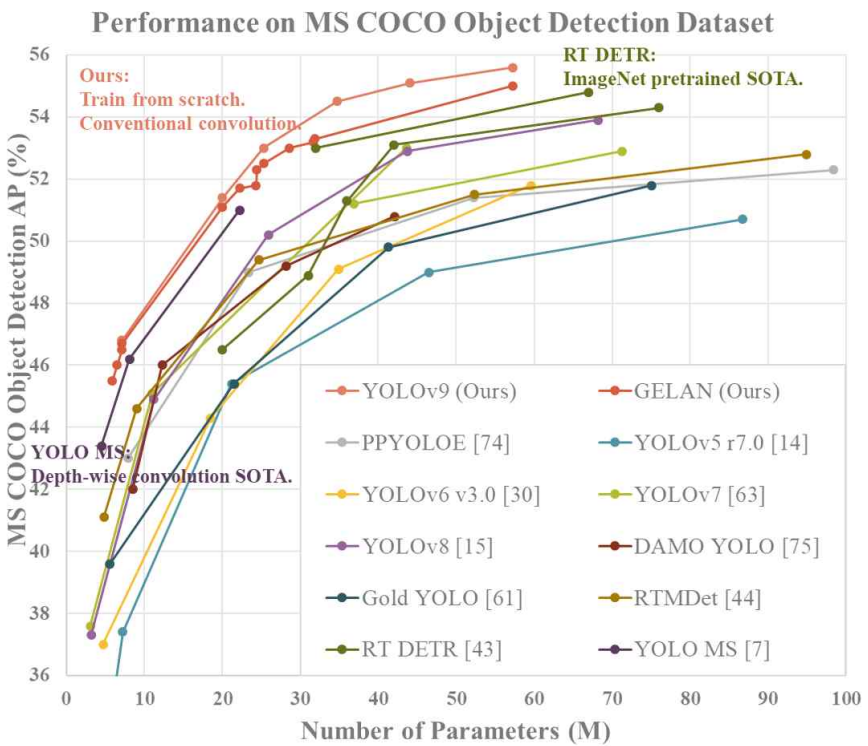
프로젝트 개요

기준 모델 YOLOv9 결정

최신 YOLO Series, 2024년 2월 공개

논문 및 코드 공개, 기존 시리즈 존재

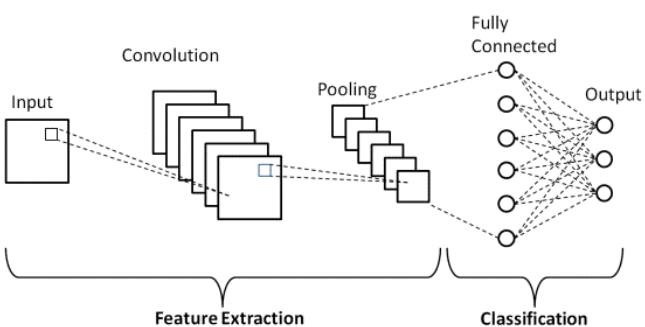
논문 및 코드 분석 연구
기존모델간 비교를 통한 아이디어



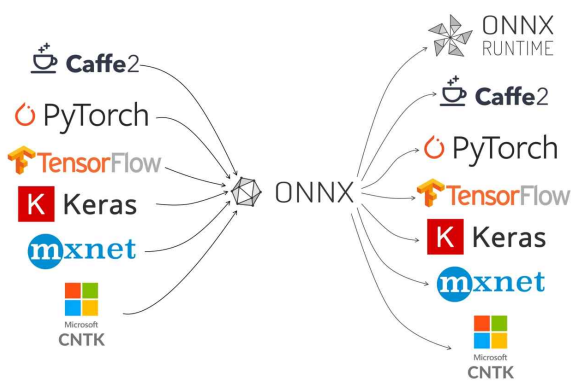
프로젝트 목표

프로젝트 목표

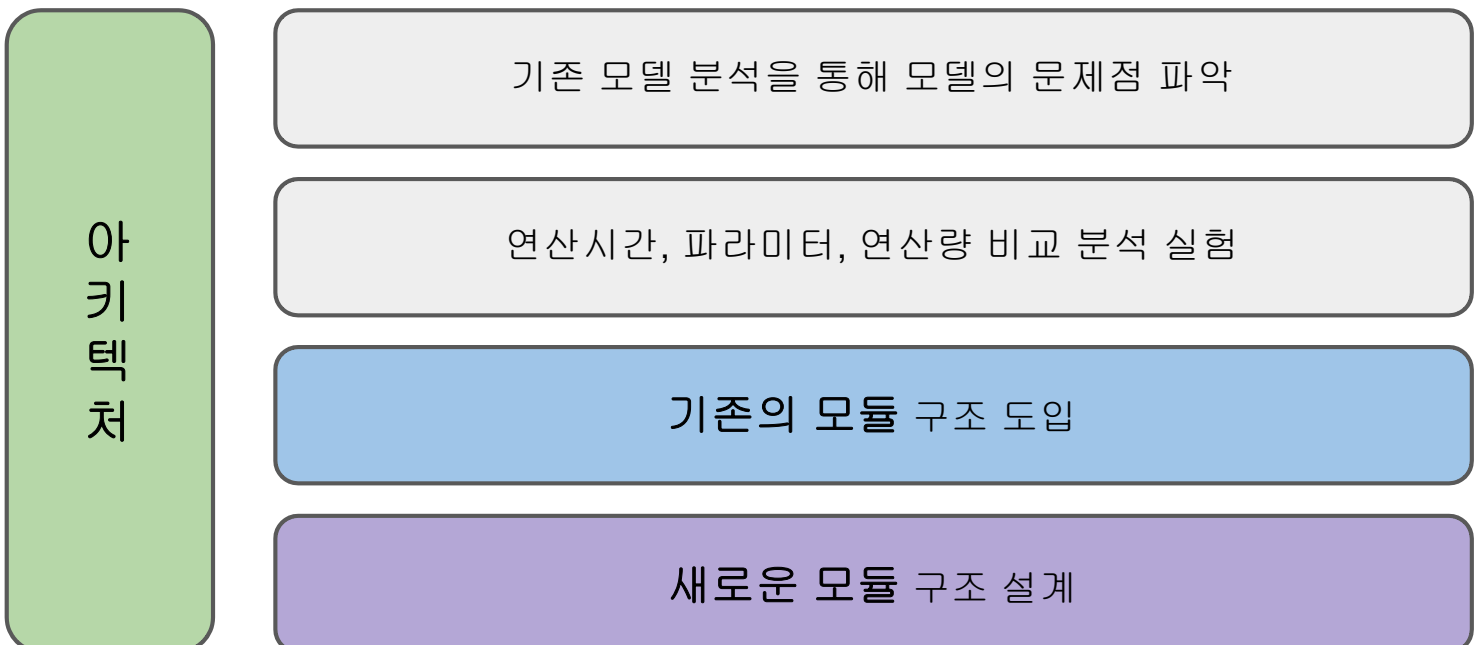
아키텍처
설계



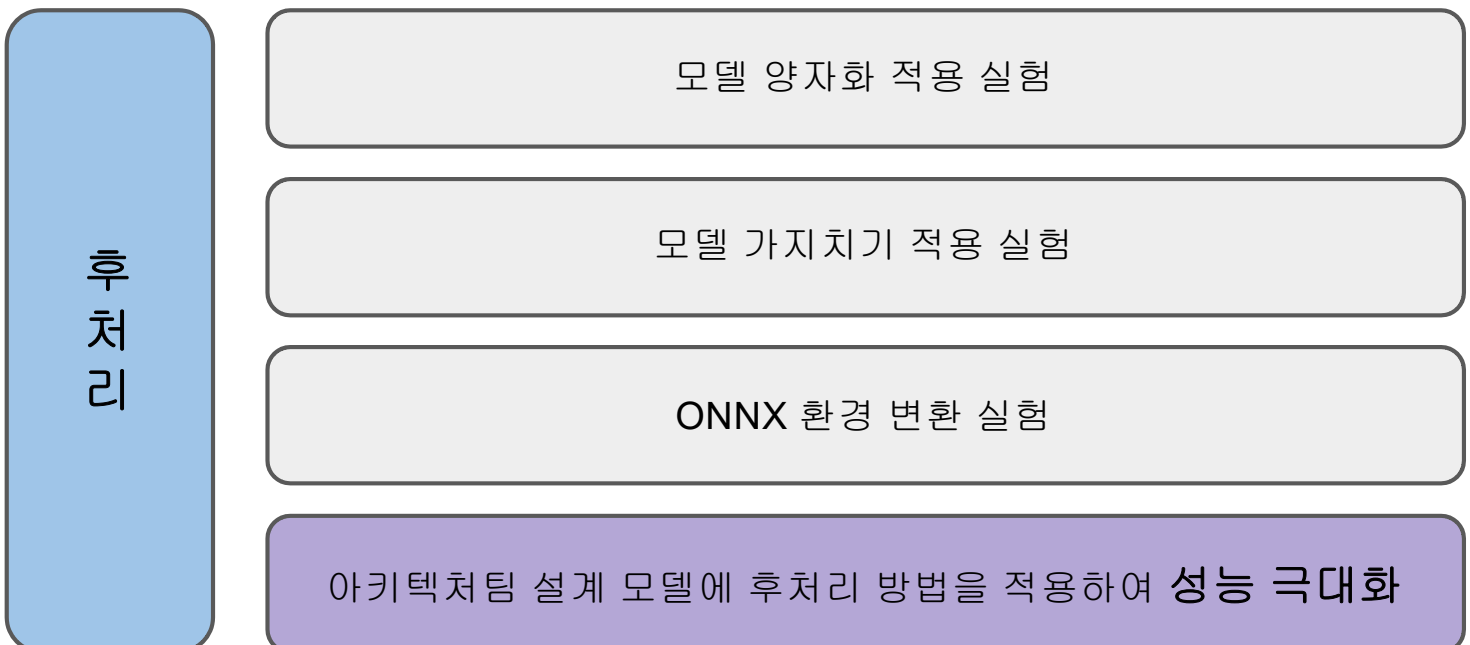
후처리
적용



프로젝트 목표


아키텍처
기존 모델 분석을 통해 모델의 문제점 파악
연산시간, 파라미터, 연산량 비교 분석 실험
기존의 모듈 구조 도입
새로운 모듈 구조 설계

프로젝트 목표


후처리
모델 양자화 적용 실험
모델 가지치기 적용 실험
ONNX 환경 변환 실험
아키텍처팀 설계 모델에 후처리 방법을 적용하여 성능 극대화

프로젝트 목표

주제 신청서

딥러닝 모델에 경량화 적용
저사양 임베디드 보드에서 모델 구동 및 성능 비교

중간 발표

YOLOv9 Tiny 수준의 성능 달성
: mAP 하락 폭이 크단 피드백

응용 고려

쓸만한 정도의 성능을 달성?
타당한 기준 점 설정의 어려움

mAP를 최대한 유지하면서
동시에 추론속도를 향상시키는 것

연구의 특성상 프로젝트의 진행에 따라 배우는 내용이 많아짐

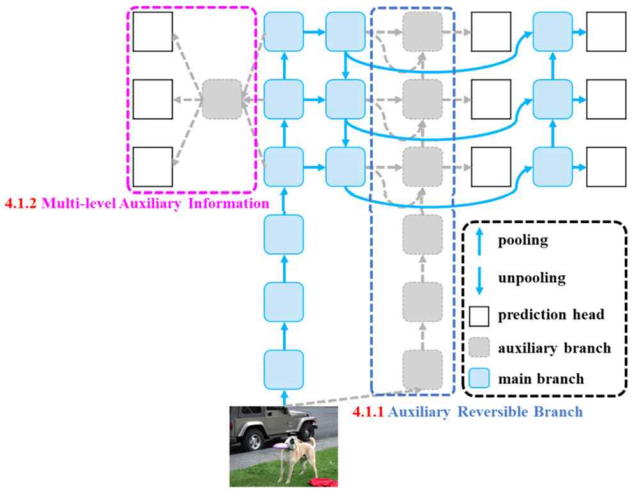
더 많은 지식과, 발표 피드백을 통해
지속적으로 목표를 수정하며 프로젝트를 진행

연구 과정 및 실험 결과

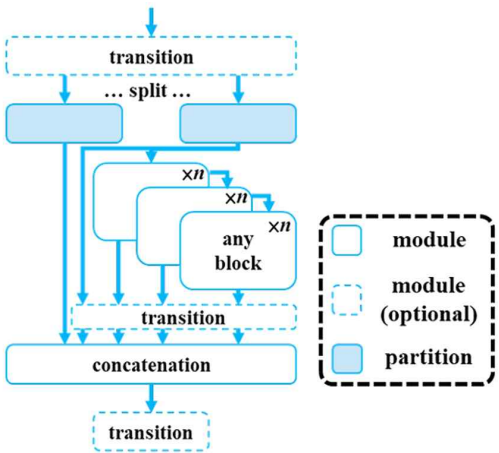
아키텍처 설계

YOLOv9 모델 분석

논문 분석
재현 실험



(d) Programmable Gradient Information



(c) GELAN

PGI
(Programmable Gradient Information)

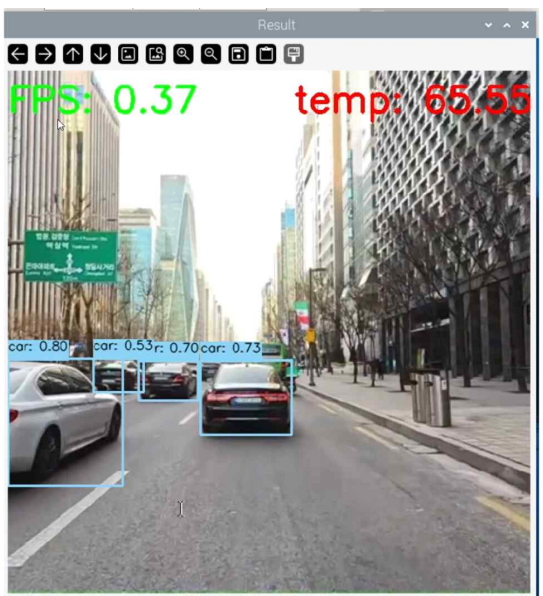
Information bottleneck 완화
추론 시 보조 분기를 제거
추론 속도 향상

GELAN
(Generalized ELAN)

ELAN의 연산 블록에
확장성을 더함
학습 작업에 따라 적합한 블록 교체

기존 모델의 문제점

yolov9-Compact 시연 영상



기존 모델은 **RPi 5** 임베디드 보드 환경에서 정상적 구동이 어려움 **(FPS: 0.37)**

문제점 찾기 - 레이어 연산시간 분석

YOLOv9-c를 구성하는 모든 레이어에 대한 분석이 필요 판단
각 레이어의 연산시간과 파라미터 수, 연산량 분석 코드 작성

분석 코드를 이용하여
RPi 5 에서 각 레이어의 연산 수행 시간을 측정함

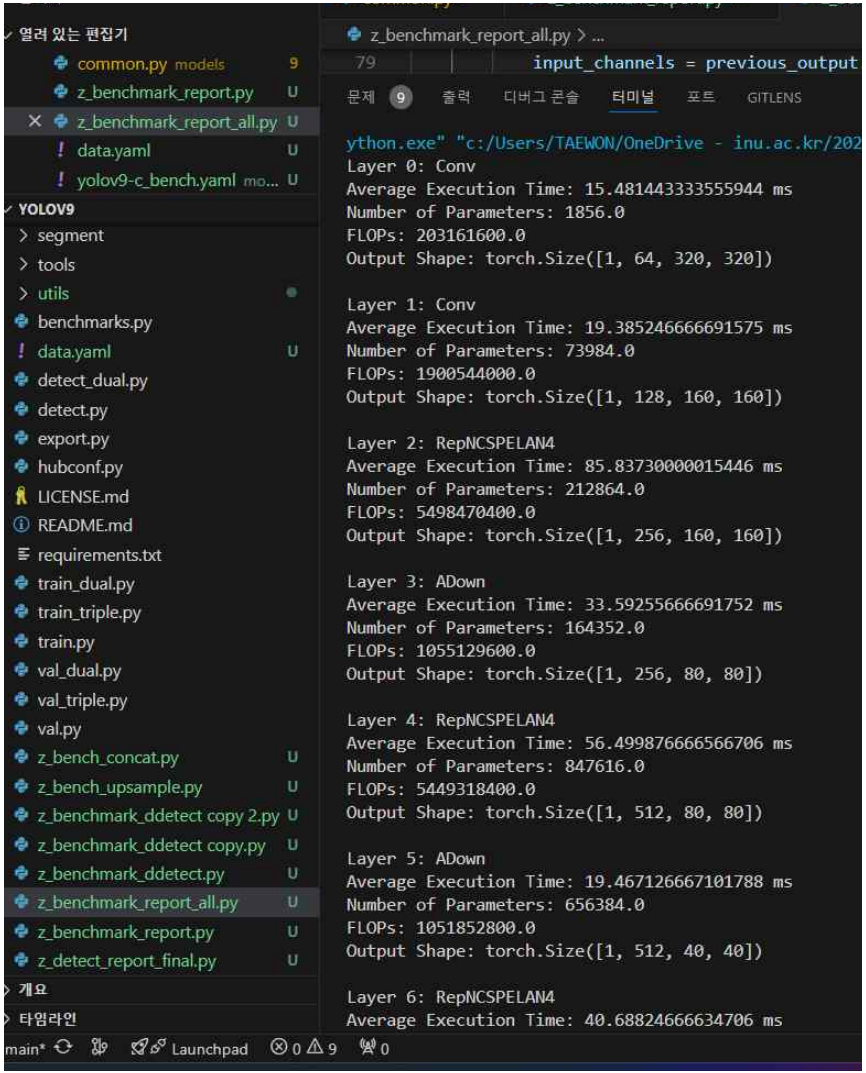
분석 코드

레이어마다 적절한 입력을 생성하여 **자동으로 전달**
연산시간 측정은 **50번 수행시간 측정 결과의 10회 평균치 적용**
연산시간, 파라미터 수, 연산량 등 **측정 지표 결과 리포트**

레이어 연산시간 분석

```
pi@raspberrypi: ~/MGH/yolov9-main
File  Edit  Tabs  Help
(test) pi@raspberrypi:~/MGH/yolov9-main $ python z_benchmark.py
/home/pi/miniforge3/envs/test/lib/python3.7/site-packages/torchvision/io/image.py:13
: UserWarning: Failed to load image Python extension:
  warn(f"Failed to load image Python extension: {e}")
Enter the input size, channels, layer name, and layer arguments separated by commas:
 80, 256, RepNCSPELAN4, 256, 512, 256, 128, 1
Average Execution Time: 367.90977693333764 ms
Average Execution Time: 370.50416326666584 ms
Average Execution Time: 382.00743619999 ms
Average Execution Time: 384.80526646667386 ms
Average Execution Time: 385.35001146666065 ms
Average Execution Time: 405.02152569999527 ms
Average Execution Time: 397.2781744333323 ms
Average Execution Time: 402.7189486333403 ms
Average Execution Time: 390.5775369333507 ms
Average Execution Time: 402.1122529666703 ms
User Input: 80, 256, RepNCSPELAN4, 256, 512, 256, 128, 1
BenchMark Module Name: RepNCSPELAN4
Total Average Execution Time: 388.8285093000017 ms
Number of Parameters: 847616.0
FLOPs: 5449318400.0
Output Shape: torch.Size([1, 512, 80, 80])
(test) pi@raspberrypi:~/MGH/yolov9-main $
```

실제 연산시간 측정 과정
이를 바탕으로 모델 구조 분석 수행



레이어 연산시간 분석

BACKBONE

Layer Name	연산 시간	Params	GFlops
(0)Conv	74.09ms	1856	0.2031616
(1)Conv	118.00ms	73984	1.9005440
(2)RepNCSPELAN4	565.94ms	212864	5.4984704
(3)ADown	91.91ms	164352	1.0551296
(4)RepNCSPELAN4	357.29ms	847616	5.4493184
(5)ADown	57.98ms	656384	1.0518528
(6)RepNCSPELAN4	217.60ms	2857472	4.5826048
(7)ADown	14.46ms	656384	0.2629632
(8)RepNCSPELAN4	54.59ms	2857472	1.1456512

HEAD

Layer Name	연산 시간	Params	GFlops
(9)SPPELAN	16.59ms	656896	0.2633728
(10)nn.Upsample	2.63ms	0	0.0008192
(11)Concat	2.77ms	0	0
(12)RepNCSPELAN4	228.33ms	3119616	5.0020352
(13)nn.Upsample	7.78ms	0	0.0032768
(14)Concat	9.25ms	0	0
(15)RepNCSPELAN4	362.29ms	912640	5.8621952
(16)ADown	24.18ms	164352	0.2637824
(17)Concat	3.26ms	0	0
(18)RepNCSPELAN4	224.27ms	2988544	4.7923200
(19)ADown	14.72ms	656384	0.2629632
(20)Concat	0.99ms	0	0
(21)RepNCSPELAN4	57.47ms	3119616	1.2505088

주요 구조인 **RepNCSPELAN4**는 연산 구조가 복잡해, **RPi 5**에서 구동하기에 부적합 판단

아키텍처 수정 실험

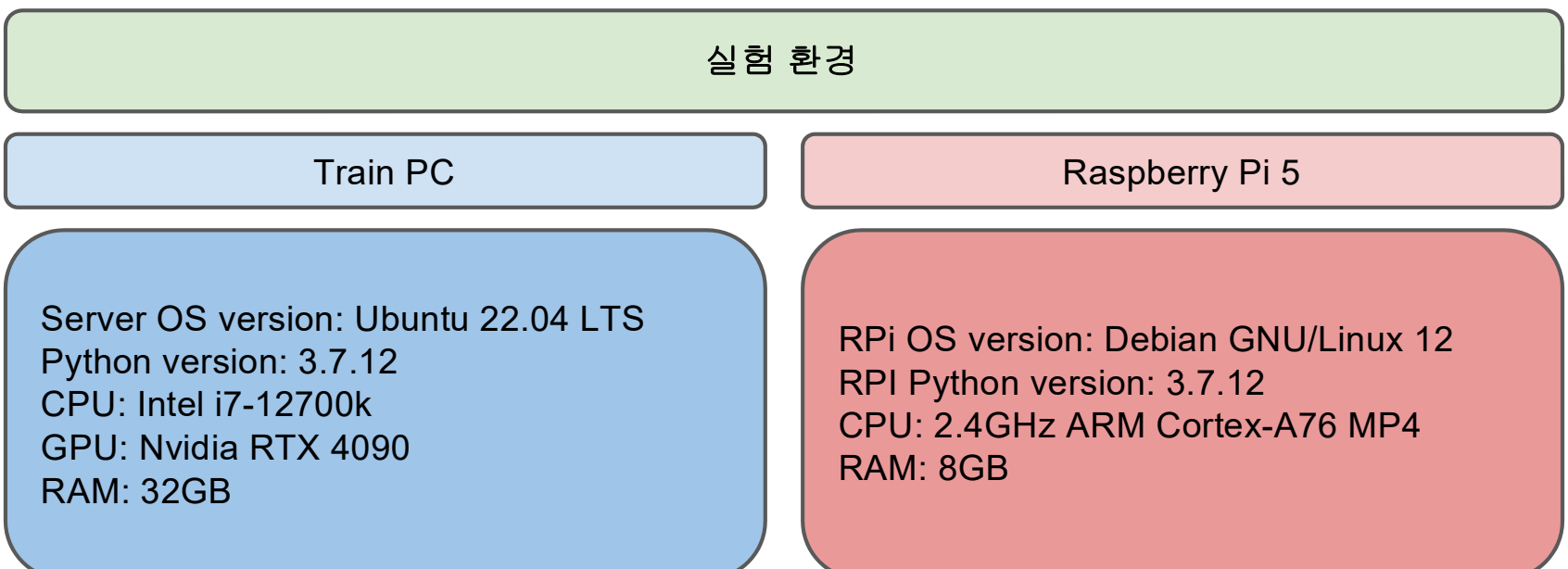
활성화 함수 변경

레이어 크기 조절

새 모듈 설계 교체

기존 모듈 도입

아키텍처 수정 실험 - 환경


실험 환경
Train PC
Raspberry Pi 5
Server OS version: Ubuntu 22.04 LTS
Python version: 3.7.12
CPU: Intel i7-12700k
GPU: Nvidia RTX 4090
RAM: 32GB
RPi OS version: Debian GNU/Linux 12
RPI Python version: 3.7.12
CPU: 2.4GHz ARM Cortex-A76 MP4
RAM: 8GB

아키텍처 수정 실험 - 환경

실험 환경

MODEL NAME	SAMPLE TABLE
mAP50 score:	
mAP50-95 score:	
RPI avg inference time:	
RPI max inference time:	
RPI min inference time:	
RPI avg CPU temp:	
PRAMS:	
GFLOPS:	
LAYERS:	

RPi 5 환경에서
실험 측정 결과의 신뢰성을 높이기 위해
변인을 찾고 통제하는 것에 집중

측정 시, 마우스를 움직이게 되면
GUI 자원의 사용으로 추론속도 차이 발생
ms 단위의 측정이므로, 결과값에 큰 영향

CPU 온도에 따른 Throttling 통제
입력 전압에 따른 다운 클럭 통제
기록용 RPi는 제공 충전기 사용 X

아키텍처 수정 실험 - 데이터셋

데이터셋

COCO 2017	PASCAL VOC	KITTI
Train image: 118287 Validate image: 5000 Test image: -	Train image: 13690 Validate image: 3422 Test image: -	Train image: 7481 Validate image: - Test image: 7518

중간 발표 이후 피드백을 반영하여 Benchmark dataset 추가 도입
[COCO 2017], [PASCAL VOC] 사용하여 실험

학습 시간 [COCO 2017] 약 5일 ~ 14일, PASCAL VOC 약 2일, 실험 진행을 위해 도입

아키텍처 수정 실험

활성화 함수 변경

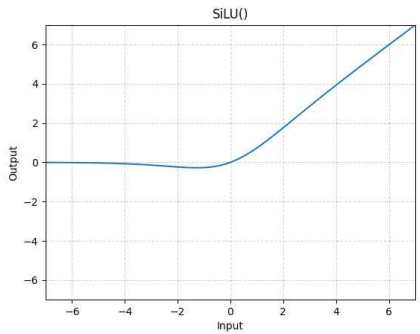
레이어 크기 조절

새 모듈 설계 교체

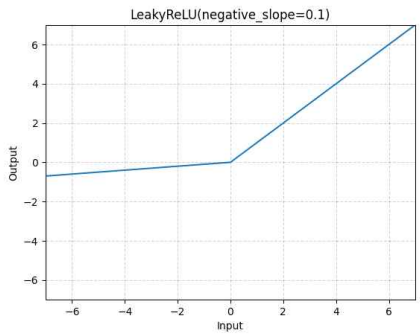
기존 모듈 도입

활성화 함수

$$SiLU(x) = x \times Sigmoid(x)$$



$$LReLU(x) = \begin{cases} x, & x \geq 0 \\ 0.01 \times x, & x < 0 \end{cases}$$



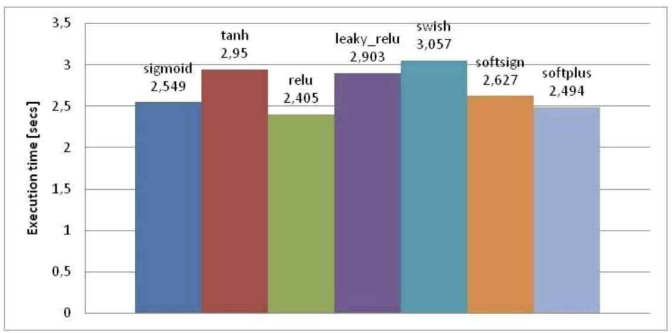


Fig. 14. Average time to classify 10 000 images with each AF

Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks(2020)

Leaky ReLU의 연산이 SiLU보다 상대적으로 간단
추론 속도가 빠를 것으로 예상

논문에서도 swish(SiLU)보다 Leaky ReLU의
추론 속도가 빠른 것을 확인할 수 있음

활성화 함수 실험

MODEL NAME	test12_3_PASCAL_500-converted
mAP50 score:	0.601
mAP50-95 score:	0.408
RPI avg inference time:	647.34ms
RPI max inference time:	689.22
RPI min inference time:	632.31
RPI avg CPU temp:	67.2C
PRAMS:	2766924
GFLOPS:	12.2
LAYERS:	194

[PASCAL] test 12_3
(Rep → Shuffle), Layer/2

MODEL NAME	test12_4_PASCAL_500-converted
mAP50 score:	0.599
mAP50-95 score:	0.405
RPI avg inference time:	618.1ms
RPI max inference time:	653.2
RPI min inference time:	608.4
RPI avg CPU temp:	67.26C
PRAMS:	2766924
GFLOPS:	12.2
LAYERS:	194

[PASCAL] test 12_4
(Rep → Shuffle), Layer/2, **LReLU**

추론 속도 향상 효과 확인

아키텍처 수정 실험

레이어 크기 조절

파라미터 수 및 연산량 감소를 위해
레이어 크기 조절 실험을 진행

레이어 크기 감소에 따른 mAP 손실이 발생

RPi 5 에서 사용할 수 있을 수준이려면
추론 속도 향상이 필수적 판단

기존 YOLO series 및 다른 객체 탐지 딥러닝 모델 구조를 참고

레이어 크기 조절 실험

[COCO 2017] Test 10 Layer / 2	[COCO 2017] Test 2 Layer / max 128
전체 레이어의 크기를 **비율에 맞춰 절반**으로 감소	전체 레이어에 대해서 **최대 레이어의 크기를 128**로 제한
기존 **모델의 구조를 유지**하기 위함	**과도한 연산 발생**을 막기 위함

*레이어 크기 조절: 모듈에서 INPUT 및 OUTPUT의 채널 개수를 조절

레이어 크기 조절 실험

MODEL NAME	YOLOv9-c-converted (Paper DL)
mAP50 score:	0.699
mAP50-95 score:	0.53
RPI avg inference time:	2892.96ms
RPI max inference time:	2980.88
RPI min inference time:	2861.62
RPI avg CPU temp:	72.58C
PRAMS:	25288768
GFLOPS:	102.1
LAYERS:	387

MODEL NAME	test2-converted_COCO_300
mAP50 score:	0.578
mAP50-95 score:	0.423
RPI avg inference time:	1330.86ms
RPI max inference time:	1447.05ms
RPI min inference time:	1317.9ms
RPI avg CPU temp:	69.69C
PRAMS:	3012928
GFLOPS:	27.4
LAYERS:	387

MODEL NAME	test10-converted_COCO_300
mAP50 score:	0.607
mAP50-95 score:	0.448
RPI avg inference time:	1113.22ms
RPI max inference time:	1187.47ms
RPI min inference time:	1096.41ms
RPI avg CPU temp:	70.42C
PRAMS:	6552000
GFLOPS:	26.6
LAYERS:	387

[COCO 2017] YOLOv9-c
Reference model

[COCO 2017] Test 2
Layer / max 128

[COCO 2017] Test 10
Layer / 2 , Leaky ReLU

최종 파라미터는 Test 2가 더 적지만 추론 속도는 Test 10이 더 빠름

비율로 줄이는 것이 더 효과적

아키텍처 수정 실험

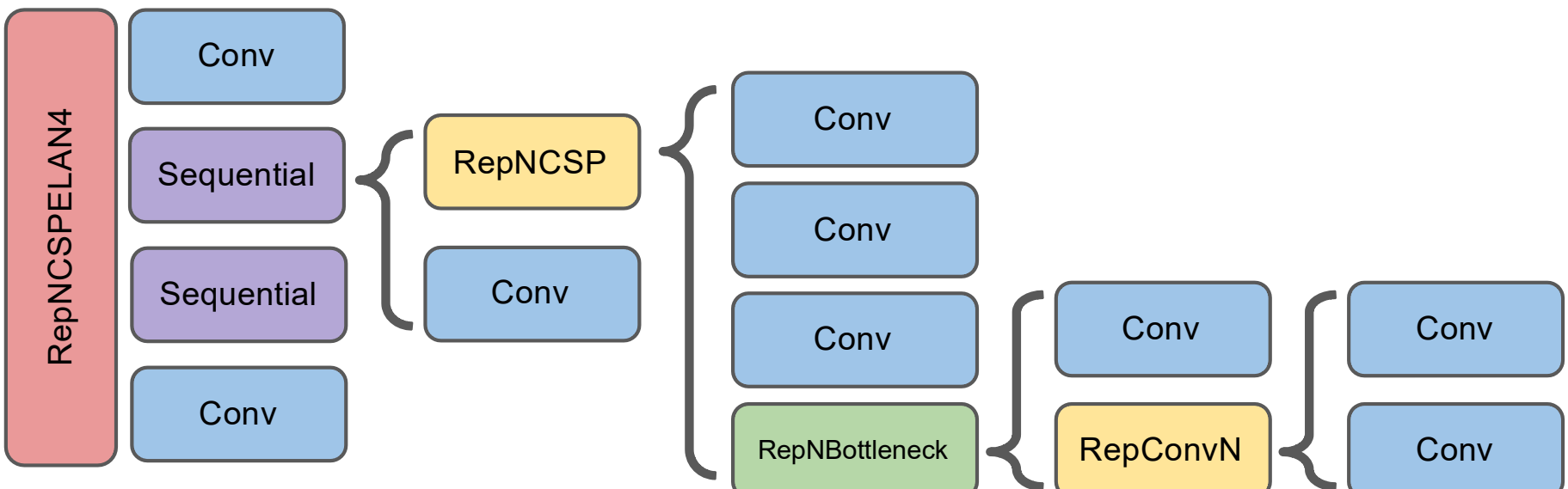
활성화 함수 변경

레이어 크기 조절

새 모듈 설계 교체

기존 모듈 도입

기존 모듈 구조 - RepNCSPELAN4



정보 손실을 최소화 하기 위한 블록 (**GELAN**)
16번의 Conv 사용으로 연산 시간 증가 문제

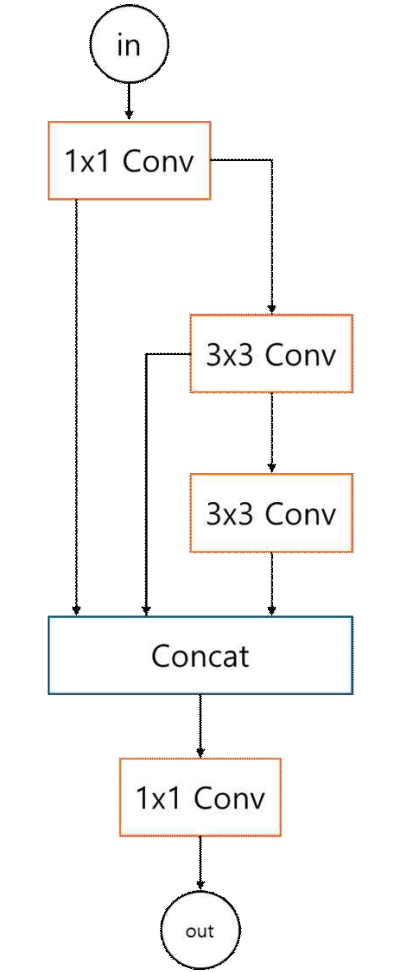
새 모듈 설계 - MGHELAN

```python
class MGHELAN(nn.Module):
    def __init__(self, c1, c2, c3, c4):
        super().__init__()
        self.cv1 = Conv(c1, c3, 1, 1)
        self.cv2 = Conv(c3//2, c4, 3, 1)
        self.cv3 = Conv(c4, c4, 3, 1)
        self.cv4 = Conv(c3 + 2 * c4, c2, 1, 1)

    def forward(self, x):
        y = list(self.cv1(x).chunk(2, 1))
        y.append(self.cv2(y[-1]))
        y.append(self.cv3(y[-1]))
        return self.cv4(torch.cat(y, 1))
```

RepNCSPELAN4의 Sequential 구조를 **Conv** 연산으로 변경

ELAN 구조를 참고하여 구성, 기존 모듈의 **설계 구조** 이식



새 모듈 설계 실험

	입력/ 출력	RepNCSPELAN4	MGHELAN	속도 증가 (rep -> MGHELAN)	향상 속도 (rep -> MGHELAN)
#2	1, 128, 160, 160	565.9418075	305.4590294	1.852758481	260.4827781
	1, 256, 160, 160				
#4	1, 256, 80, 80	357.2928271	205.0253923	1.742675983	152.2674348
	1, 512, 80, 80				
#6	1, 512, 40, 40	217.6081795	125.8302494	1.729378909	91.77793007
	1, 512, 40, 40				
#8	1, 512, 20, 20	54.59876157	31.40201079	1.73870272	23.19675079
	1, 512, 20, 20				
#12	1, 1024, 40, 40	228.3354055	135.9103981	1.680043681	92.42500744
	1, 512, 40, 40				
#15	1, 1024, 80, 80	362.2990732	207.0867661	1.749503747	155.2123071
	1, 256, 80, 80				
#18	1, 768, 40, 40	224.2779644	130.9398237	1.712832338	93.3381407
	1, 512, 40, 40				
#21	1, 1024, 20, 20	57.47755535	35.21750178	1.632073612	22.26005357
	1, 512, 20, 20				

파라미터 분석 결과, 약 1.6 ~ 1.8배의 **추론속도 향상 결과**를 확인

새 모듈 설계 실험

MODEL NAME	YOLOv9-c-converted (Paper DL)
mAP50 score:	0.699
mAP50-95 score:	0.53
RPI avg inference time:	2892.96ms
RPI max inference time:	2980.88
RPI min inference time:	2861.62
RPI avg CPU temp:	72.58C
PRAMS:	25288768
GFLOPS:	102.1
LAYERS:	387

[COCO 2017] YOLOv9-c
Reference model

MODEL NAME	test1- converted_COCO_500
mAP50 score:	0.669
mAP50-95 score:	0.5
RPI avg inference time:	2135.83ms
RPI max inference time:	2211.61ms
RPI min inference time:	2112.17ms
RPI avg CPU temp:	71.58C
PRAMS:	20540096
GFLOPS:	84.4
LAYERS:	162

[COCO 2017] YOLOv9-c
(seq → MGHELAN)

mAP가 소폭 하락하였지만, **추론 속도의 향상 결과**를 확인

아키텍처 수정 실험

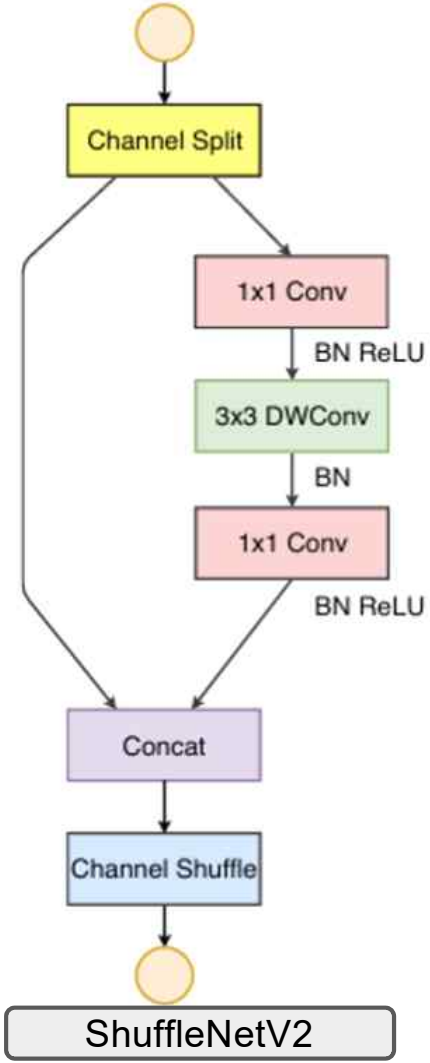
기존 모듈 도입

모듈 특징

Depth-wise Separable Convolution를 사용하여
연산 속도를 높임

Conv 연산에서 input channel 수와
output channel 수를 같게 만들어 연산 속도를 높임

Channel Shuffle을 사용해
채널간의 정보 교환을 가능하게 함



기존 모듈 도입

	입력/ 출력	RepNCSPELAN4	ShuffleNetV2	속도 증가 (rep -> shuff)	향상 속도 (rep -> shuff)
#2	1, 128, 160, 160	565.9418075	150.076678	3.77101769	415.8651295
	1, 256, 160, 160				
#4	1, 256, 80, 80	357.2928271	83.9460257	4.256220877	273.3468014
	1, 512, 80, 80				
#6	1, 512, 40, 40	217.6081795	25.42094112	8.560193679	192.1872384
	1, 512, 40, 40				
#8	1, 512, 20, 20	54.59876157	7.649391743	7.137660536	46.94936983
	1, 512, 20, 20				
#12	1, 1024, 40, 40	228.3354055	31.94938867	7.146784806	196.3860168
	1, 512, 40, 40				
#15	1, 1024, 80, 80	362.2990732	61.81595064	5.860931838	300.4831225
	1, 256, 80, 80				
#18	1, 768, 40, 40	224.2779644	27.95418327	8.023055518	196.3237811
	1, 512, 40, 40				
#21	1, 1024, 20, 20	57.47755535	8.961811457	6.413609082	48.51574389
	1, 512, 20, 20				

Input - Output 채널의 수가 같은 #6 레이어에서 효과적

기존 모듈 도입

MODEL NAME	test11_PASCAL_500-converted
mAP50 score:	0.698
mAP50-95 score:	0.527
RPI avg inference time:	838.48ms
RPI max inference time:	894.64
RPI min inference time:	830.62
RPI avg CPU temp:	68.88C
PRAMS:	5339212
GFLOPS:	22.1
LAYERS:	162

MODEL NAME	test12_4_PASCAL_500-converted
mAP50 score:	0.599
mAP50-95 score:	0.405
RPI avg inference time:	618.1ms
RPI max inference time:	653.2
RPI min inference time:	608.4
RPI avg CPU temp:	67.26C
PRAMS:	2766924
GFLOPS:	12.2
LAYERS:	194

[PASCAL VOC] test11
(Rep → **MGHELAN**), Layer/2, LReLU

[PASCAL VOC] test12_4
(Rep → **Shuffle**), Layer/2, LReLU

속도는 빨라졌지만, 정확도 손실이 큼

최종 모듈 설계

		RepNCSPELAN4	MGHELAN	ShuffleNetV2
#2	1, 128, 160, 160	565.9418075	305.4590294	150.076678
	1, 256, 160, 160			
#4	1, 256, 80, 80	357.2928271	205.0253923	83.9460257
	1, 512, 80, 80			
#6	1, 512, 40, 40	217.6081795	125.8302494	25.42094112
	1, 512, 40, 40			
#8	1, 512, 20, 20	54.59876157	31.40201079	7.649391743
	1, 512, 20, 20			
#12	1, 1024, 40, 40	228.3354055	135.9103981	31.94938867
	1, 512, 40, 40			
#15	1, 1024, 80, 80	362.2990732	207.0867661	61.81595064
	1, 256, 80, 80			
#18	1, 768, 40, 40	224.2779644	130.9398237	27.95418327
	1, 512, 40, 40			
#21	1, 1024, 20, 20	57.47755535	35.21750178	8.961811457
	1, 512, 20, 20			

[ShuffleNetV2] 기존 모듈 대비 속도 향상 8배 레이어 #6 적용

[MGHEALN4] 병목 현상이 큰 레이어 #2, #4, #8, # 12 적용

[RepNCSPELAN4] Prediction head #15, # 18, #21 적용, mAP 향상

최종 모듈 설계 실험

MODEL NAME	c_PASCAL_500-converted
mAP50 score:	0.747
mAP50-95 score:	0.596
RPI avg inference time:	2915.35ms
RPI max inference time:	2955.98
RPI min inference time:	2891.54
RPI avg CPU temp:	71.72C
PRAMS:	25242508
GFLOPS:	101.9
LAYERS:	387

[PASCAL VOC] test11
(Rep → **MGHELAN**), Layer/2, LReLU

MODEL NAME	test13_PASCAL_500-converted
mAP50 score:	0.686
mAP50-95 score:	0.535
RPI avg inference time:	884.9ms
RPI max inference time:	922.4
RPI min inference time:	859
RPI avg CPU temp:	62.67C
PRAMS:	5379404
GFLOPS:	22.2
LAYERS:	251

[PASCAL VOC] test13
(), Layer/2, LReLU, ONNX

속도는 빨라졌지만, 정확도 손실이 큼

연구 과정 및 실험 결과

후처리 적용

양자화

실수형 변수(floating-point type)를 정수형 변수(Qint8)로 변환해 추론속도 향상을 목표로 함

QNNPACK 라이브러리를 이용해 PTQ 수행. but, detection 실패

QAT 수행. pt모델의 scale값을 fine-tuning을 통해 보정해줌.
but, inference time 증가

가지치기

딥러닝 모델의 weight들 중 가중치가 낮은 weight의 연결을 제거하여 딥러닝 모델의 파라미터를 줄이는 방법

연산량이 많은 RepNCSPELAN4 블록 내부의
conv 레이어에서 l1 및 l2 norm을 활용한 filter 중요도 측정

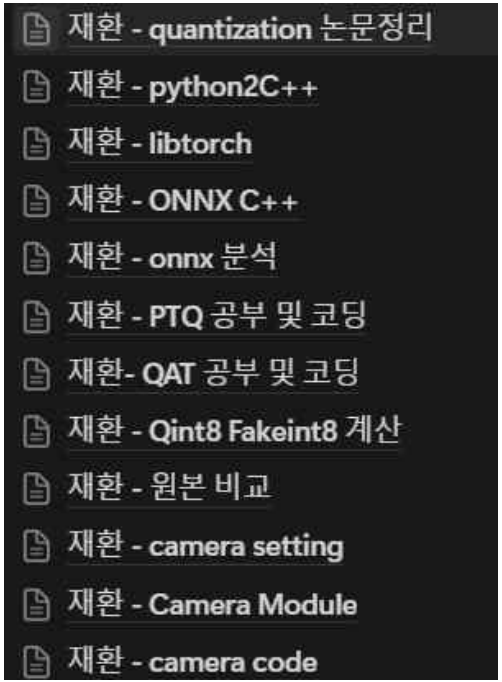
pt 파일 및 가중치를 0으로 전환하는 unstructured pruning과 실제로 제거하는 structured pruning

문서화

후처리 방법 적용과 관련하여

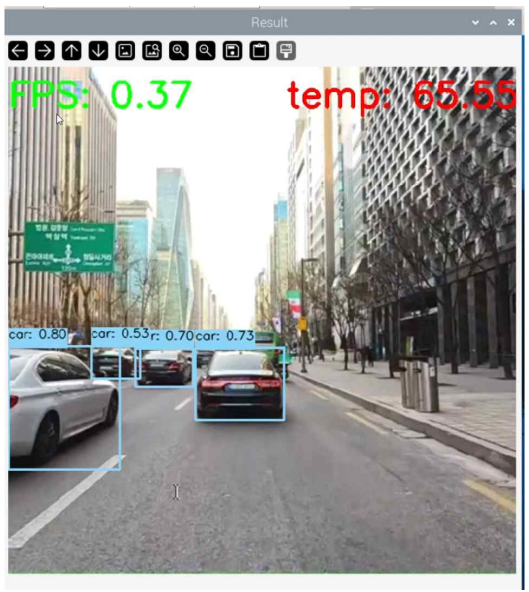
실제 적용에는 실패하였지만
모든 과정을 문서화 하여 정리

수많은 시행착오 과정 또한 의미가 있으며
앞으로 나아가기 위해 필수적이었던 과정

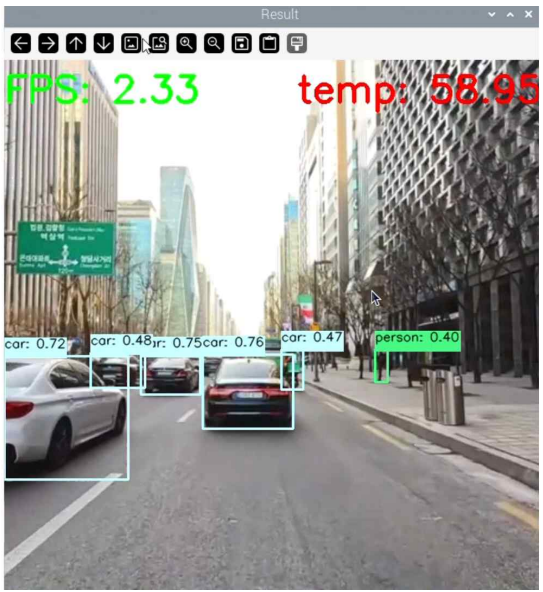


최종 모델 결과 시연

최종 결과 시연



yolov9-Compact 시연 영상



Test 13 시연 영상

추가 확인 사항

YOLOv9-Tiny가 2일 전 공개됨 ..

YOLOv9-Tiny가 발표 2일 전 공개됨

MODEL NAME	t_PASCAL_500 -converted
mAP50 score:	0.671
mAP50-95 score:	0.518
RPI avg inference time:	547.09ms
RPI max inference time:	580.9
RPI min inference time:	538.47
RPI avg CPU temp:	66.18C
PRAMS:	1882524
GFLOPS:	7.1
LAYERS:	489

MODEL NAME	test13_PASCAL_500-converted onnx
mAP50 score:	0.686
mAP50-95 score:	0.535
RPI avg inference time:	403.87ms
RPI max inference time:	405.25
RPI min inference time:	402.48
RPI avg CPU temp:	68.59C
PRAMS:	5379404
GFLOPS:	22.2
LAYERS:	251

YOLOv9-Tiny가 발표 2일 전 공개됨

```python
class ELAN1(nn.Module):

    def __init__(self, c1, c2, c3, c4):  # ch_in, ch_out, number,
        super().__init__()
        self.c = c3//2
        self.cv1 = Conv(c1, c3, 1, 1)
        self.cv2 = Conv(c3//2, c4, 3, 1)
        self.cv3 = Conv(c4, c4, 3, 1)
        self.cv4 = Conv(c3+(2*c4), c2, 1, 1)

    def forward(self, x):
        y = list(self.cv1(x).chunk(2, 1))
        y.extend(m(y[-1]) for m in [self.cv2, self.cv3])
        return self.cv4(torch.cat(y, 1))
```

```python
class MGHELAN(nn.Module):
    def __init__(self, c1, c2, c3, c4):
        super().__init__()
        self.cv1 = Conv(c1, c3, 1, 1)
        self.cv2 = Conv(c3//2, c4, 3, 1)
        self.cv3 = Conv(c4, c4, 3, 1)
        self.cv4 = Conv(c3 + 2 * c4, c2, 1, 1)

    def forward(self, x):
        y = list(self.cv1(x).chunk(2, 1))
        y.append(self.cv2(y[-1]))
        y.append(self.cv3(y[-1]))
        return self.cv4(torch.cat(y, 1))
```

기존 계획서와의 비교

기존 계획서의 목표

딥러닝 모델에
최적화 및 경량화 알고리즘을 적용하여 성능 향상

여러 경량화 방법을 적용하여
모델 학습 및 추론 성능 향상

저사양 임베디드 보드에서
모델을 구동시켜 성능을 비교 및 분석

RPi 5에서의 구동 및 분석 실험 진행
실시간 추론 성능 확인

구현 내용을 바탕으로 논문 작성

실험 과정 정리 및 문서화
기반으로 실험 고도화 계획

프로젝트 추진 기록

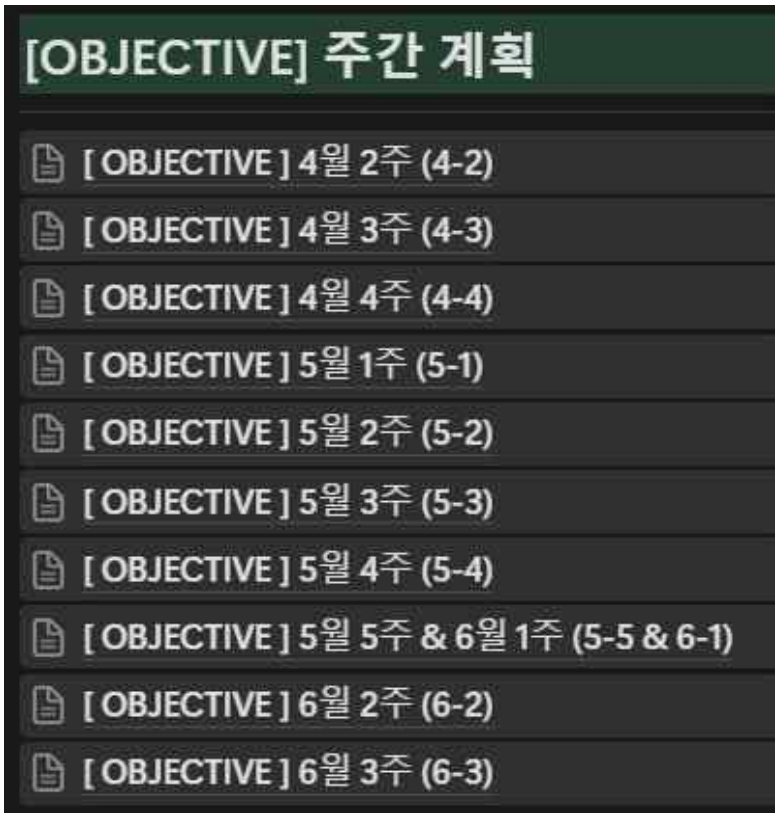
프로젝트 진행

월간 목표, 주간 목표 수립

일일 프로젝트 진행 전/후 브리핑 진행

성과 및 문제상황 공유

프로젝트 진행도 고려, 역할 배분



팀원별 역할분담

아키텍처 팀

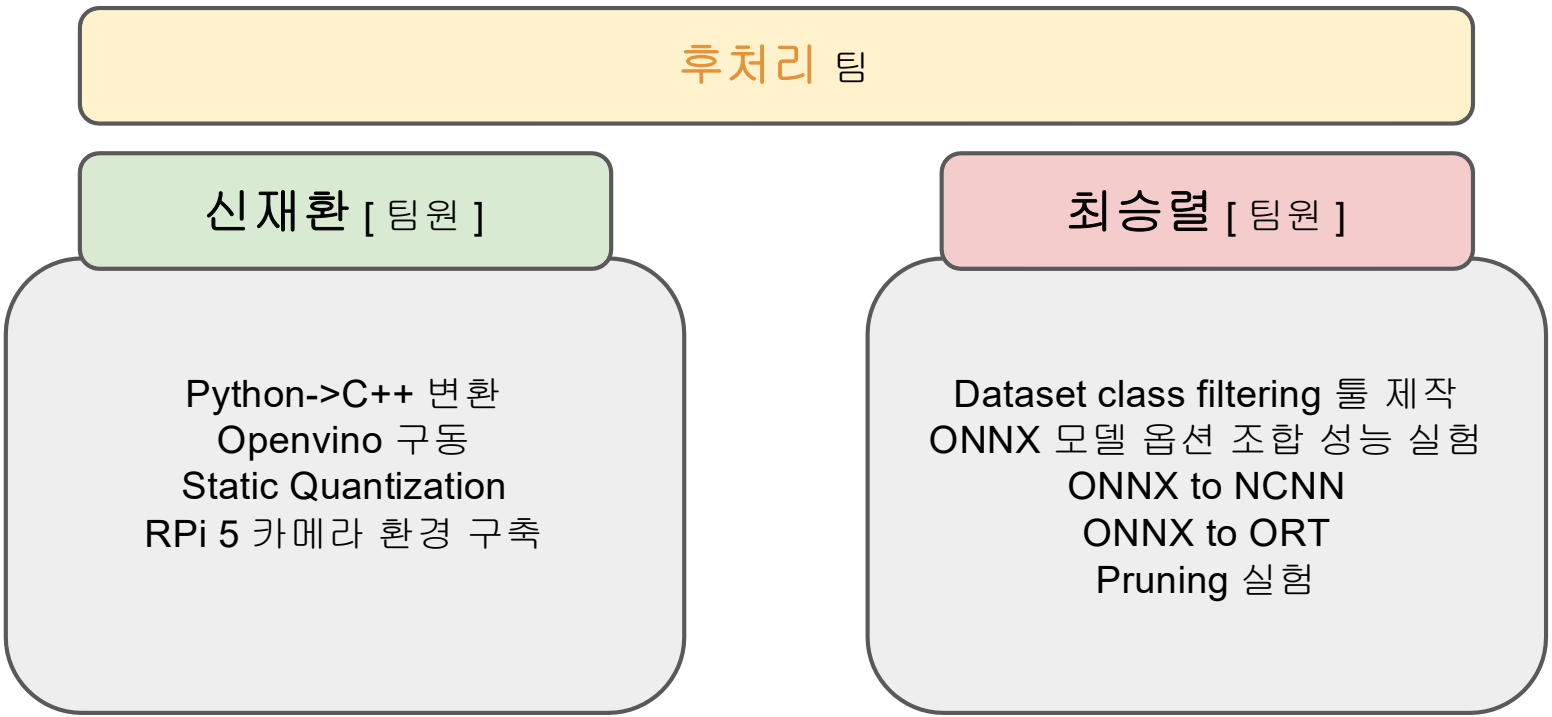
서태원 [팀장]

YOLOv9 논문 분석 및 실험 구현
YOLOv9 코드 및 연산구조 분석
연산량 및 추론속도 분석 실험
CPU 온도 및 추론속도 영향 실험
실험 환경 통제 계획 수립
MGHELAN 설계 및 분석
최종 모델 설계

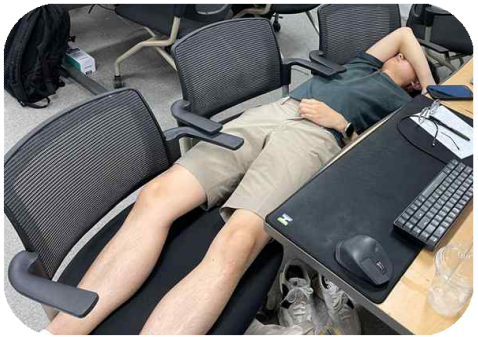
류은환 [팀원]

YOLOv9 논문 실험 구현
YOLOv9 논문 분석
YOLOv9 코드 분석
파라미터 분석
경량화 모델 설계 및 실험
ShuffleNetV2 분석 및 도입
활성화 함수 분석 및 변경

팀원별 역할분담


후처리 팀
신재환 [팀원]
Python->C++ 변환
Openvino 구동
Static Quantization
RPi 5 카메라 환경 구축
최승렬 [팀원]
Dataset class filtering 툴 제작
ONNX 모델 옵션 조합 성능 실험
ONNX to NCNN
ONNX to ORT
Pruning 실험

활동 시간


1월	2월	3월	4월	5월	6월
13시간	83시간	98시간	89시간	159시간	83시간

아이디어 회의 - 통과
[136시간]

통과 - 중간 발표
[212시간]

중간 발표 - 최종 발표
[212시간]

총 투입 시간 [**525** 시간]

연구 과정 리뷰

Thank you

무[遡]계획

인천대학교 INCHEON NATIONAL UNIVERSITY | 임베디드시스템공학과