주간 보고서-13주차 (2024.05.27~ 2024.06.02)

팀명 : 무계획

회의 참석자 : 서태원, 신재환, 최승렬, 류은환

회의 일자 및 회의 장소 :

2024.05.27(월)

- 서태원
  - 회의 시간 : 16:00 ~ 23:00
  - 회의 장소 : 7호관 302호
- 신재환
  - 회의 시간 : 14:50 ~ 23:00
  - 회의 장소 : 7호관 302호
- 최승렬
  - 회의 시간 : 14:00 ~ 23:00
  - 회의 장소 : 7호관 302호
- 류은환
  - 회의 시간 : 14:00 ~ 23:00
  - 회의 장소 : 7호관 302호

2024.05.28(화)

- 서태원
  - 회의 시간 : 14:30 ~ 16:30, 18:30 ~ 23:30
  - 회의 장소 : 7호관 302호
- 신재환
  - 회의 시간 : 18:30 ~ 23:10
  - 회의 장소 : 7호관 302호
- 최승렬
  - 회의 시간 : 18:30 ~ 23:30
  - 회의 장소 : 7호관 302호
- 류은환
  - 회의 시간 : 14:00 ~ 16:00, 18:30 ~ 23:30
  - 회의 장소 : 7호관 302호

2024.05.29(수)

- 신재환
  - 회의 시간 : 16:45 ~ 21:45
  - 회의 장소 : 7호관 302호
- 최승렬
  - 회의 시간 : 16:45 ~ 21:45
  - 회의 장소 : 7호관 302호

2024.05.30(목)

- 서태원
  - 회의 시간 : 13:00 ~ 23:30
  - 회의 장소 : 7호관 302호
- 신재환
  - 회의 시간 : 15:30 ~ 23:55
  - 회의 장소 : 7호관 302호
- 최승렬

    - 회의 시간 : 19:40 ~ 23:50
    - 회의 장소 : 7호관 302호
- 류은환
    - 회의 시간 : 15:00 ~ 23:30
    - 회의 장소 : 7호관 302호

2024.06.01(토)

- 서태원
    - 회의 시간 : 19:20 ~ 20:40
    - 회의 장소 : 디스코드 음성채널
- 신재환
    - 회의 시간 : 13:40 ~ 17:40, 19:00 ~ 22:20
    - 회의 장소 : 7호관 302호, 디스코드 음성채널
- 최승렬
    - 회의 시간 : 18:20 ~ 22:20
    - 회의 장소 : 7호관 302호, 디스코드 음성채널
- 류은환
    - 회의 시간 : 19:20 ~ 22:20
    - 회의 장소 : 디스코드 음성채널

총 활동 시간 :

- 서태원 : 25시간 50분
- 신재환 : 33시간 55분
- 최승렬 : 27시간 10분
- 류은환 : 27시간 30분

진행 사항 :

현재 역할

서태원

[업무]

류은환

[업무]

레이어 연산시간 분석 도구 개발

레이어 별 연산시간 분석

연구 방향성 수립

주간 역할 배분 및 회의 진행

레이어 파라미터 분석

논문 탐독 및 근거 자료 준비

신재환

[업무]

pt, onnx 실시간 추론 코딩

캡스톤 디자인 시연 계획

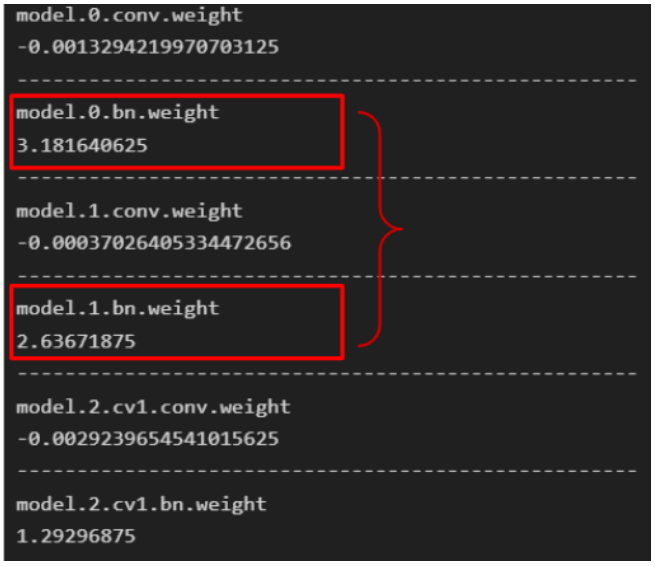
최승렬

[업무]

onnx 모델 최적화 실험

캡스톤 디자인 시연 계획

레이어 파라미터 분석



- 레이어마다 모든 weights 값을 추출하여 평균 활성화 정도를 구함. 이에 conv.weight는 0에 가깝게, bn.weight는 1에 가깝게 나타남.
- 다른 bn.weight들에 비해 높은 값이고, 이는 모델이 초기에 입력 데이터를 정규화 하는 과정에서 발생하는 값이고, 이 외에는 이상치를 발견 하지 못함.

레이어 연산시간 분석

- 레이어의 연산시간과 파라미터 수, 연산량 분석 코드를 작성하고 YOLOv9-c를 구성하는 모든 레이어에 대해 분석을 수행함
- 각 계층에 입력되는 크기에 맞춰 데이터를 생성하고, RPi5 에서 레이어 연산 수행 시간을 측정함

- [50번 수행 결과의 평균 값]의 10번 평균 연산 시간을 측정하고 분석함.

```
# conv down
[-1, 1, Conv, [64, 3, 2]],  # 0-P1/2

# conv down
[-1, 1, Conv, [128, 3, 2]],  # 1-P2/4

# elan-1 block
[-1, 1, RepNCSPELAN4, [256, 128, 64, 1]],  # 2

# avg-conv down
[-1, 1, ADown, [256]],  # 3-P3/8

# elan-2 block
[-1, 1, RepNCSPELAN4, [512, 256, 128, 1]],  # 4

# avg-conv down
[-1, 1, ADown, [512]],  # 5-P4/16

# elan-2 block
[-1, 1, RepNCSPELAN4, [512, 512, 256, 1]],  # 6

# avg-conv down
[-1, 1, ADown, [512]],  # 7-P5/32

# elan-2 block
[-1, 1, RepNCSPELAN4, [512, 512, 256, 1]],  # 8
]
```

```
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: 370.58416326666584 ms
Average Execution Time: 382.08743619999 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 $
```

```
import torch
import torch.nn as nn
import time
from models.common import *  # models.common 모듈 임포트
from thop import profile

class BenchmarkModule:

    def __init__(self, input_size, module_class, *module_args):
        # 입력 랜덤 데이터 생성 부분
        self.device = torch.device('cpu')
        self.input_size = input_size
        self.input_data = torch.zeros(*input_size).to(self.device)

        # 모듈 설정 및 파라미터 전달
        self.module = module_class(*module_args).to(self.device).eval()

    def warmup(self, iterations=10):
        # 웜업 코드 작성
        for _ in range(iterations):
            with torch.no_grad():
                _ = self.module(self.input_data)
```

```
    def analyze_performance(self, num_iterations=30):
        # 웜업
        self.warmup()

        # 성능 측정
        execution_times = []
        for _ in range(num_iterations):
            start_time = time.perf_counter()
            with torch.no_grad():
                output = self.module(self.input_data)
            execution_time = time.perf_counter() - start_time
            execution_times.append(execution_time)

        avg_execution_time = sum(execution_times) / len(execution_times)

        flops, params = profile(self.module, inputs=(self.input_data,), v

        return {
            'execution_time': avg_execution_time * 1000,  # ms 단위로 변환
            'num_params': params,
            'flops': flops,
            'shape': output.shape
        }


if __name__ == "__main__":
    # 사용자 입력 받기
    user_input = input("Enter the input size, channels, layer name, and l
    # 입력된 문자열을 콤마로 분리하여 리스트로 변환
    inputs = user_input.split(',')
    # 입력 크기와 채널 수, 레이어 이름을 추출
    input_height = int(inputs[0].strip())
    input_channels = int(inputs[1].strip())
    module_name = inputs[2].strip()

    # 모듈 이름에 해당하는 클래스를 동적으로 가져오기
    module_class = globals()[module_name]
    module_args = [int(arg.strip()) if arg.strip().isdigit() else arg.str
    input_size = (1, input_channels, input_height, input_height)
    analyzer = BenchmarkModule(input_size, module_class, *module_args)

    avg_exes = []
    for _ in range(10):
        results = analyzer.analyze_performance()
        print(f"Average Execution Time: {results['execution_time']} ms")
```

```
        avg_exes.append(results['execution_time'])

    avg_exe = sum(avg_exes) / len(avg_exes)
    print(f"User Input: {user_input}")
    print(f"BenchMark Module Name: {module_name}")
    print(f"Total Average Execution Time: {avg_exe} ms")
    print(f"Number of Parameters: {results['num_params']}")
    print(f"FLOPs: {results['flops']}")
    print(f"Output Shape: {results['shape']}")
```

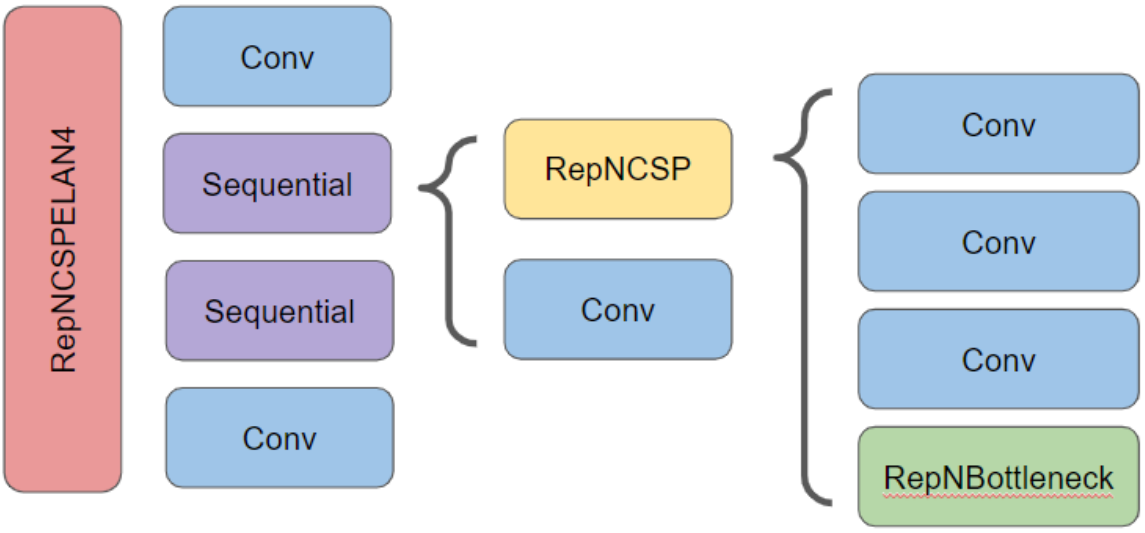
BACKBONE

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

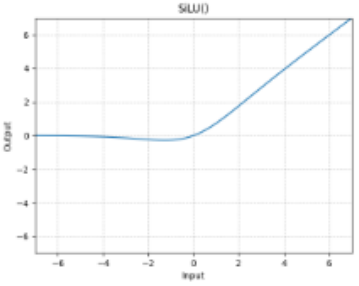
YOLOv9-c 모델 모든 레이어의 연산 시간 측정하였고, 연산 시간 측정 결과 연산 시간이 오래 걸리는 블록을 확인하였고, RepNCSPELAN4이 이에 해당함을 확인



블록 내부 구조를 확인해본 결과 정보 손실을 최소화 하기 위한 블록(GELAN)으로 10번의 Conv 사용으로 연산 시간 증가 문제가 있음. 그렇기에 라즈베리파이 5에서 연산시간이 오래 걸리는 블록을 연산 시간이 짧고, 라즈베리파이에 적합한 블록으로 교체하고자 함. 이 블록은 기존 논문에서 사용되는 블록 혹은 기존 딥러닝 모델에 사용되는 블록 등을 조사하고 이를 라즈베리파이 5에 적합하게 설계를 변경하여 모델 블록을 대체하는 쪽으로 설계 방향을 세움.

Activation Function

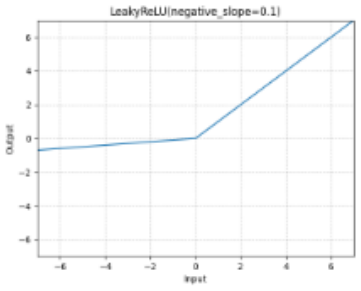
$$SiLU(x) = x \times Sigmoid(x)$$



swish

- Searching for Activation Functions(2017)
- Sigmoid에 입력값을 한번 더 곱해준 함수

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



Leaky ReLU

- Rectifier Nonlinearities Improve Neural Network Acoustic Models(2013)
- ReLU 함수의 Dead ReLU 문제를 해결하기 위해 일정한 기울기를 추가한 함수

Leaky ReLU의 연산이 SiLU보다 상대적으로 간단하므로 추론 속도가 빠를 것으로 예상함

yolov7-Tiny

```
backbone:
  # [from, number, module, args] c2, k=1, s=1, p=None, g=1, act=True
  [[-1, 1, Conv, [32, 3, 2, None, 1, nn.LeakyReLU(0.1)]],  # 0-P1/2

   [-1, 1, Conv, [64, 3, 2, None, 1, nn.LeakyReLU(0.1)]],  # 1-P2/4

   [-1, 1, Conv, [32, 1, 1, None, 1, nn.LeakyReLU(0.1)]],
   [-2, 1, Conv, [32, 1, 1, None, 1, nn.LeakyReLU(0.1)]],
   [-1, 1, Conv, [32, 3, 1, None, 1, nn.LeakyReLU(0.1)]],
   [-1, 1, Conv, [32, 3, 1, None, 1, nn.LeakyReLU(0.1)]],
   [[-1, -2, -3, -4], 1, Concat, [1]],
   [-1, 1, Conv, [64, 1, 1, None, 1, nn.LeakyReLU(0.1)]],  # 7
```

yolov7-default

```
class Conv(nn.Module):
    # Standard convolution
    def __init__(self, c1, c2, k=1, s=1, p=None, g=1, act=True):  # ch_in, ch_out, kernel, stride, padding, groups
        super(Conv, self).__init__()
        self.conv = nn.Conv2d(c1, c2, k, s, autopad(k, p), groups=g, bias=False)
        self.bn = nn.BatchNorm2d(c2)
        self.act = nn.SiLU() if act is True else (act if isinstance(act, nn.Module) else nn.Identity())

    def forward(self, x):
        return self.act(self.bn(self.conv(x)))

    def fuseforward(self, x):
        return self.act(self.conv(x))
```

YOLOv7-Tiny에서도 SiLU대신 Leaky ReLU를 사용하는 것을 확인할 수 있음

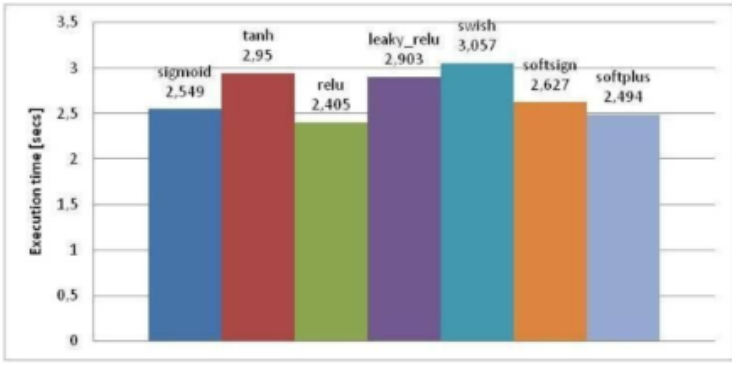


Fig. 14. Average time to classify 10 000 images with each AF

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

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

ONNX Optimization

또한 여러 방면으로 onnx 모델의 최적화를 진행함

MODEL NAME	**YOLOv9-c-converted_graph_optimization_3.8**
RPI avg inference time:	1768.5ms
RPI max inference time:	1829.05ms
RPI min inference time:	1693.53ms
RPI avg CPU temp:	73.96C

onnx 모델에서 그래프를 최적화한 모델. 그래프 최적화란 기본적으로 작은 그래프 단순화 및 노드제거부터 보다 복잡한 노드 융합 및 레이아웃 최적화에 이를 그래프 수준 변환을 의미함. 여기서는 작은 그래프 단순화, 중복 노드 및 중복 연산 제거함.

MODEL NAME	**YOLOv9-c-converted_spo4onnx_3.8**
RPI avg inference time:	1710.78ms
RPI max inference time:	1806.31ms
RPI min inference time:	1619.08ms
RPI avg CPU temp:	74.02C

onnx simplifier로 최적화 할 수 없는 일부 모델들을 더 최적화 하는 방법을 적용함. 즉 최적화 용량을 더욱 높이는 방법을 적용해서 실험함

MODEL NAME	**YOLOv9-c-converted_onnx_preprocessing_3.8**	**MODEL NAME**	**YOLOv9-c-converted_onnx_dynamicQuan**
RPI avg inference time:	1786.57ms	RPI avg inference time:	1674.83ms

RPI max inference time:	2264.29ms	RPI max inference time:	1811.65ms
RPI min inference time:	1714.93ms	RPI min inference time:	1645.87ms
RPI avg CPU temp:	74.12C	RPI avg CPU temp:	65.14C

양자화가 보다 좋은 성능을 내기 위한 전처리 단계의 모델, onnx 모델 최적화를 거친 모델이고, 실제로 이를 가지고 동적 양자화를 진행하여 실험함

MODEL NAME	**yolov9-test10-converted_onnx_3.7**
RPI avg inference time:	495.49ms
RPI max inference time:	565.37ms
RPI min inference time:	458.57ms
RPI avg CPU temp:	71.15C

보조분기를 제거(Converted)하고, LeakyReLU 적용(yolov9-c.yaml 파일에 parameters 부분에 activation: nn.LeakyReLU(0.1)를 추가) 및 비율에 맞추어 레이어 크기 감소(1/2)한 test10 모델에 onnx를 적용함. 기존 모델 대비 추론속도가 6배 정도 빨라진 것을 확인함.

Camera Setting

yolov9 pt 돌리는 방법

```
    def create_session(self, fp16=False, fuse=True) -> None:
        # opt_session = onnxruntime.SessionOptions()
        # opt_session.graph_optimization_level = onnxruntime.GraphOptimiz
        # providers = ['CPUExecutionProvider']
        # if self.device.casefold() != "cpu":
        #     providers.append("CUDAExecutionProvider")
        # session = onnxruntime.InferenceSession(self.model_path, provide
        # self.session = session
        # self.model_inputs = self.session.get_inputs()
        # self.input_names = [self.model_inputs[i].name for i in range(le
        # self.input_shape = self.model_inputs[0].shape
        # self.model_output = self.session.get_outputs()
        # self.output_names = [self.model_output[i].name for i in range(l
        # self.input_height, self.input_width = self.input_shape[2:]

        self.model = attempt_load(self.model_path, device=self.device, in

        # Ensure the model is on the correct device
        self.model.to(self.device)
```

```
        # Convert model to half precision if fp16 is enabled, otherwise u
        if fp16 and self.device == 'cuda':
            self.model.half()
        else:
            self.model.float()

        # Retrieve stride and class names from the model
        self.stride = max(int(self.model.stride.max()), 32)  # model stri
        self.names = self.model.module.names if hasattr(self.model, 'modu

        # Set input dimensions based on the model's stride
        self.input_height, self.input_width = [x // self.stride * self.st

        if self.class_mapping_path is not None:
            with open(self.class_mapping_path, 'r') as file:
                yaml_file = yaml.safe_load(file)
                self.classes = yaml_file['names']
                self.color_palette = np.random.uniform(0, 255, size=(len(
```

- 위 코드는 onnx코드를 pt파일로 바꾼 것.
- yolov9 DetectMultiBackend 함수에서 pt파일을 실행하려면 다음과 같이 모델을 맞춰줘야 함.

```
if pt:  # PyTorch
    model = attempt_load(weights if isinstance(weights, list) else w, dev
    stride = max(int(model.stride.max()), 32)  # model stride
    names = model.module.names if hasattr(model, 'module') else model.nam
    model.half() if fp16 else model.float()
    self.model = model  # explicitly assign for to(), cpu(), cuda(), half
```

attempt_load 함수 추가

```
def attempt_load(weights, device=None, inplace=True, fuse=True):
        # Loads an ensemble of models weights=[a,b,c] or a single model w
        from models.yolo import Detect, Model

        model = Ensemble()
        for w in weights if isinstance(weights, list) else [weights]:
            ckpt = torch.load(attempt_download(w), map_location='cpu')  #
            ckpt = (ckpt.get('ema') or ckpt['model']).to(device).float()

            # Model compatibility updates
            if not hasattr(ckpt, 'stride'):
                ckpt.stride = torch.tensor([32.])
```

```
            if hasattr(ckpt, 'names') and isinstance(ckpt.names, (list, t
                ckpt.names = dict(enumerate(ckpt.names))  # convert to di

            model.append(ckpt.fuse().eval() if fuse and hasattr(ckpt, 'fu

        # Module compatibility updates
        for m in model.modules():
            t = type(m)
            if t in (nn.Hardswish, nn.LeakyReLU, nn.ReLU, nn.ReLU6, nn.Si
                m.inplace = inplace  # torch 1.7.0 compatibility
                # if t is Detect and not isinstance(m.anchor_grid, list):
                #    delattr(m, 'anchor_grid')
                #    setattr(m, 'anchor_grid', [torch.zeros(1)] * m.nl)
            elif t is nn.Upsample and not hasattr(m, 'recompute_scale_fac
                m.recompute_scale_factor = None  # torch 1.11.0 compatibi

        # Return model
        if len(model) == 1:
            return model[-1]

        # Return detection ensemble
        print(f'Ensemble created with {weights}\n')
        for k in 'names', 'nc', 'yaml':
            setattr(model, k, getattr(model[0], k))
        model.stride = model[torch.argmax(torch.tensor([m.stride.max() fo
        assert all(model[0].nc == m.nc for m in model), f'Models have dif
        return model
```

- 위 코드를 참조해서 아래와 같이 코드를 변경함.

```
self.model = attempt_load(self.model_path, device=self.device, inplace=Tr
# Ensure the model is on the correct device
self.model.to(self.device)
# Convert model to half precision if fp16 is enabled, otherwise use full
if fp16 and self.device == 'cuda':
    self.model.half()
else:
    self.model.float()
# Retrieve stride and class names from the model
self.stride = max(int(self.model.stride.max()), 32)  # model stride
self.names = self.model.module.names if hasattr(self.model, 'module') els
# Set input dimensions based on the model's stride
self.input_height, self.input_width = [x // self.stride * self.stride for
```

process함수 변경

```
def preprocess(self, img: np.ndarray) -> torch.Tensor:
    # 이미지를 RGB로 변환
    image_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    # 모델 입력 크기에 맞게 이미지 크기 조정
    resized = cv2.resize(image_rgb, (self.input_width, self.input_height)
    # 이미지 픽셀 값을 [0, 1] 범위로 정규화
    input_image = resized / 255.0
    # 채널 차원을 앞으로 이동 (채널 우선 형식)
    input_image = np.transpose(input_image, (2, 0, 1))
    # numpy 배열을 PyTorch 텐서로 변환
    input_tensor = torch.tensor(input_image, dtype=torch.float32).unsquee
    if self.device == 'cuda':
        input_tensor = input_tensor.cuda()  # GPU 사용 시 텐서를 CUDA로 이동
    return input_tensor
```

- 전처리 하는 이유 : numpy.ndarray가 input 이미지가 전달됨. 이는 데이터 타입에 맞지 않기에 input image를 tensor로 변환함. conv2D가 tensor 형식을 입력받아야 함.
- 입력 이미지를 pytorch tensor로 변환하는 전처리를 해줌.

detect 함수 수정

```
def detect(self, img: np.ndarray) -> List:
    input_tensor = self.preprocess(img)
    with torch.no_grad():  # 그래디언트 계산 비활성화
        outputs = self.model(input_tensor)  # 모델 실행
    # 결과를 CPU로 이동하고 NumPy 배열로 변환
    outputs = outputs.cpu().numpy()
    return self.postprocess(outputs)
```

- 이제 tensor로 변환된 모델을 detect함. input을 numpy 배열로 잘못 넘겨주지 않도록 해야함.
- output은 numpy가 되야함. 즉, 텐서를 모델에 전달해야 하며 또한 모든 데이터 처리가 PyTorch 텐서로 이뤄져야함. numpy()는 model output에만 전달 해야함.

최종 demostration 코드

```
import os
import cv2
from pathlib import Path
from yolov9 import YOLOv9
import subprocess
import numpy as np
```

```
import threading
import queue
import argparse
import time
import datetime

def read_cpu_temperature():
    """Reads the CPU temperature of a Raspberry Pi from the system file."
    try:
        with open("/sys/class/thermal/thermal_zone0/temp", "r") as f:
            temp = f.read().strip()
        return float(temp) / 1000.0  # Convert to Celsius
    except FileNotFoundError:
        return "CPU temperature not available"


def start_camera_stream():
    return subprocess.Popen(['libcamera-vid', '-t', '0', '--width', '640'
                             stdout=subprocess.PIPE, bufsize=10**6)

def read_frame(process, width, height):
    frame_size = width * height * 3 // 2
    try:
        raw_frame = process.stdout.read(frame_size)
        if len(raw_frame) == frame_size:
            yuv = np.frombuffer(raw_frame, dtype=np.uint8).reshape((heigh
            return cv2.cvtColor(yuv, cv2.COLOR_YUV2BGR_I420)
    except Exception as e:
        print(f"Error reading frame: {e}")
    return None

def get_detector(args):
    assert os.path.isfile(args.weights), f"There's no weight file with na
    assert os.path.isfile(args.classes), f"There's no classes file with n
    detector = YOLOv9(model_path=args.weights,
                      class_mapping_path=args.classes,
                      original_size=(640, 640),
                      score_threshold=args.score_threshold,
                      conf_thresold=args.conf_threshold,  # ?섒젙??遺遺?
                      device=args.device)
    return detector

def capture_frames(process, frame_queue):
    while True:
        frame = read_frame(process, 640, 640)
        if frame is not None:
```

```
            if frame_queue.full():
                frame_queue.get()  # 踰꾪뜥媛 媛??李⑤㉣ 媛???ㅅ옒???꾨젅?꾩쓿
            frame_queue.put(frame)

def process_frames(detector, frame_queue):
    window_name = 'YOLOv9 Detection'
    cv2.namedWindow(window_name, cv2.WND_PROP_FULLSCREEN)
    cv2.setWindowProperty(window_name, cv2.WND_PROP_FULLSCREEN, cv2.WINDO
    while True:
        start = datetime.datetime.now()
        cpu_temp = read_cpu_temperature()
        if not frame_queue.empty():
            frame = frame_queue.get()
            detections = detector.detect(frame)
            detector.draw_detections(frame, detections=detections)

            end = datetime.datetime.now()
            total = (end - start).total_seconds()
            fps = 1 / total
            cv2.putText(frame, f'FPS: {fps:.2f}', (0, 50), cv2.FONT_HERSH
            cv2.putText(frame, f'temp: {cpu_temp:.2f}', (350, 50), cv2.FO

            cv2.imshow(window_name, frame)
            if cv2.waitKey(1) == ord('q'):
                break
        else:
            time.sleep(0.1)  # Reduce CPU usage when no frame is availabl

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Argument for YOLOv9 Inf
    parser.add_argument("--source", type=str, required=True, help="Path t
    parser.add_argument("--weights", type=str, required=True, help="Path
    parser.add_argument("--classes", type=str, required=True, help="Path
    parser.add_argument("--score-threshold", type=float, required=False,
    parser.add_argument("--conf-threshold", type=float, required=False, d
    parser.add_argument("--iou-threshold", type=float, required=False, de
    parser.add_argument("--image", action="store_true", required=False, h
    parser.add_argument("--video", action="store_true", required=False)
    parser.add_argument("--camera", action="store_true", required=False)
    parser.add_argument("--show", required=False, type=bool, default=True
    parser.add_argument("--device", type=str, required=False, help="Devic
    args = parser.parse_args()

    if args.camera:
        frame_queue = queue.Queue(maxsize=5)  # ???ш린瑜?議곗젙?섎㈃ 踰꾪뜥留
        process = start_camera_stream()
```

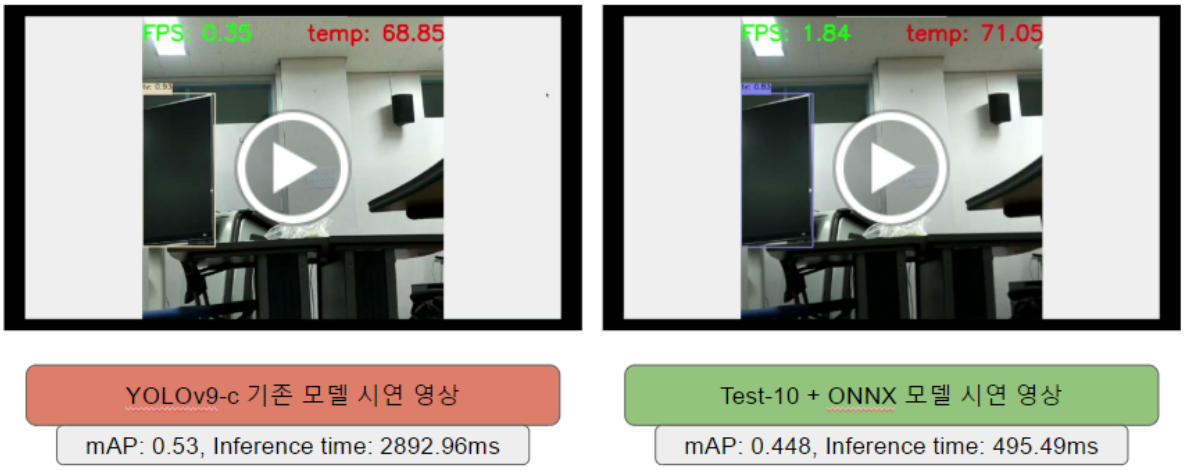
```
        detector = get_detector(args)

        capture_thread = threading.Thread(target=capture_frames, args=(pr
        process_thread = threading.Thread(target=process_frames, args=(de

        capture_thread.start()
        process_thread.start()

        capture_thread.join()
        process_thread.join()
    elif args.image:
        pass  # inference_on_image(args=args)
    elif args.video:
        pass  # inference_on_video(args=args)
    else:
        raise ValueError("You must define the source type (camera, video,
```

이를 바탕으로 라즈베리파이 Pi-camera를 사용하여 yolov9-c-converted 모델과 성능을 비교하는 영상을 촬영함.



비고

- 캡스톤 디자인 시연에서 라즈베라파이 5와 Pi-camera를 사용하여 실시간 영상 객체 탐지 성능 시연할 계획. 다중 물체에 대한 Object detection을 수행할 수 있는 사진을 준비하여 각 모델에 대하여 현장에서 실시간으로 추론 시연할 계획임.

추후 계획

- 모듈 대체 관련 자료 및 근거 조사

- 수정 모듈 준비 및 코드 수정
- 개선 모델 학습 진행
- 성능 비교 및 결과 분석
- 최종 모델 및 시연 준비