주간 보고서-12주차 (2024.05.20~ 2024.05.26)

팀명 : 무계획

회의 참석자 : 서태원, 신재환, 최승렬, 류은환

회의 일자 및 회의 장소 :

2024.05.20(월)

- 서태원
  - 회의 시간 : 16:30 ~ 00:00
  - 회의 장소 : 7호관 302호
- 신재환
  - 회의 시간 : 15:20 ~ 00:20
  - 회의 장소 : 7호관 302호
- 최승렬
  - 회의 시간 : 15:10 ~ 00:10
  - 회의 장소 : 7호관 302호
- 류은환
  - 회의 시간 : 15:00 ~ 23:30
  - 회의 장소 : 7호관 302호

2024.05.21(화)

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

2024.05.22(수)

- 서태원
  - 회의 시간 : 19:00 ~ 22:00
  - 회의 장소 : 7호관 302호
- 신재환
  - 회의 시간 : 12:40 ~ 23:45
  - 회의 장소 : 7호관 302호

2024.05.23(목)

- 서태원
  - 회의 시간 : 10:30 ~ 23:30
  - 회의 장소 : 7호관 302호
- 신재환
  - 회의 시간 : 10:55 ~ 00:05
  - 회의 장소 : 7호관 302호

- 최승렬
  - 회의 시간 : 14:30 ~ 23:45
  - 회의 장소 : 7호관 302호
- 류은환
  - 회의 시간 : 20:00 ~ 22:00
  - 회의 장소 : 7호관 302호

2024.05.24(금)

- 서태원
  - 회의 시간 : 18:00 ~ 23:30
  - 회의 장소 : 7호관 302호
- 신재환
  - 회의 시간 : 17:55 ~ 23:50
  - 회의 장소 : 7호관 302호
- 최승렬
  - 회의 시간 : 18:00 ~ 23:45
  - 회의 장소 : 7호관 302호
- 류은환
  - 회의 시간 : 13:00 ~ 23:30
  - 회의 장소 : 7호관 302호

2024.05.26(일)

- 서태원
  - 회의 시간 : 16:30 ~ 23:30
  - 회의 장소 : 7호관 302호
- 신재환
  - 회의 시간 : 16:20 ~ 23:30
  - 회의 장소 : 7호관 302호
- 최승렬
  - 회의 시간 : 14:30 ~ 23:30
- 최승렬
  - 회의 시간 : 13:30 ~ 00:00
  - 회의 장소 : 7호관 302호
- 류은환
  - 회의 시간 : 10:00 ~ 23:30
  - 회의 장소 : 7호관 302호

2024.05.25(토)

- 최승렬
  - 회의 시간 : 14:30 ~ 18:00. 19:30 ~ 23:00
  - 회의 장소 : 7호관 302호

    - 회의 장소 : 7호관 302호
- 류은환
    - 회의 시간 : 16:00 ~ 23:30
    - 회의 장소 : 7호관 302호

총 활동 시간 :

- 서태원 : 41시간 30분
- 신재환 : 50시간 20분
- 최승렬 : 56시간 30분
- 류은환 : 47시간 30분

진행 사항 :

현재 역할

서태원

[업무]

YOLOv9 논문 실험 구현

YOLOv9 논문 분석

YOLOv9 코드 및 연산구조 분석

연산량 및 추론속도 분석 실험

CPU 온도 및 추론속도 영향 실험

류은환

[업무]

YOLOv9 논문 실험 구현

YOLOv9 논문 및 코드 분석

파라미터 분석

경량화 모델 설계 및 실험

PTQ(Post Training Quantization)

신재환

[업무]

Python→C++ 변환

Openvino 구동

Static Quantization

최승렬

[업무]

ONNX 모델 옵션 조합 성능 실험

ONNX to ORT

Pruning 실험

CPU 온도 및 추론속도 영향 실험

- **YOLOv9-c 모델을 기반으로 추론 속도 향상 모델 설계 실험**

yolov9-c-converted 모델에 보조분기를 적용하고, 최대 레이어의 크기를 128로 제한한 모델인 test2의 성능을 확인함.

MODEL NAME	**YOLOv9-c-converted (Paper DL)**	**test2-converted_COCO_300**
mAP50 score:	0.699	0.578
mAP50-95 score:	0.53	0.423
RPI avg inference time:	3334.0ms	1435.2ms

- **YOLOv9 논문 및 코드, 모델 블록 분석**

논문을 바탕으로 모델 구조 파악 및 논문 실험을 재현함.

보조분기 제거 실험 진행, mAP 유지하며 모델 추론 속도 향상 모델 제작 성공.

- **모델 파라미터(weights) 디스플레이 및 수정 코드 작성**

학습된 모델의 전체 파라미터(weights) 디스플레이 및 수정 코드를 작성함.

- **연산량(FLOPs) 및 추론 시간(Speed)을 분석 실험**

추론속도와의 상관관계를 찾고, 이를 바탕으로 모델 구조를 최적화 실험을 계획함.

FLOPS 및 Params와 같은 간접적 지표로 모델 추론시간 향상을 평가하려 하였으나, FLOPS 외에도 Memory Access Cost [ShuffleNet V2 (CVPR 2018**)]** 등이 추론속도에 영향을 미침.

- **RPi 5의 CPU 온도에 따른 추론 속도 실험**

CPU 온도에 따른 RPi 5 성능 하락 분석 및 구동 환경 통제 관련 실험 진행.

yolov9-c-coverted model을 coco128 dataset에 5번 돌려보면서 inference time과 cpu temperature를 나타내고 이에 따른 추세선을 나타내는 그래프를 작성하여 경과를 확인함.

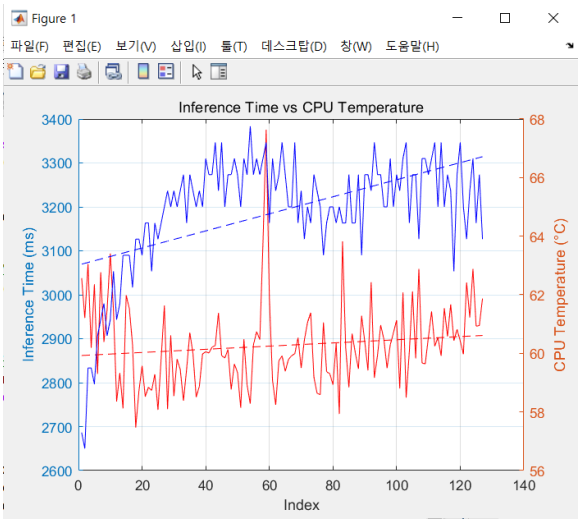
Figure 1
파일(F) 편집(E) 보기(V) 삽입(I) 툴(T) 데스크탑(D) 창(W) 도움말(H)
Inference Time vs CPU Temperature
Inference Time (ms)
CPU Temperature (°C)
Index

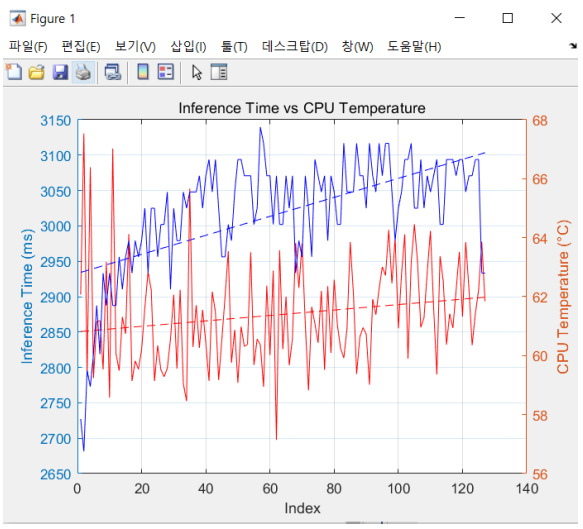
Figure 1
파일(F) 편집(E) 보기(V) 삽입(I) 툴(T) 데스크탑(D) 창(W) 도움말(H)
Inference Time vs CPU Temperature
Inference Time (ms)
CPU Temperature (°C)
Index

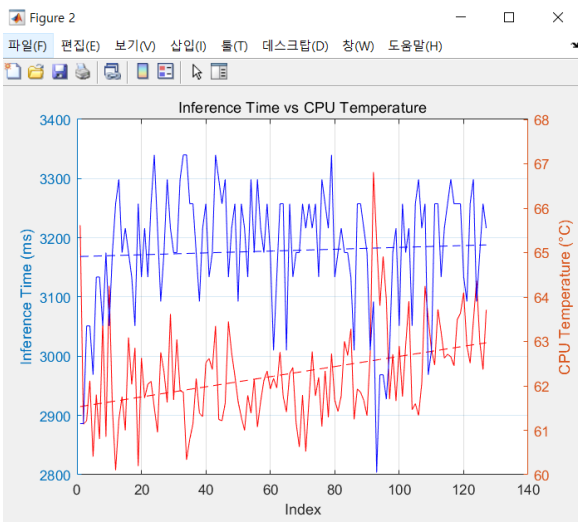
Figure 2
파일(F) 편집(E) 보기(V) 삽입(I) 툴(T) 데스크탑(D) 창(W) 도움말(H)
Inference Time vs CPU Temperature
Inference Time (ms)
CPU Temperature (°C)
Index

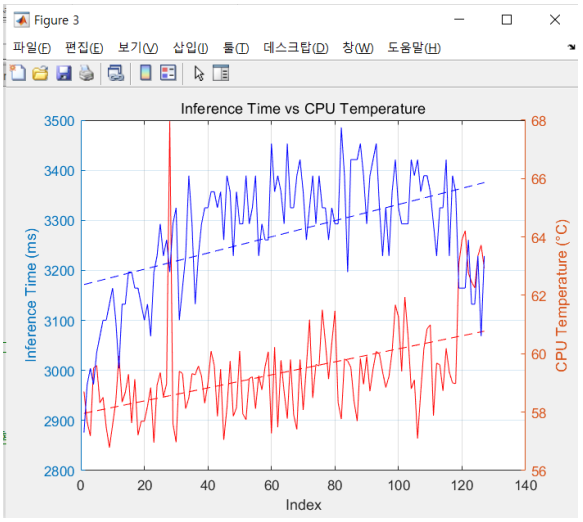
Figure 3
파일(F) 편집(E) 보기(V) 삽입(I) 툴(T) 데스크탑(D) 창(W) 도움말(H)
Inference Time vs CPU Temperature
Inference Time (ms)
CPU Temperature (°C)
Index

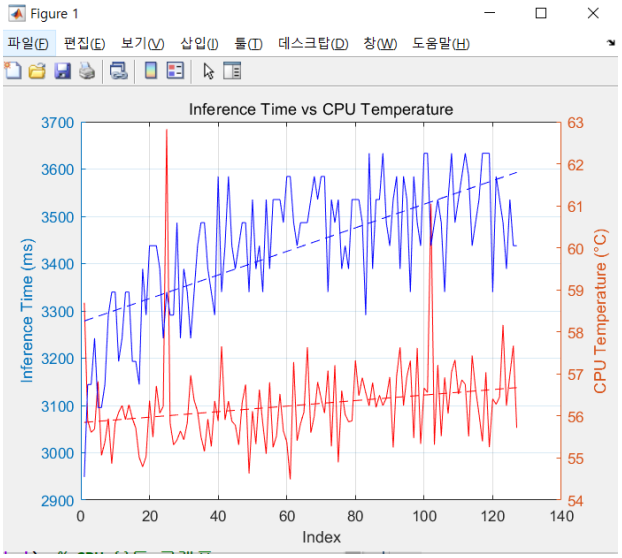
Figure 1
파일(F) 편집(E) 보기(V) 삽입(I) 툴(T) 데스크탑(D) 창(W) 도움말(H)
Inference Time vs CPU Temperature
Inference Time (ms)
CPU Temperature (°C)
Index

- **ONNX 옵션 실험을 이어서 진행**

ONNX의 여러 옵션을 조합하며 RPI5 에서의 추론 속도 향상 실험을 진행함.

옵션 --simplify, --dynamic, --half, --int8 등 사용할 수 있는 옵션 조합을 실험하였고, --simplify와 --dynamic 옵션과 함께 사용 시 RPi5 에서 가장 좋은 성능 결과를 얻음.

MODEL NAME	YOLOv9-c-converted (Paper DL)	YOLOv9-c-converted_onnx_3.7 (Paper DL)	YOLOv9-c-converted_onnx_simplify_dynamic_3.7 (Paper DL)
mAP50 score:	0.699	0.696	0.696
mAP50-95 score:	0.53	0.528	0.528
RPI avg inference time:	3334.0ms	1662.9ms	1622.3ms
RPI max inference time:	3915.7ms	1943.1ms	2087.4ms
RPI min inference time:	3210.6ms	1602.6ms	1576.6ms
RPI avg CPU temp:	71.74C	68.48C	71.42C

- python 버전 별 inference time 비교

python 환경에 따라 추론 속도가 다를 것으로 예상하여 python 3.7 - 3.11 중 가장 빠른 추론 속도를 가지는 python 버전을 찾는 실험을 진행함.

yolov9-c-converted-onnx-3.7.12 : 1662.9ms inference (max: 1943.1ms, min: 1602.6ms)

yolov9-c-converted-onnx-3.8.19 : 1761.5ms inference (max: 2151.0ms, min: 1684.8ms)

yolov9-c-converted-onnx-3.9.19 : 1751.0ms inference (max: 2187.5ms, min: 1711.1ms)

yolov9-c-converted-onnx-3.10.14 : 1744.7ms inference (max: 2191.3ms, min: 1707.4ms)

yolov9-c-converted-onnx-3.11.9 : 1760.0ms inference (max: 2458.4ms, min: 1708.7ms)

python 3.7 버전이 추론속도가 높았기에 python 3.7버전 환경으로 실험을 계속해서 이어나가기로 함.

- **ORT 추가 실험**

ONNX RUNTIME 변환 실험을 진행함. 하지만 ort버전으로 변경한 모델은 python 3.8버전 이상에서 구동되었기에 python 3.8에서의 onnx모델보다 향상이 제대로 되었는지 확인해보고, python 3.7버전onnx와 비교해보았을 때 어느정도의 결과인지 확인해봄.

Python 3.8 ONNX 보다 3.8 ORT 추론속도 향상. (mAP 유지, 1762.5ms → 1689.3ms)

Python 3.7 ONNX 대비 3.8 ORT는 비슷한 성능 결과. (mAP 유지, 1662.9m → 1762.5ms)

- **Pruning 실험**

pretrained 된 가중치를 파인튜닝하여 pruning을 진행하고자 함. 그래서 train.py에 argument를 추가

```
parser.add_argument('--sparsity', type=float, default=0.1, help='가
parser.add_argument('--num_epochs_to_prune', type=int, default=4, h
parser.add_argument('--prune_norm', type=str, default='L1')
```

- --sparsity는 pruning은 어느 정도의 비율로 진행하는지 명시
- --num_epochs_to_prune은 몇 epoch당 pruning을 진행하는지 명시
- --prune_norm은 pruning 옵션으로 L1을 작성
- L1 pruning **:** 가중치의 절대값을 기준으로 중요도를 평가하여 가장 낮은 값들을 제거. 이는 모델에서 덜 중요한 연결을 제거하는데 효과적

```
def prune_model(model, sparsity, norm='L1'):
    for name, param in model.named_parameters():
        if 'weight' in name:
            threshold = torch.quantile(torch.abs(param.data), spars
            mask = torch.abs(param.data) > threshold
            param.data *= mask
```

1. 임곗값 결정: torch.quantile 함수를 사용하여 주어진 sparsity 값에 따라 가중치의 절대값에서 임곗값을 결정. 예를 들어, sparsity가 0.001이면, 가중치의 절대값 중에서 하위 0.1%에 해당하는 값이 임곗값이 됨.
2. 마스크 생성: mask는 param.data의 절대값이 임곗값보다 큰 경우 True, 그렇지 않은 경우 False가 되는 boolean 텐서.
3. 가중치 조정: param.data *= mask는 mask의 False 위치에서 param.data를 0으로 설정함. 즉, mask가 False인 위치의 가중치는 0이 되어 해당 가중치가 실제로 pruning됨

sparsity : 0.001, epoch : 20 조건과 sparsity : 0.005, epoch : 100으로 조건을 두어 실험을 진행함.

이러한 pruning은 파라미터를 직접 제거하는 방식이 아닌 값을 o으로 바꾸는 형식임. 그렇기에 모든 파라미티를 0으로 바꿔서 추론속도에 영향이 있는지 확인을 해보았고, 직접 파라미터를 제거하는 방식보다는 추론속도에 향상 폭이 작은 것을 확인함. 그렇기에 직접 파라미터를 제거하는 pruning 방식을 재고해보기로 함.

- **openvino C++**

CPU 환경에서 C++에 의한 Conv 계산 속도 증가가 추론 속도 향상 시킬 수 있을 것이라 예상하여 YOLOv9 파이썬 코드를 C++ 코드로 변경 후 추론속도를 확인해보고자 하였음.

CMakeLists.txt 파일 작성

- set(OpenCV_DIR "/usr/local/lib/cmake/opencv4") : c++전용 opencv를 설치 후 폴더 경로를 설정하고 디렉토리를 연결해준다.
- set(OPENVINO_DIR "/home/pi/yes/envs/py310") : 가상환경 py310에 openvino 파일을 다운로드 후 경로를 설정해준다. (conda install -c conda-forge openvino=2023.3.0)

add_executable(${PROJECT_NAME} ${SOURCES} ${HEADERS})

- yolov9-openvino로 실행 파일 이름을 설정해줌.

object file이 완성되었으면 link후 실행파일을 만듦.

- # Link libraries
  target_link_libraries(${PROJECT_NAME}
  ${OpenCV_LIBS}
  #${OPENVINO_LIBS}
  ${OPENVINO_DIR}/lib/libopenvino.so
  ${OPENVINO_DIR}/lib/libopenvino_c.so
)

```
cmake_minimum_required(VERSION 3.12)
project(yolov9-openvino)

# Set C++ standard
set(CMAKE_CXX_STANDARD 17)


# Include CUDA directories

# Add source files
set(SOURCES
    main.cpp
```

```
    yolov9_openvino.cpp
)

# Add headers
set(HEADERS
    yolov9_openvino.h
)

# Set your OpenCV path
set(OpenCV_DIR "/usr/local/lib/cmake/opencv4")
find_package(OpenCV REQUIRED)
include_directories(${OpenCV_INCLUDE_DIRS})

# Set your OpenVINO path
set(OPENVINO_DIR "/home/pi/yes/envs/py310")

# Include TensorRT
include_directories(${OPENVINO_DIR}/include)
link_directories(${OPENVINO_DIR}/lib/intel64/release)
set(OPENVINO_LIBS openvino openvino_c)

# Create an executable
add_executable(${PROJECT_NAME} ${SOURCES} ${HEADERS})

# Link libraries
target_link_libraries(${PROJECT_NAME}
    ${OpenCV_LIBS}
    #${OPENVINO_LIBS}
    ${OPENVINO_DIR}/lib/libopenvino.so
    ${OPENVINO_DIR}/lib/libopenvino_c.so
)
```

- sub function.cpp 파일 작성

vector<string>coconame에서 class 종류를 선택해줌.

yolov9에 맞게 먼저 resize해주면서 전처리를 해줌, 이때 모델의 이미지를 downSampling하고 라즈베리 메모리가 수용할 수 있게 만들어 줌.

predict 함수를 작성해주고, 이제 draw함수를 작성하면서 바운딩 박스를 표시해줌.

```
#include "yolov9_openvino.h"
#include <opencv2/dnn.hpp>

const vector<string> coconame = {
    "person",         "bicycle",    "car",           "motorcycle",
    "truck",          "boat",       "traffic light", "fire hydrant"
    "bird",           "cat",        "dog",           "horse",
    "bear",           "zebra",      "giraffe",       "backpack",
    "suitcase",       "frisbee",    "skis",          "snowboard",
    "baseball glove", "skateboard", "surfboard",     "tennis racket
    "fork",           "knife",      "spoon",         "bowl",
    "orange",         "broccoli",   "carrot",        "hot dog",
    "chair",          "couch",      "potted plant",  "bed",
    "laptop",         "mouse",      "remote",        "keyboard",
    "toaster",        "sink",       "refrigerator",  "book",
    "teddy bear",     "hair drier", "toothbrush" };


Resize Yolov9::resize_and_pad(cv::Mat& img)
{
    ov::Shape input_shape = compiled_model.input().get_shape();

    float width = img.cols;
    float height = img.rows;
    float r = float(input_shape[1] / max(width, height));
    int new_unpadW = int(round(width * r));
    int new_unpadH = int(round(height * r));
    Resize resize;
    cv::resize(img, resize.resized_image, cv::Size(new_unpadW, new_

    resize.dw = input_shape[1] - new_unpadW;
    resize.dh = input_shape[2] - new_unpadH;
    cv::Scalar color = cv::Scalar(100, 100, 100);
    cv::copyMakeBorder(resize.resized_image, resize.resized_image,

    return resize;
}

Yolov9::Yolov9(const string &model_path)
{
    // Step 1. Initialize OpenVINO Runtime core
```

```
    ov::Core core;
    // Step 2. Read a model
    std::shared_ptr<ov::Model> model = core.read_model(model_path);
    // Step 3. Inizialize Preprocessing for the model
    ov::preprocess::PrePostProcessor ppp = ov::preprocess::PrePostP
    // Specify input image format
    ppp.input().tensor().set_element_type(ov::element::u8).set_layo
    // Specify preprocess pipeline to input image without resizing
    ppp.input().preprocess().convert_element_type(ov::element::f32)
    //  Specify model's input layout
    ppp.input().model().set_layout("NCHW");
    // Specify output results format
    ppp.output().tensor().set_element_type(ov::element::f32);
    // Embed above steps in the graph
    model = ppp.build();
    compiled_model = core.compile_model(model, "CPU");

    // Create random colors
    random_device rd;
    mt19937 gen(rd());
    uniform_int_distribution<int> dis(100, 255);
    for (int i = 0; i < coconame.size(); i++)
    {
        Scalar color = Scalar(dis(gen), dis(gen), dis(gen));
        colors.push_back(color);
    }
}

void Yolov9::predict(cv::Mat &img, std::vector<Detection> &output)
{
    // Step 5. Create tensor from image
    float* input_data = (float*)img.data;
    ov::Tensor input_tensor = ov::Tensor(compiled_model.input().get

    // Step 6. Create an infer request for model inference
    ov::InferRequest infer_request = compiled_model.create_infer_re
    infer_request.set_input_tensor(input_tensor);
    infer_request.infer();

    //Step 7. Retrieve inference results
    const ov::Tensor& output_tensor = infer_request.get_output_tens
    ov::Shape output_shape = output_tensor.get_shape();
```

```
    float* detections = output_tensor.data<float>();

    // Step 8. Postprocessing including NMS
    vector<Rect> boxes;
    vector<int> class_ids;
    vector<float> confidences;

    const Mat det_output(output_shape[1], output_shape[2], CV_32F,

    for (int i = 0; i < det_output.cols; ++i) {
        const Mat classes_scores = det_output.col(i).rowRange(4, ou
        Point class_id_point;
        double score;
        minMaxLoc(classes_scores, nullptr, &score, nullptr, &class_

        if (score > CONFIDENCE_THRESHOLD) {
            const float cx = det_output.at<float>(0, i);
            const float cy = det_output.at<float>(1, i);
            const float ow = det_output.at<float>(2, i);
            const float oh = det_output.at<float>(3, i);
            Rect box;
            box.x = static_cast<int>((cx - 0.5 * ow));
            box.y = static_cast<int>((cy - 0.5 * oh));
            box.width = static_cast<int>(ow);
            box.height = static_cast<int>(oh);

            boxes.push_back(box);
            class_ids.push_back(class_id_point.y);
            confidences.push_back(score);
        }
    }

    vector<int> nms_result;
    dnn::NMSBoxes(boxes, confidences, CONFIDENCE_THRESHOLD, NMS_THR

    for (int i = 0; i < nms_result.size(); i++)
    {
        Detection result;
        int idx = nms_result[i];
        result.class_id = class_ids[idx];
        result.confidence = confidences[idx];
```

```
        result.box = boxes[idx];
        output.push_back(result);
    }
}

void Yolov9::draw(Mat& img, vector<Detection>& output, float dw, fl
{
    // Step 9. Print results and save Figure with detections
    ov::Shape input_shape = compiled_model.input().get_shape();

    for (int i = 0; i < output.size(); i++)
    {
        auto detection = output[i];
        auto box = detection.box;
        auto classId = detection.class_id;
        auto confidence = detection.confidence;
        float rx = (float)img.cols / (float)(input_shape[1] - dw);
        float ry = (float)img.rows / (float)(input_shape[2] - dh);
        box.x = rx * box.x;
        box.y = ry * box.y;
        box.width = rx * box.width;
        box.height = ry * box.height;
        float xmax = box.x + box.width;
        float ymax = box.y + box.height;

        rectangle(img, Point(box.x, box.y), Point(box.x + box.width

        // Detection box text
        string class_string = coconame[classId] + ' ' + to_string(c
        Size text_size = getTextSize(class_string, FONT_HERSHEY_DUP
        Rect text_rect(box.x, box.y - 40, text_size.width + 10, tex
        rectangle(img, text_rect, colors[classId], FILLED);
        putText(img, class_string, Point(box.x + 5, box.y - 10), FO
    }
}

void Yolov9::setConf(float conf)
{
    CONFIDENCE_THRESHOLD = conf;
}

void Yolov9::setNMS(float nms)
```

```
{
    NMS_THRESHOLD = nms;
}
```

- main code 작성 → image, images, video 파일을 주면 추론 시간과 fps를 출력해주는 코드를 작성

```
#ifdef _WIN32
#include <windows.h>
#else
#include <sys/stat.h>
#include <unistd.h>
#endif

#include <iostream>
#include <string>
#include "yolov9_openvino.h"
#include <chrono>


bool IsPathExist(const string& path) {
#ifdef _WIN32
    DWORD fileAttributes = GetFileAttributesA(path.c_str());
    return (fileAttributes != INVALID_FILE_ATTRIBUTES);
#else
    return (access(path.c_str(), F_OK) == 0);
#endif
}
bool IsFile(const string& path) {
    if (!IsPathExist(path)) {
        printf("%s:%d %s not exist\n", __FILE__, __LINE__, path.c_s
        return false;
    }

#ifdef _WIN32
    DWORD fileAttributes = GetFileAttributesA(path.c_str());
    return ((fileAttributes != INVALID_FILE_ATTRIBUTES) && ((fileAt
#else
    struct stat buffer;
    return (stat(path.c_str(), &buffer) == 0 && S_ISREG(buffer.st_m
#endif
```

```
}

int main(int argc, char** argv)
{
    const string model_file_path{ argv[1] };
    const string path{ argv[2] };
    vector<string> imagePathList;
    bool                    isVideo{ false };
    assert(argc >= 3);

    float conf_thresh = 0.2;
    float nms_thresh = 0.3;
    if (argc > 3)
    {
        conf_thresh = std::stof(argv[3]);
    }
    else if (argc > 4)
    {
        nms_thresh = std::stof(argv[3]);
    }

    if (IsFile(path))
    {
        string suffix = path.substr(path.find_last_of('.') + 1);
        if (suffix == "jpg" || suffix == "jpeg" || suffix == "png")
        {
            imagePathList.push_back(path);
        }
        else if (suffix == "mp4" || suffix == "avi" || suffix == "m
        {
            isVideo = true;
        }
        else {
            printf("suffix %s is wrong !!!\n", suffix.c_str());
            abort();
        }
    }

    else if (IsPathExist(path))
    {
        glob(path + "/*.jpg", imagePathList);
    }
```

```
// Assume it's a folder, add logic to handle folders
// init model
Yolov9 model(model_file_path);
model.setConf(conf_thresh);
model.setNMS(nms_thresh);

if (isVideo) {
    //path to video
    string VideoPath = path;
    // open cap
    VideoCapture cap(VideoPath);



    int width = cap.get(CAP_PROP_FRAME_WIDTH);
    int height = cap.get(CAP_PROP_FRAME_HEIGHT);

    // Create a VideoWriter object to save the processed video
    VideoWriter output_video("output_video.avi", VideoWriter::f

    int frameCount = 0;
    double fps = 0.0;
    double startTime = (double)getTickCount();

    while (1)
    {
        Mat frame;
        cap >> frame;

        if (frame.empty()) break;
        Resize res = model.resize_and_pad(frame);
        vector<Detection> bboxes;
        model.predict(res.resized_image, bboxes);
        model.draw(frame, bboxes, res.dw, res.dh);

        frameCount++;
        double currentTime = ((double)getTickCount() - startTim
        if (currentTime >= 1.0) {
            fps = frameCount / currentTime;
            frameCount = 0;
            startTime = (double)getTickCount();
```

```
            }
        string fpsText = format("FPS: %.2f", fps);
        putText(frame, fpsText, Point(10, 30), FONT_HERSHEY_SIMPLEX


            cv::imshow("prediction", frame);
            output_video.write(frame);
            cv::waitKey(1);
        }

        // Release resources
        cv::destroyAllWindows();
        cap.release();
        output_video.release();
    }

    else {
        // path to folder saves images
        //string imageFolderPath_out = "results/";
        for (const auto& imagePath : imagePathList)
        {
            // open image
            Mat frame = imread(imagePath);
            if (frame.empty())
            {
                cerr << "Error reading image: " << imagePath << end
                continue;
            }
            auto start = std::chrono::high_resolution_clock::now();
            //cout << "save"<<endl;

            Resize res = model.resize_and_pad(frame);

            vector<Detection> bboxes;
            model.predict(res.resized_image, bboxes);
            model.draw(frame, bboxes, res.dw, res.dh);

            auto end = std::chrono::high_resolution_clock::now();
            auto duration = std::chrono::duration_cast<std::chrono:

            cout << "Inference time for image " << imagePath << ":
```

```
            //istringstream iss(imagePath);
            // string token;
            // while (getline(iss, token, '/'))
            // {
            // }
            //imwrite(imageFolderPath_out + token, frame);
            //std::cout << imageFolderPath_out + token << endl;

            // cv::imshow("prediction", frame);
            // cv::waitKey(0);
        }
    }

    return 0;
}
```

비고

- 05/23에 진행했던 교수님 미팅 때의 피드백을 바탕으로 yolov9 모델에 문제점 및 취약점을 파악하여 대체할 만한 새로운 연산 블럭을 만들어 도입 및 교체하여 적용해보고자 함.
- 기존에 실험했던 test1 실험을 논문에서 제공하는 coco dataset으로 학습을 다시시키기로 함.

추후 계획

- YOLOv7 시리즈를 참조하여 YOLOv9-Tiny와 유사한 모델 설계 실험을 진행하고, 이 중 activation 함수를 SILU에서 LeakyReLU로 변경하고, 비율에 맞추어 레이어 크기 감소 및 보조분기를 적용한 test10의 성능 실험
- 계층(Layer)별 파라미터 분석 및 활성화 정도 반영하여 모델 구조 설계 실험 진행.
- RPI5 환경에서 MAC 최적화 모델 설계 관련 실험.
- 추가 실험을 통해 최적화 모델 성능 향상
- 기존의 조사 및 공부, 실험한 내용을 바탕으로 연구 개발 과정과, 실험 과정 및 결과를 정리하여 졸업논문 형식으로 초안 작성