INFOWORKS

AI kit

SSD+MobileNet 예제 사용설명서

목 차

01. 딥러닝 기반 어플리케이션 개발

02. Ai kit 사용방법

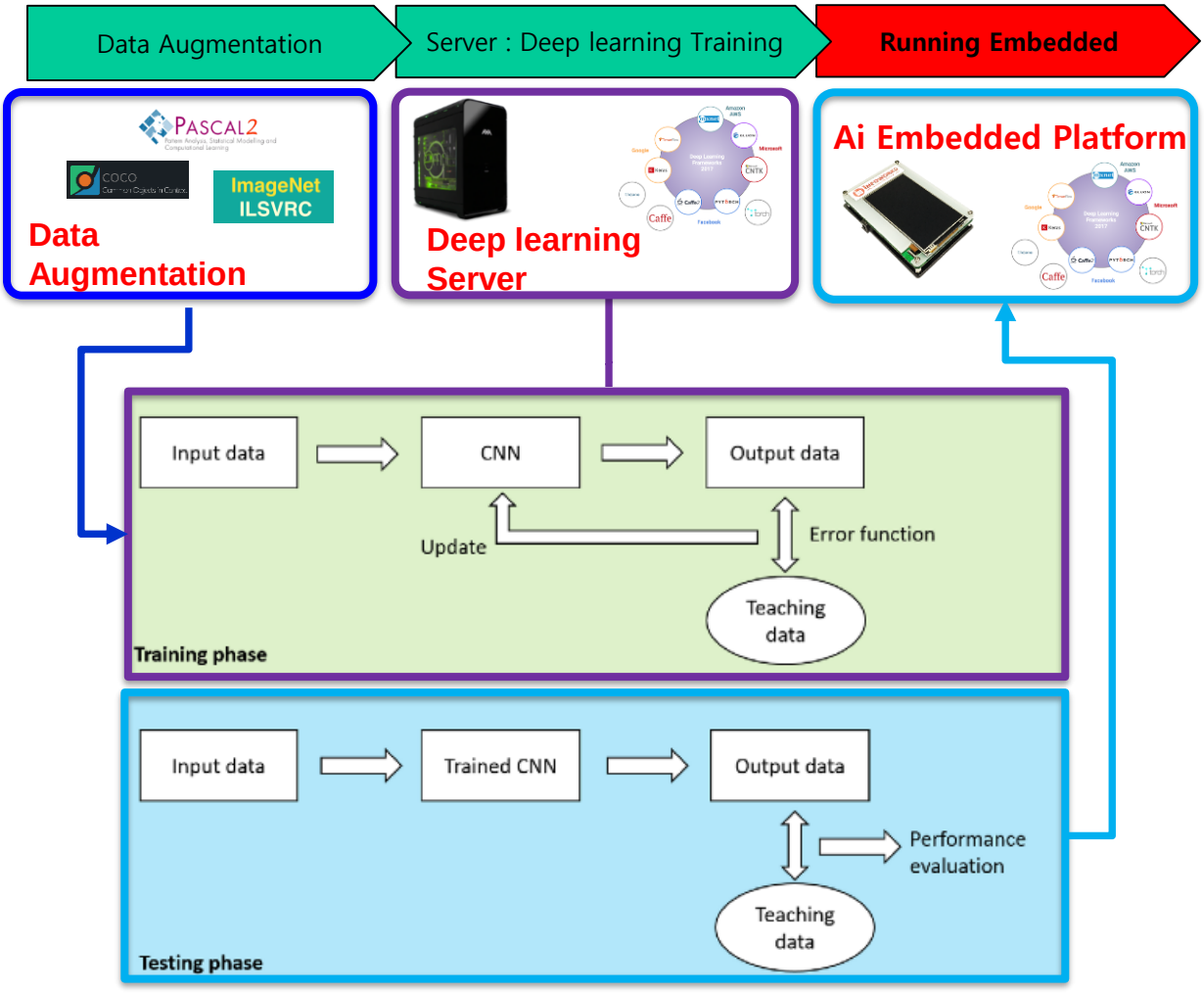
03. 학습 데이터 생성

04. SSD+MobileNet 예제

01.

딥러닝 기반 어플리케이션 개발

01. 딥러닝 기반 어플리케이션 개발 절차 1



전형적인 딥러닝은 두가지 단계를 거친다.

1. 훈련 단계

훈련단계에서, 컨볼루션 신경망으로부터 출력 데이터는 티칭 데이터와 함께 에러 함수로 공급 된다.

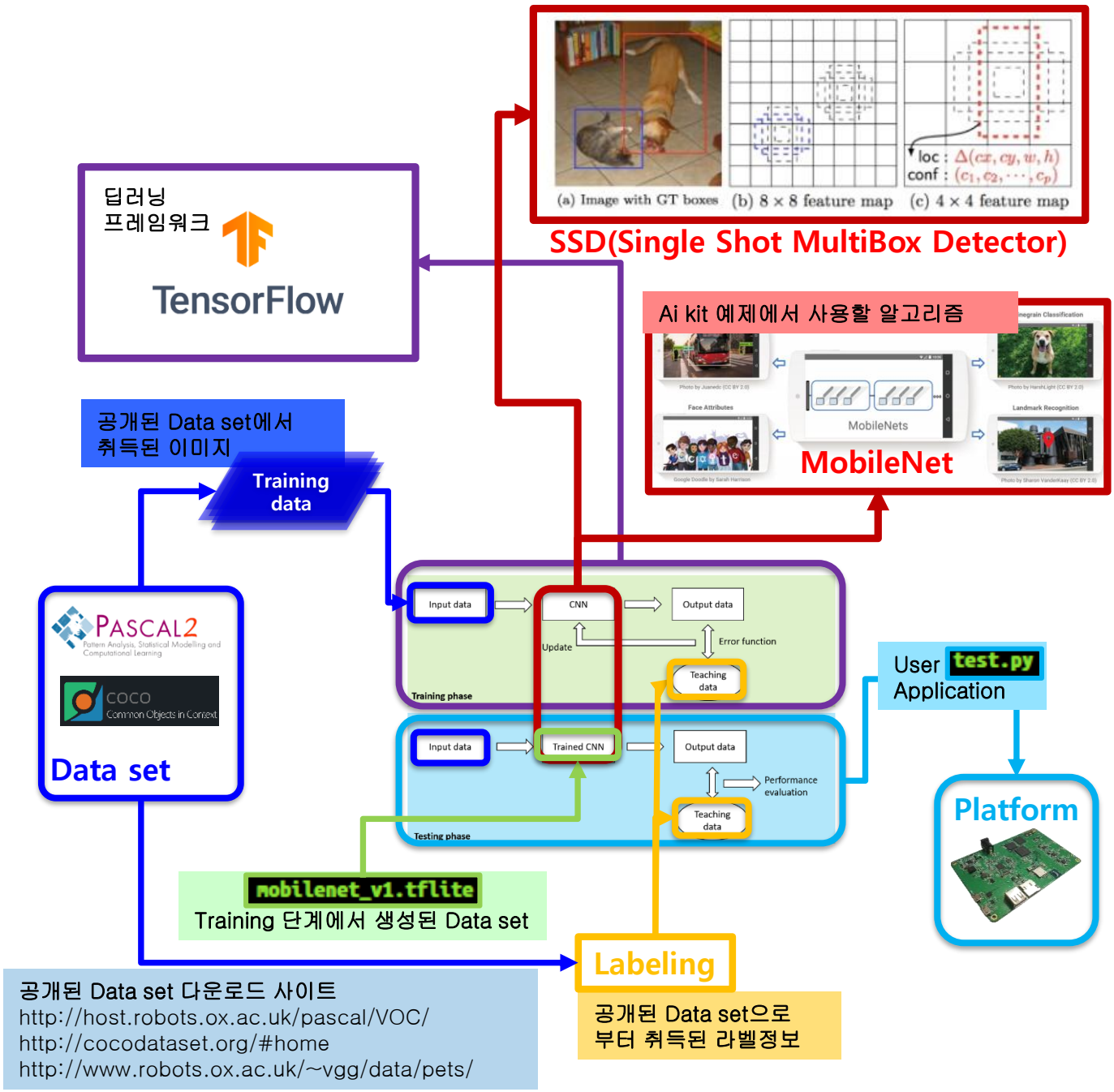
CNN내의 매개 변수는 반복된 학습을 통해서 계속 오류가 작아지도록 조정되며 이렇게 훈련된 컨볼루션 뉴런 네트워크가 생성되게 된다.

2. 테스트 단계

테스트 단계에서는 훈련데이터와 별도로 준비된 입력 데이터가 훈련된 컨볼루션 뉴런네트워크를 통해서 입력에 대한 출력이 이루어지게 된다.

01. 딥러닝 기반 어플리케이션 개발 절차 2

딥러닝 개발절차를 위한 세부 사항

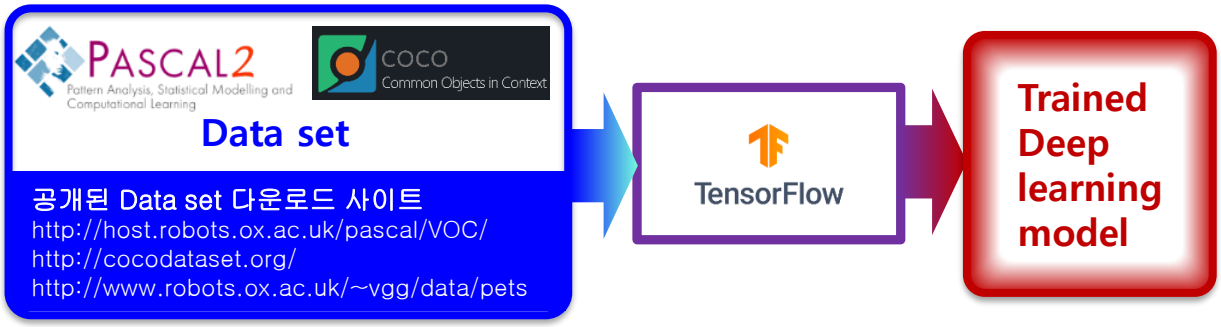


텐서플로우 프래임워크를 이용하여 공개된 Data set을 기반으로 SSD+MobilNet 트레이닝 모델을 만들고 생성된 트레이닝 모델을 플랫폼에서 사용할 수 있는 모델로 변경 후 어플리케이션을 이용하여 이를 수행 한다

01. 딥러닝 기반 어플리케이션 개발 절차 3

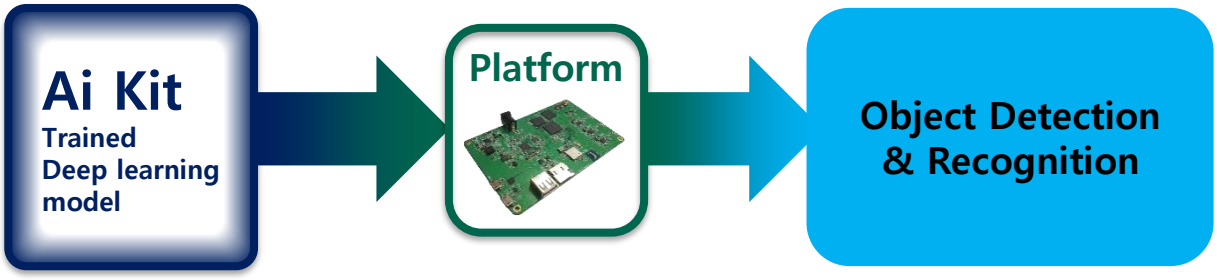
Ai kit내 SSD+MobileNet 예제를 사용하기 위한 3 단계

1. 인공지능framework(Tensorflow)와 공개된 Dataset (PascalVOC or MSCOCO etc..)을 이용해 Deeplearning Model을 생성



2. 생성된 모델을 4장에서 소개하는 예제를 이용해 Ai kit 플랫폼에 호환되는 모델로 변경

3. 예제 코드를 이용해 NPU에 모델 및 입력 데이터 로드 후 인공지능 모델 구동



02.

Ai kit 사용방법

SDK 다운로드

1.컴파일에 필요한 패키지를 설치

```
sudo apt-get install repo git u-boot-tools device-tree-compiler mtools parted libudev-dev libusb-1.0-0-dev lib32gcc-7-dev python-linaro-image-tools linaro-image-tools gcc-arm-linux-gnueabihf gcc-aarch64-linux-gnu libstdc++-7-dev autoconf autotools-dev libsigsegv2 m4 intltool libdrm-dev curl sed make binutils build-essential gcc g++ bash patch gzip bzip2 perl tar cpio python unzip rsync file bc wget libncurses5 libqt4-dev libglib2.0-dev libgtk2.0-dev libglade2-dev cvs mercurial rsync openssh-client subversion asciidoc w3m dblatex graphviz python-matplotlib libssl-dev pv e2fsprogs fakeroot devscripts libi2c-dev libncurses5-dev texinfo liblz4-tool genext2fs
```

2. Repo 다운로드

Repo는 구글에서 제공하는 **git 저장소 관리를 위한 유틸리티**로, manifest라는 설정 파일을 통해 pull, push, review 등의 작업을 설정할 수 있습니다.

```
git clone https://github.com/rockchip-linux/repo
mkdir linux && cd linux
```

3. Repo를 통해 레포지토리 안에 있는 서브모듈 다운로드

```
../repo/repo init --repo-url=https://github.com/rockchip-linux/repo -u https://github.com/rockchip-linux/manifests -b master -m rk1808_linux_release.xml
../repo/repo sync
```

SDK compile

1. 아래 명령을 통해 SDK를 컴파일 합니다

```
./build.sh
```

최초 컴파일 시 1시간 30분 정도 소요됩니다.

2. 컴파일 완료 시 [SDK 폴더 루트]/IMAGE 폴더 내에 아래와 같은 폴더가 생성됩니다.

```
RK1808-EVB-V10_YYYYMMDD.HHMM_RELEASE_TEST
```

· 이 폴더 내의 IMAGES 폴더에 업데이트에 필요한 파일들이 저장됩니다.

```
## ln {SDK_ROOT}/IMAGE/{RK1808-EVB-V10_YYYYMMDD.HHMM_RELEASE_TEST}/IMAGES

$ ls boot.img MiniLoaderAll.bin misc.img oem.img parameter.txt recovery.img rootfs.ext4 rootfs.img trust.img uboot.img update.img userdata.img
```

· 각 파일들의 역할 및 주소는 아래와 같습니다.

File name	Description	Load Address
MiniLoaderAll .bin	CPU Boot ROM에 의해 로드되는 첫번째 부트로더 입니다.	
parameter.txt	커널에 전달할 부팅 파라미터와 파티션 정보를 담고 있습니다.	
trust.img	ARM Trusted Firmware 입니다. (https://github.com/ARM-software/arm-trusted-firmware)	
misc.img	misc 파티션 이미지로, 안드로이드 부팅 모드를 제어하는데 사용됩니다.	
kernel.img	리눅스 커널 이미지 입니다.	
resource.img	디바이스 트리와 부트 로고 파일을 담고 있는 이미지 입니다.	
boot.img	initramfs 이미지로, 초기화 등의 작업을 수행하는 파일이 담겨있습니다.	
rootfs.img	루트 파일 시스템 이미지입니다.	
update.img	위의 패키지들을 패키징 해 놓은 업데이트용 이미지 입니다.	

Update Tool 복사

· 펌웨어 업데이트 툴을 복사합니다.

```
## In SDK_ROOT
sudo cp
tools/linux/Linux_Upgrade_Tool/Linux_Upgrade_Tool/upgrade_tool
/usr/bin/rk_upgrade_tool
```

Maskrom Mode 진입 및 펌웨어 다운로드

· 아래의 순서로 RK1808을 Maskrom Mode로 진입시킵니다.

1. 보드의 전원 및 USB 연결을 해제시킨다.
2. 보드에 "MARKROM" 이라고 기재되어 있는 버튼을 누른 상태에서 전원을 인가한다.
3. 버튼을 누른 상태에서 약 5초간 기다린 후 버튼에서 손을 뗀다.

Maskrom Mode 진입 확인을 위해서 rk_upgrade_tool을 실행합니다.
· 아래와 같이 RK1808이 인식되고, Maskrom 상태가 표시되면 정상적으로 진입한 것입니다..

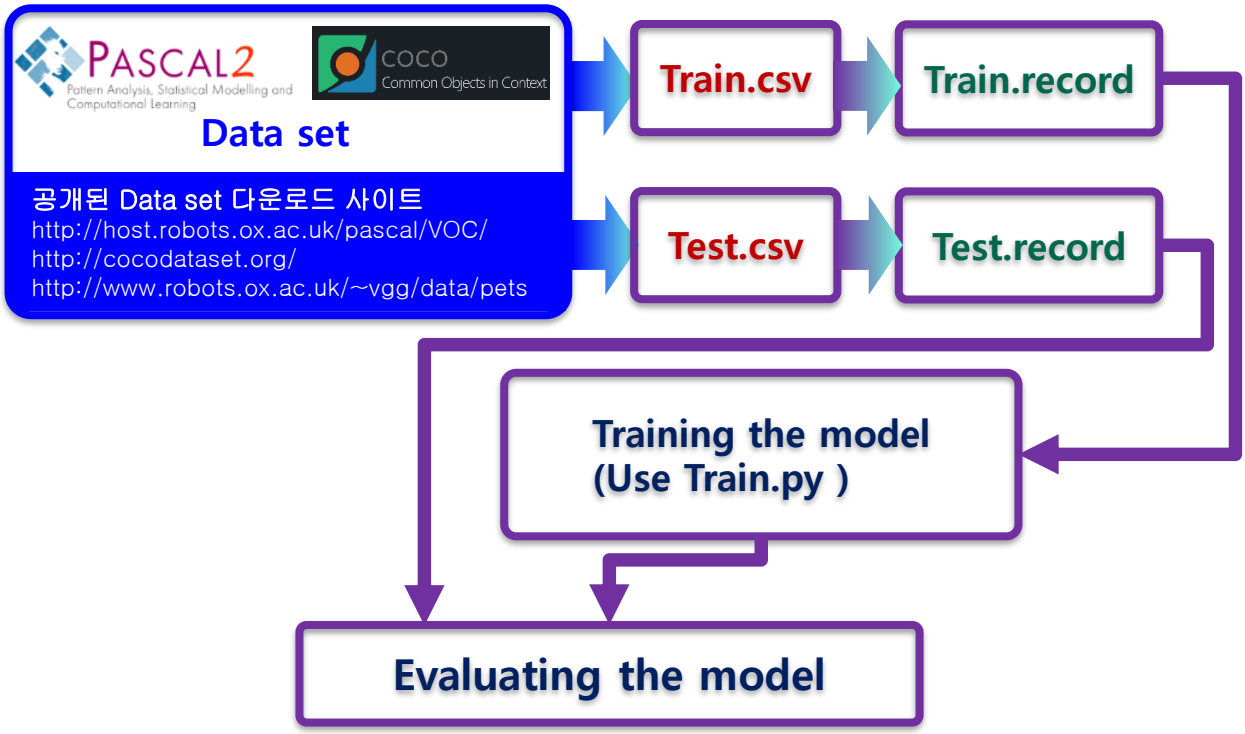
인식이 되지 않은 경우, 위의 순서대로 다시 시도하여 Maskrom Mode로 재진입 시키거나 아래의 방법을 시도해 볼 수 있습니다.

```
echo "SUBSYSTEM==₩"usb₩", ATTR{idVendor}==₩"2207₩",
MODE=₩"0666₩", GROUP=₩"plugdev₩"" | sudo tee
/etc/udev/rules.d/53-rk-rockusb.rules udevadm control --reload-
rules && udevadm trigger
```

update.img 파일이 있는 폴더로 이동 후, 아래의 명령을 실행합니다.

```
rk_upgrade_tool uf update.img
```

딥러닝 학습 모델 생성 방법



1. tensorflow 다운로드.
   git clone https://github.com/tensorflow/models.git
2. Data set 의 이미지로부터 Image 의 라벨링 정보와 크기 등의 정보를 담은 csv를 추출
3. csv파일을 TFRecord(.record)로 변환
   (TFRecord : Tensorflow에서 사용하는 데이터 형식)
4. Train.py를 이용해서 Training
   - **ssd_mobilenet_v1_coco**를 이용해서 트레이닝
5. 훈련시킨 모델을 평가한다.

자세한 내용은 하기 링크를 참조

https://becominghuman.ai/tensorflow-object-detection-api-tutorial-training-and-evaluating-custom-object-detector-ed2594afcf73

04.

SSD+MobileNet 예제

04. SSD+MobileNet 예제 목록

1. 학습된 데이터 Ai kit에서 사용할 모델(.rknn)로 변환 예제

https://github.com/rockchip-linux/rknn-toolkit/tree/master/example/rknn_convert

2. MobileNet_V1.tflite 파일을 Ai kit 전용 모델 mobilenet_v1.rknn로 변환 후 성능 테스트 예제

https://github.com/rockchip-linux/rknn-toolkit/tree/master/example/mobilenet_v1

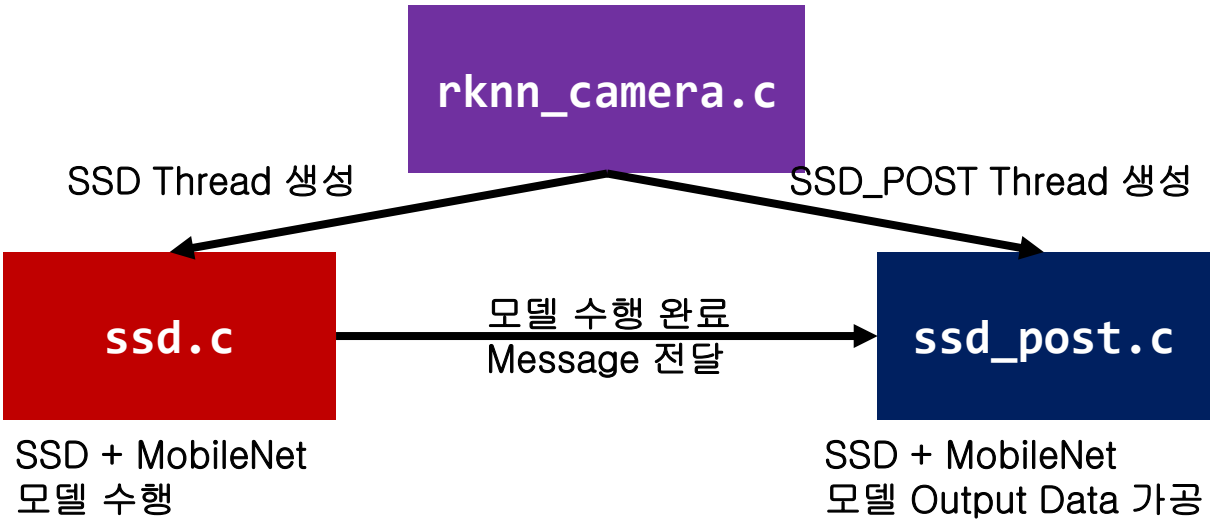
3. ssd_mobilenet_v2.pb 파일을 Ai kit 전용 모델 ssd_mobilenet_v2.rknn로 변환 후 성능 테스트 예제

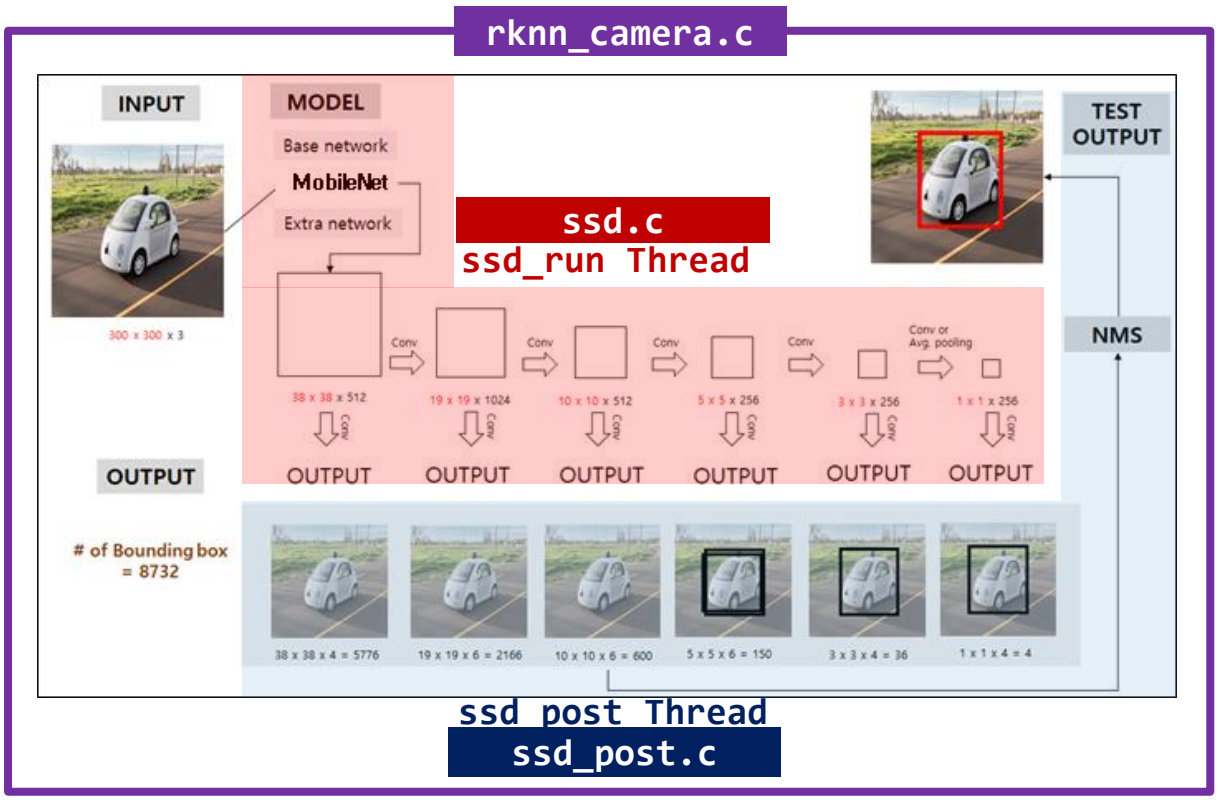
https://github.com/rockchip-linux/rknn-toolkit/tree/master/example/ssd_mobilenet_v2

4. 연결된 카메라를 이용한 SSD+MobileNet 예제

https://github.com/rockchip-linux/rknn_demo수정예정

04. SSD+MobileNet 실습 예제 분석 - 전체구성





rknn_camera.c
카메라영상을 기준으로 SSD_MobileNet을 수행하기 위한 메인

ssd.c
SSD_MobileNet 모델을 수행하기위한 Thread

ssd_post.c
SSD_MobileNet 모델의 수행결과를 출력하기 위한 Thread

04. SSD+MobileNet 실습 예제 분석 - Main

\rknn_demo\rknn_camera.c

```
int MiniGUIMain(int argc, const char* argv[])
{
    struct stat st;

    parse_args(argc, argv);
    if (strcmp(cam_device, "usb") == 0)
    dev_name = get_device("uvc");

    if (strcmp(cam_device, "mipi") == 0)
        dev_name = get_device("rkisp");

    if (!dev_name) {
    printf("do not get usb camera or mipi camera vide node, use image to run rknn_demo\n");
    dev_name = "/usr/local/share/rknn_demo/resource/test_image.nv12";
    if (-1 == stat(dev_name, &st)) {
        fprintf(stderr, "Cannot identify '%s': %d, %s\n", dev_name, errno, strerror(errno));
        exit(EXIT_FAILURE);
    }
    }

    rknn_demo_init();
    rknn_ui_show();
    rknn_demo_deinit();

    exit(0);
}
```

알고리즘을 수행하기 위한 Thread 설정

MiniGUIMain
어플리케이션의 메인 함수

rknn_demo_init
SSD관련 Thread 초기화

rknn_ui_show
Ui 설정

rknn_demo_deinit
SSD관련 Thread 해제

04. SSD+MobileNet 실습 예제 분석 - Main

\rknn_demo\rknn_camera.c

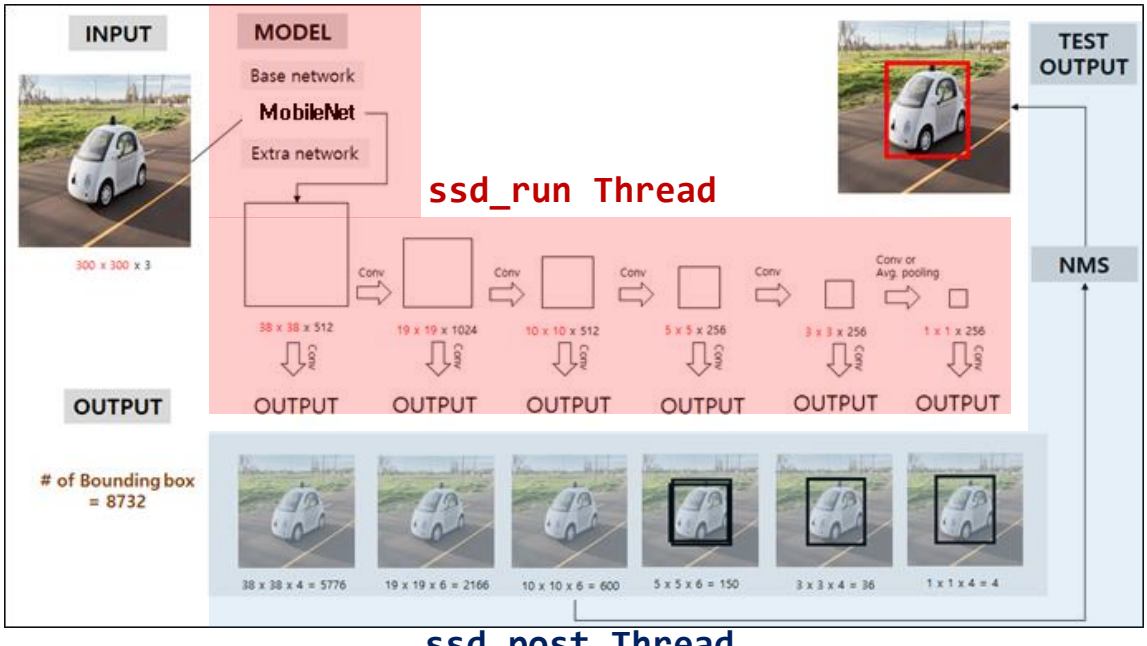
```
int rknn_demo_init()
{
    rknn_callback_t post;
    rknn_callback_t run;

    ssd_init(dev_name);
    post = ssd_post;
    run = ssd_run;

    rknn_post_pth_create(post);
    rknn_run_pth_create(run);

    if (!check_uvc_video_id()) {
      mpi_enc_set_format(MPP_FMT_YUV420P);
      add_uvc_video();
      register_callback_for_uvc(uvc_read_camera_buffer);
    }
}
```

rknn_post_pth_create(post); **ssd_post Thread** 실행
rknn_run_pth_create(run); **ssd_run Thread** 실행

SSD는 두개의 Thread를 기반으로 동작

ssd_run Thread
딥러닝 연산을 통해서 네트워크 결과물 산출

ssd_post Thread
ssd_run Thread 의 출력 데이터를 이용해서 Recognition 정보와 Dectection 정보를 생성



04. SSD+MobileNet 실습 예제 분석 ssd_run Thread

\rknn_demo\rknn\ssd\ssd_1808\ssd.c

```
int ssd_run(void *flag)
{
    int status = 0;
    int model_len = 0;
    unsigned char* model;

    model = load_model(MODEL_NAME, &model_len);

    status = rknn_init(&ctx, model, model_len, 0);
    if(status < 0) {
        printf("rknn_init fail! ret=%d\n", status);
        return -1;
    }

    // Get Model Input Output Info
    rknn_input_output_num io_num;
    status = rknn_query(ctx, RKNN_QUERY_IN_OUT_NUM, &io_num, sizeof(io_num));
    if (status != RKNN_SUCC) {
        printf("rknn_query fail! ret=%d\n", status);
        return -1;
    }
    printf("model input num: %d, output num: %d\n", io_num.n_input, io_num.n_output);

    printf("input tensors:\n");
    rknn_tensor_attr input_attrs[io_num.n_input];
    memset(input_attrs, 0, sizeof(input_attrs));
    for (int i = 0; i < io_num.n_input; i++) {
        input_attrs[i].index = i;
        status = rknn_query(ctx, RKNN_QUERY_INPUT_ATTR, &(input_attrs[i]), sizeof(rknn_tensor_attr));
        if (status != RKNN_SUCC) {
            printf("rknn_query fail! ret=%d\n", status);
            return -1;
        }
    }
    printf("output tensors:\n");
    rknn_tensor_attr output_attrs[io_num.n_output];
    memset(output_attrs, 0, sizeof(output_attrs));
    for (int i = 0; i < io_num.n_output; i++) {
        output_attrs[i].index = i;
        status = rknn_query(ctx, RKNN_QUERY_OUTPUT_ATTR,&(output_attrs[i]),sizeof(rknn_tensor_attr));
        if (status != RKNN_SUCC) {
            printf("rknn_query fail! ret=%d\n", status);
            return -1;
        }
    }

    cameraRun(dev_name, SRC_W, SRC_H, SRC_FPS, SRC_V4L2_FMT,
              ssd_camera_callback, (int*)flag);

    // Release
    if(model) {
        free(model);
    }
    if(ctx > 0) {
        rknn_destroy(ctx);
    }
    return status;
}
```

미리 생성한 트레이닝된 모델을 로드 **(mobilenet_ssd.rknn)**

트레이닝된 모델의 입출력 관련 속성 취득

SSD 알고리즘을 수행 콜백함수

load_model, rknn_init
함수를 이용해서 미리 생성한 **트레이닝된 모델을** 로드

rknn_query
RKNN_QUERY_IN_OUT_NUM, RKNN_QUERY_INPUT_ATTR, RKNN_QUERY_OUTPUT_ATTR
세 옵션을 통해서 입력 텐서와 출력 텐서의 속성을 취득

cameraRun
카메라영상 취득 및 callback함수를 통해 SSD 알고리즘을 수행함

04. SSD+MobileNet 실습 예제 분석 ssd_run Thread

\rknn_demo\rknn\ssd\ssd_1808\ssd.c

```
void ssd_camera_callback(void *p, int w, int h, int *flag)
{
    unsigned char* srcbuf = (unsigned char *)p;
    // Send camera data to minigui Layer
    yuv_draw(srcbuf, 0, SRC_RKRGA_FMT, w, h);
    cal_fps(&g_fps);

    if (*flag) {
    YUV420toRGB24_RGA(SRC_RKRGA_FMT, srcbuf, w, h,
            DST_RKRGA_FMT, g_rga_buf_fd, DST_W, DST_H);
    memcpy(g_mem_buf, g_rga_buf_bo.ptr, DST_W * DST_H * DST_BPP / 8);
        ssd_rknn_process(g_mem_buf, DST_W, DST_H, DST_BPP);
    }
}
```

SSD 알고리즘 메인 프로세스

ssd_camera_callback 함수에서 **ssd_rknn_process** 함수를 호출함으로써 SSD의 해당 네트워크연산이 수행됨

04. SSD+MobileNet 실습 예제 분석 ssd_rknn_process

\rknn_demo\rknn\ssd\ssd_1808\ssd.c

```
int ssd_rknn_process(char* in_data, int w, int h, int c)
{
    int status = 0;
    int in_size;
    int out_size0;
    int out_size1;
    float *out_data0 = NULL;
    float *out_data1 = NULL;
  //    printf("camera callback w=%d h=%d c=%d\n", w, h, c);
    long runTime1 = getCurrentTime();
    long setinputTime1 = getCurrentTime();
    // Set Input Data
    rknn_input inputs[1];
    memset(inputs, 0, sizeof(inputs));
    inputs[0].index = 0;
    inputs[0].type = RKNN_TENSOR_UINT8;
    inputs[0].size = w*h*c/8;
    inputs[0].fmt = RKNN_TENSOR_NHWC;
    inputs[0].buf = in_data;

    status = rknn_inputs_set(ctx, 1, inputs);
        printf("rknn_input_set fail! ret=%d\n", status);
        return -1;
    }
    long setinputTime2 = getCurrentTime();
    // printf("set input time:%0.2lfms\n", setinputTime2-setinputTime1);

    status = rknn_run(ctx, NULL);
        printf("rknn_run fail! ret=%d\n", status);
        return -1;
    }
    // Get Output
    rknn_output outputs[2];
    memset(outputs, 0, sizeof(outputs));
    outputs[0].want_float = 1;
    outputs[1].want_float = 1;
    status = rknn_outputs_get(ctx, 2, outputs, NULL);
        printf("rknn_outputs_get fail! ret=%d\n", status);
        return -1;
    }
    int out0_elem_num = NUM_RESULTS * NUM_CLASS;
    int out1_elem_num = NUM_RESULTS * 4;
    float *output0 = malloc(out0_elem_num*sizeof(float));
    float *output1 = malloc(out1_elem_num*sizeof(float));
    memcpy(output0, outputs[0].buf, out0_elem_num*sizeof(float));
    memcpy(output1, outputs[1].buf, out1_elem_num*sizeof(float));

    rknn_outputs_release(ctx, 2, outputs);

    long runTime2 = getCurrentTime();
    // printf("rknn run time:%0.2lfms\n", runTime2 - runTime1);

    // long postprocessTime1 = getCurrentTime();

    int ret = rknn_msg_send(void *)output1,(void *)output0, w, h, &g_ssd_group[!cur_group]);
    if (ret
        return -1;
    while(send_count - recv_count >= 5) {
        printf("sleep now \n");
        usleep(2000);
    }
    send_count++;
}
```

rknn_query로 취득된 입력 출력 정보 세팅

SSD model이 수행 되는 부분

SSD_post에 메시지 전달

앞서 설명한 **rknn_query** 함수를 통해서 rknn_context에 모델 관련 정보가 업데이트 되고 이를 기반으로 하기 함수들을 이용해 입출력 텐서가 세팅 됨

rknn_inputs_set 딥러닝네트워크의 입력 텐서 세팅 함수

rknn_outputs_get 딥러닝네트워크의 출력 텐서 취득 함수

rknn_run 모델이 실제로 수행되는 부분

rknn_msg_send ssd_post Thread 동작 메세지

04. SSD+MobileNet 실습 예제 분석 ssd_post Thread

\rknn_demo\rknn\ssd\ssd_1808\ssd.c

```
int ssd_post(void *flag)
{
    int width;
    int heigh;
    float *predictions;
    float *output_classes;
    struct ssd_group *group;

    while(*(int *)flag) {

        if (!rknn_msg_recv(void **)&predictions, (void **)&output_classes, &width, &heigh,
 (void *)&group))
            recv_count++;
        group = &g_ssd_group[!cur_group];
        // if (group->count > 0 && group->posted > 0)
        // {
        //     if (predictions)
        //         free(predictions);
        //     if (output_classes)
        //         free(output_classes);
        //     printf("throw data\n");
        //     cur_group = !cur_group;
        //     continue;
        // }
        postProcessSSD(predictions, o
        cur_group = !cur_group;
        if (predictions)
            free(predictions);
        if (output_classes)
            free(output_classes);
        ssd_paint_object_msg();
    }
}
```

SSD_run에서 메시지를 받아 쓰래드 동작

SSD 모델 수행 결과 가공 프로세스 함수

PostProcessSSD

rknn_msg_recv 로 ssd_run Thread 의 아웃풋 메시지를 받아 동작하는 프로세서로서 SSD모델의 output을 이용해 객체의 **Detection**정보 (x, y, width, height) 와 **recognition** 정보를 해석한다.

04. SSD+MobileNet 실습 예제 분석 postProcessSSD

\rknn_demo\rknn\ssd\ssd_1808\ssd_post.c

```
int postProcessSSD(float * predictions, float *output_classes, int width,
                   int heigh, struct ssd_group *group)
{
    static int init = -1;
    if (init == -1) {
        int ret = 0;
        printf("loadLabelName\n");
        ret = loadLabelName(LABEL_NALE_TXT_PATH, labels);
        if (ret < 0) {
            return -1;
        }
        // for (int i = 0; i < 91; i++) {
        //     printf("%s\n", labels[i]);
        // }

        printf("loadBoxPriors\n");
        ret = loadBoxPriors(BOX_PRIORS_TXT_PATH, box_priors);
        if (ret < 0) {
            return -1;
        }
        // for (int i = 0; i < 4; i++) {
        //     for (int j = 0; j < 1917; j++) {
        //         printf("%f ", box_priors[i][j]);
        //     }
        //     printf("\n");
        // }
        init = 0;
    }
    int output[2][NUM_RESULTS];
    decodeCenterSizeBoxes(predictions, box_priors);

    int validCount = filterValidResult(out
    if (validCount > 100) {
        printf("validCount too much !!\n");
        return -1;
    }
/* detect nest box */
    nms(validCount, predictions, output);
```

box_priors.txt
coco_labels_list.txt
바운딩박스 정보와
라밸링 정보를 로드

탐색(detect)된 객체의
바운딩박스 매칭

인식(recognition)된 객체의
라밸링 매칭

검출된 바운딩박스 후가공
프로세스

loadLabelName loadBoxPriors
객체의 Detection정보(x,y,width,height)와 recognition 정보를 해석하는 **PostProcessSSD** 도입부에 객체 Label 정보 (**coco_labels_list.txt**)와 Bounding Box 정보 (**box_priors.txt**)로드

decodeCenterSizeBoxes 탐색된 객체의 바운딩박스를 매칭

filterValidResult
검출된 바운딩박스 내 객체를 라밸데이터에 따라 분류

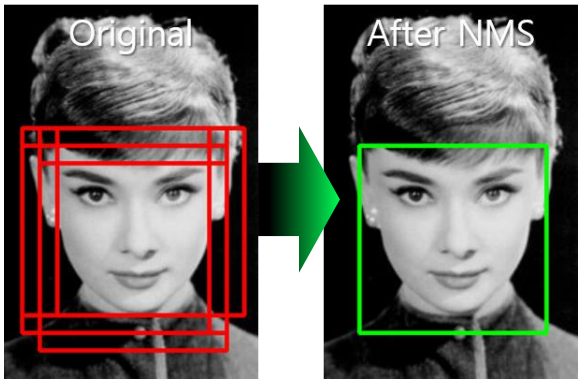
NMS 함수를 이용해서 최종 Bounding Box를 산출하고 이후에는 트랙킹루틴을 수행

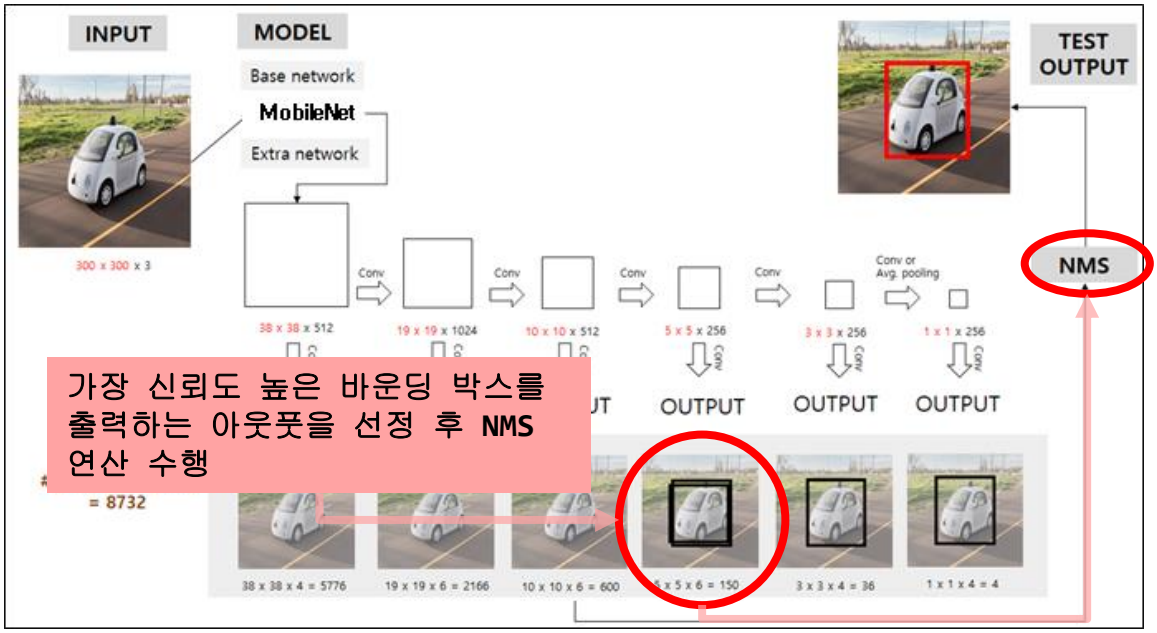
04. SSD+MobileNet 실습 예제 분석 NMS

\rknn_demo\rknn\ssd\ssd_1808\ssd_post.c

```
int nms(int validCount, float* outputLocations, int (*output)[NUM_RESULTS])
{
    for (int i=0; i < validCount; ++i) {
        if (output[0][i] == -1) {
            continue;
        }
        int n = output[0][i];
        for (int j=i + 1; j<validCount; ++j) {
            int m = output[0][j];
            if (m == -1) {
                continue;
            }
            float xmin0 = outputLocations[n*4 + 1];
            float ymin0 = outputLocations[n*4 + 0];
            float xmax0 = outputLocations[n*4 + 3];
            float ymax0 = outputLocations[n*4 + 2];

            float xmin1 = outputLocations[m*4 + 1];
            float ymin1 = outputLocations[m*4 + 0];
            float xmax1 = outputLocations[m*4 + 3];
            float ymax1 = outputLocations[m*4 + 2];

            float iou = CalculateOverlap(xmin0, ymin0, xmax0, ymax0, xmin1, ymin1, xmax1, ymax1);

            if (iou >= NMS_THRESHOLD) {
                output[0][j] = -1;
            }
        }
    }
}
```

NMS(Non-Maximum Suppression)
예측된 바운딩박스 중 가장 신뢰도 높은 박스 하나만 남기고 나머지는 모두 지우는 보편적인 알고리즘.





INFOWORKS

"시대에 맞춘 교육으로 글로벌 인재 육성을 지향합니다."

| 서울특별시 강남구 봉은사로 159, 정글빌딩 5F
| Tel : (02) 569-3346 | Fax : (02) 569-3347
| E-mail : legend4park@info-works.co.kr
| Home-page : WWW.INFO-WORKS.CO.KR