4조 결과 보고서

전기정보공학부 20학번 김도엽 (조장), 전기정보공학부 20학번 김재진, 전기정보공학부 20학번 하태윤

[1차: Yolo를 이용한 Object Detection]

Yolov4 학습

(1) Darknet 학습 준비

Darknet 프레임워크를 다운로드 받고 YoloV4에 맞는 label을 사용하기 위해 제공받은 convert.py 코드를 이용하여 xml label을 txt label로 변환하였다. 제공받은 make_list_cur.py 코드를 이용해 train.txt와 test.txt파일을 생성하였다.

(2) 정확도 향상을 위한 작업

학습 시 정확도 향상을 위해 같은 object인 클래스들끼리 하나로 묶었다(boat1~boat14 -> boat). 이때 변경된 class의 개수에 맞게 yolov4 config 설정도 바꿔주었다. 정확도 향상을 위해, pre-trained model을 yolov4.conv.137로, Yolov4-tiny의 경우 yolov4-tiny.conv.29로 설정하였다.

(2) Yolo V4, Yolo V4-tiny 학습

- ./darknet detector train converted_data/yolov4_drone.data cfg/yolov4_drone.cfg yolov4.conv.137 –dont_show –gpus 1,2,3 명령어를 이용하여 YoloV4 학습을 실행하였다.
- ./darknet detector train converted_data/yolov4-tiny_drone.data cfg/ yolov4-tiny_drone.cfg yolov4-tiny.conv.29 –dont_show –map –gpus 1,2,3 명령어를 이용하여 YoloV4-tiny 학습을 실행하였다.

(3) 학습 결과

V4 (Total Detection Time: 12 Seconds)
for conf_thresh = 0.25, precision = 0.99, recall = 0.99, F1-score = 0.99, average IoU = 87.95 %
IoU threshold = 50 %, mean average precision (mAP@0.50) = 0.984523, or 98.45 %
V4-tiny (Total Detection Time: 4 Seconds)
for conf_thresh = 0.25, precision = 0.98, recall = 0.99, F1-score = 0.99, average IoU = 83.45 %
IoU threshold = 50 %, mean average precision (mAP@0.50) = 0.975940, or 97.59 %

Image Augmentation

-> Robustness을 높이기 위해서, ImgAug library(https://github.com/aleju/imgaug)를 이용하여 각 Class의 전체 이미지 개수가 10000개 근처가 되도록 데이터를 증강하였다.

[2차: Ultra96V2 보드를 이용한 Object Detection]

Ultra96-V2 보드 세팅

이번 대회에서 사용되는 ZYNQ 기반의 프로세서 Ultra96-V2에는 Avnet에 맞는 PYNQ를 SD 카드에 image를 설치하여 기본 운영 환경을 조성하였다. DAC-SDC 대회의 평가 기준 중 하나인 energy measurement를 위해 기존 파일들을 추가적으로 대회에서 제공한 파일로 대체하는 작업을 수행하였다. 제공된 script가 제대로 동작하지 않아, 수동으로 ultra.conf 파일과 BOOT.bin, image.ub를 제공된 파일로 대체하였다.

Object Detection Model 선정 과정

Ultra-96 보드에서 앞선 1차에서 이용되었던 Yolo를 통하여 Object detection을 사례가 있긴 했지만, V2

보드에서 알맞게 활용한 사례는 찾기 어려웠기 때문에 따라서 Yolo를 고집하지 않았다. 결과적으로 오픈 소스로 공개된 rectangular-half-squeezenet라는 모델을 우리 상황에 맞게 integrate하여 진행하기로 결정했다.

모델 활용

(1) 준비

서버에서 GPU를 사용하여 구동하기 위하여 라이브러리 버전 관리와 데이터에 알맞게 작동하게 만들기 위하여 코드를 수정하였다.

(2) 모델 학습

python3 halfsqueezenet_objdetect.py --output="/home/aiproj_4/spooNN/recthalfsqznet/result/" -gpu=0,1 --data=/home/aiproj_4/darknet/data_training --dump2_train1_test0=1

을 통하여 학습을 진행하였다. 이 과정에서 epoch 수에 따라서 오버피팅이 진행되는 것을 확인하여 epoch와 학습률을 조정하여 가장 높은 결과가 나오는 경우로 선택했다. 본 학습을 통하여 params.h와 configs.h 파일을 얻었다.

(3) Ultra 96-V2에서 구동

위 학습을 통해 얻은 결과를 Ultra96-V2에 옮기기 위하여 Vivado HLS를 진행하였다. 원하는 makefile 세팅에서 make를 진행하여 보드에서 이용될 수 있는 형태의 weights.txt 파일을 얻었다. 이후 Ultra96-V2 모델에 알맞은 설정으로 .tcl과 .bit 파일을 지정해주고, 2020년 대회 형식에 알맞도록 코드를 작성하였다. 코드를 보드에서 예시 이미지를 이용해 구동한 결과, 5초 이내의 시간에 알맞은 위치를 잡아내는 결과를 확인했다.