|  |
| --- |
| Edge computing 보드 탑재용 딥러닝기반  객체 인식 패키지 SW 개발    YM-나을텍  2021.08.19 |

목 차

[1. 개요 1](#_Toc79432184)

[2. 엣지 탑재용 딥러닝기반 객체 인식 패키지 SW 2](#_Toc79432185)

[1) 딥러닝기반 객체 탐재 학습 SW 2](#_Toc79432186)

[A. 개요 2](#_Toc79432187)

[B. 야외 객체 탐지/인식을 위한 경량 학습 AI 알고리즘 개발 2](#_Toc79432188)

[C. Darknet 을 이용한 학습 절차 및 방법 5](#_Toc79432189)

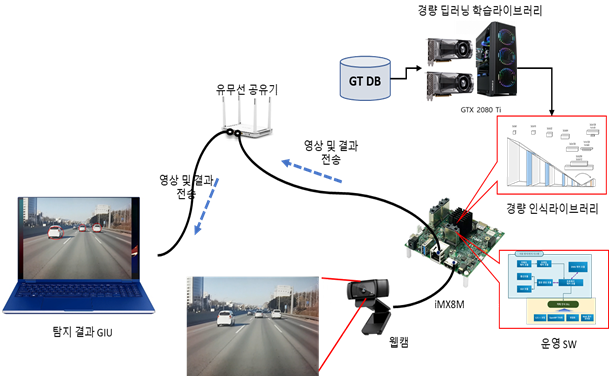
[2) 엣지 탑재 경량 인식 SW 18](#_Toc79432190)

[A. 엣지 시스템 H/W 사양 18](#_Toc79432191)

[B. 엣지 탑재 실시간 객체 탐지 인식 SW 18](#_Toc79432192)

[3. 성능 분석 결과 21](#_Toc79432193)

# 개요

1. 목적 : GPU가 없는 엣지 컴퓨터상에서 고속으로 객체를 검출 하는 기술을 개발
2. 개발 범위

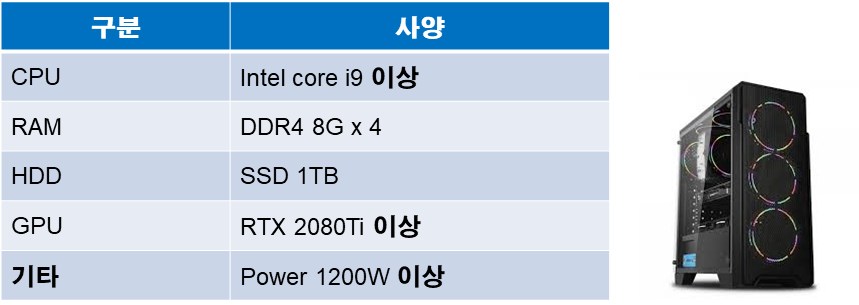
<최종 결과물>

* 딥러닝 기반 객체 탐지 학습 SW : GPU 가 없는 보드 상에서 실시간 응답 속도를 제공할 수 있는 경량 딥러닝 모델 개발
* 엣지 탑재 경량 인식 SW : iMx8M 과 같은 non-GPU 보드상에서 가속라이브러리를 탑재한 C/C++기반 경량 인식 모듈
* 경량 인식 모듈을 탑재한 운영 SW : 리눅스 환경에 구동가능한 C/C++기반에 실시간 인식기 운영 및 연동 SW

# 엣지 탑재용 딥러닝기반 객체 인식 패키지 SW

1. 딥러닝기반 객체 탐재 학습 SW
   1. 개요

본 기술은 딥러닝 기술을 활용하여 GPU가 없는 온디바이스를 활용하여 야외의 객체를 탐지/인식하기 위한 딥러닝 모델을 개발하는 학습 라이브러리이다. 객체 탐지/인식을 위한 AI 학습용 H/W 시스템은 GTX 2080 Ti GPU 보드를 탐재한 H/W를 기반으로 학습을 수행한다. 적절한 학습 수행을 기본적인 H/W 구성 및 스펙은 아래와 같다.

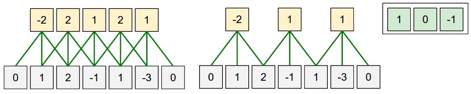


<객체 인식용 모델 개발을 위한 H/W 구성 및 스펙>

* 1. 야외 객체 탐지/인식을 위한 경량 학습 AI 알고리즘 개발

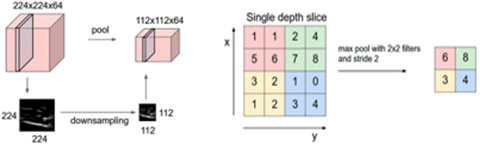
본 절에서는 경량 학습 AI 알고리즘에 대해서 설명한다. 본 알고리즘은 컨볼루션 신경망은 층으로 구성되어 있고 그 층은 가로, 세로, 깊이로 되어있다. 각 층의 종류마다 하는 역할이 다르게 구성 되어있다. 주로 사용되는 층은 convolution, pooling, fully connected, ReLu 층으로 구성됨.

* Convolution Layer: 뉴런들의 연결이 fully-connected가 아니라 지역적인 연결특성을 가지도록 만들며 뉴런들을 3차원으로 위치시킬 수 있는 방법이다. 뉴런에게 들어오는 입력 파라미터들은 입력 이미지보다 작은 필터단위로 각 뉴런들에게 입력된다. 그리고 이 파라미터들을 이전의 신경망처럼 학습한다. 즉, 다른 필터와의 연결은 없으며 입력의 깊이가 연결



<Convolution Layer>

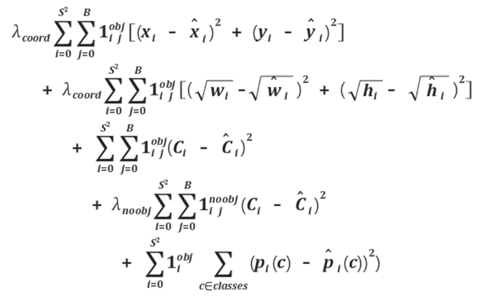
* Polling Layer : 입력을 현저하게 줄이는 역할을 한다. 그만큼 계산해야 하는 뉴런의 파라미터들도 줄어들며 그 결과로 과적 합(overfitting) 문제를 부분적으로 해결할 수 있다. 이 층에서는 각 댑스마다 각각 독립적으로 연산한다. 보통 최대값 연산을 하는 max pooling을 사용하며 2x2 필터로 2 stride로 설정해 사용함.



<Pooling층의 모델링>

신경망 알고리즘의 학습은 Loss function을 근간으로 개발되었다. 다음은 Loss function 수식에 대한 설명이다.

* DRW00000a303cc0 은 i번째 cell에 object가 나타났다는 것을 의미함.
* DRW00000a303cca 는 cell i의 j번째 bounding box predictor는 해당 예측을 확정함.
* loss function 은 object 가 해당 grid cell에 존재할 때만, classification error를 반영



<CNN 학습 알고리즘의 Loss Function>

* 1. Darknet 을 이용한 학습 절차 및 방법
     1. Darknet

<https://pjreddie.com/darknet/yolo/>

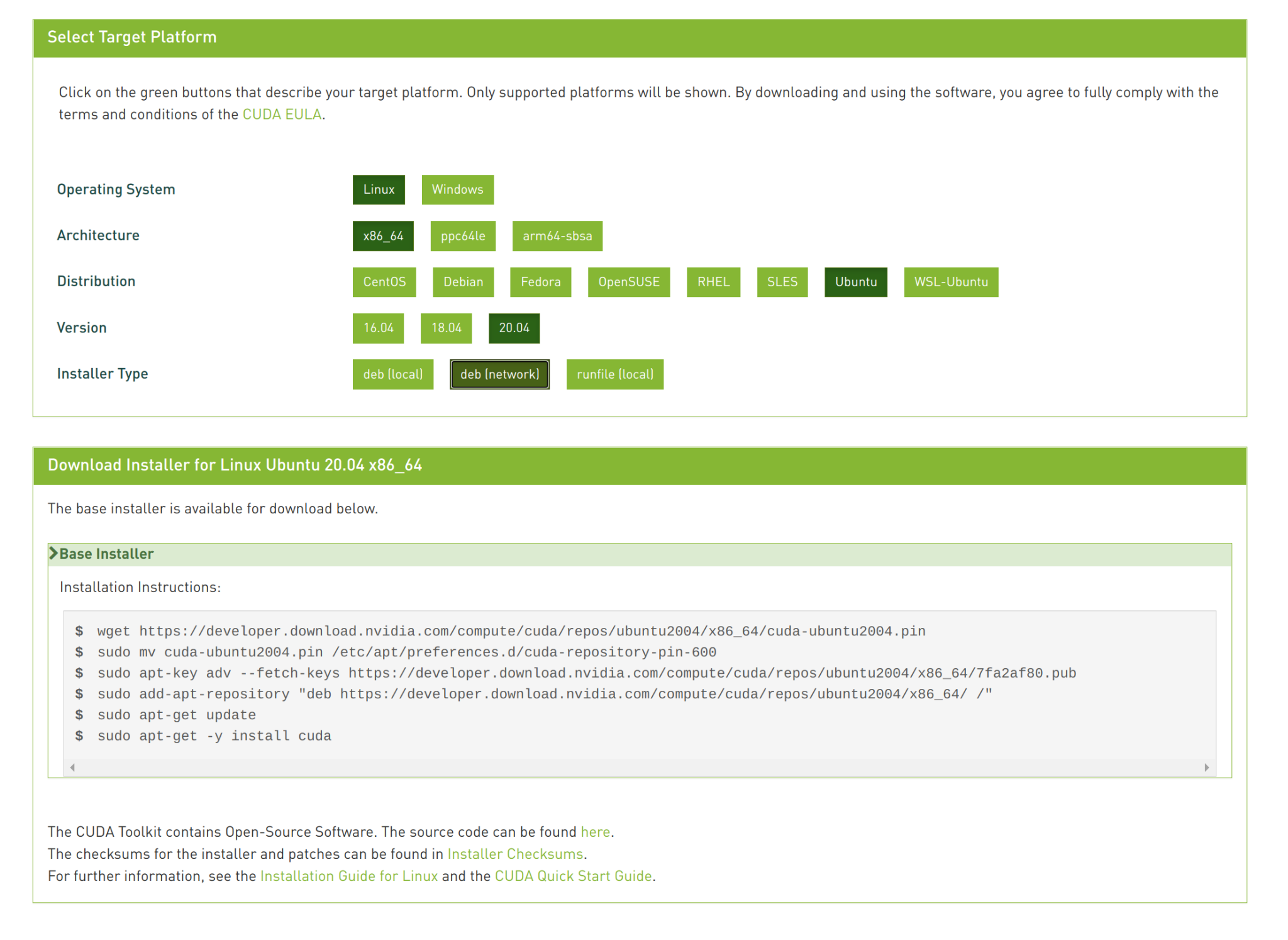
* + 1. Requirements
* OS

Linux

* CUDA

아래의 사이트에서 환경에 맞는 옵션 선택 후 설치 방법에 따라 설치

<https://developer.nvidia.com/cuda-downloads>



* + 1. Build
* Clone

|  |
| --- |
| git clone https://github.com/pjreddie/darknet |

* Makefile 수정

|  |
| --- |
| **GPU=1**  CUDNN=0  OPENCV=0  OPENMP=0  DEBUG=0  **ARCH= -gencode arch=compute\_30,code=sm\_30 \**  **-gencode arch=compute\_35,code=sm\_35 \**  **-gencode arch=compute\_50,code=[sm\_50,compute\_50] \**  **-gencode arch=compute\_52,code=[sm\_52,compute\_52]**  ... |

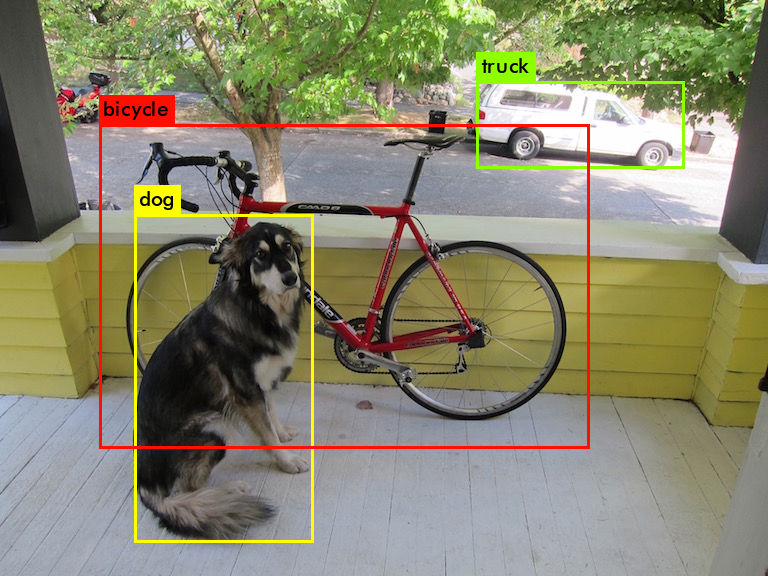
ARCH 변수는 nvcc 컴파일러에 사용되는 값으로 GPU 모델과 cuda 버전에 따라 다르게 설정해야 최대의 성능 사용이 가능 하므로 보유 모델에 맞게 변경 필요 (ex. “rtx 3090 nvcc gencode” 구글링으로 확인가능)

* Build

|  |
| --- |
| make |

* + 1. Test

|  |
| --- |
| # download pre-trained weight file  wget https://pjreddie.com/media/files/yolov3.weights  # run test with dog.jpg  ./darknet detect cfg/yolov3.cfg yolov3.weights data/dog.jpg |



|  |
| --- |
| # run test with kite.jpg  ./darknet detect cfg/yolov3.cfg yolov3.weights data/kite.jpg |



* + 1. Training
* Data 구성

[Data 구성 방법은 ms-coco 데이터를 기준으로 작성됨]

데이터는 아래의 구조와 같이 images, labels의 두개의 폴더의 하위 폴더 및 이름이 1:1로 대응하는 이미지파일과 텍스트 파일이 필요하다.

|  |
| --- |
| ├── images  │ ├── train2014  │ │ ├── COCO\_train2014\_000000000009.jpg  │ │ ├── COCO\_train2014\_000000000025.jpg  │ │ └── ...  │ └── val2014  │ ├── COCO\_val2014\_000000000042.jpg  │ ├── COCO\_val2014\_000000000073.jpg  │ └── ...  ├── labels  │ ├── train2014  │ │ ├── COCO\_train2014\_000000000009.txt  │ │ ├── COCO\_train2014\_000000000025.txt  │ │ └── ...  │ └── val2014  │ ├── COCO\_val2014\_000000000042.txt  │ ├── COCO\_val2014\_000000000073.txt  │ └── ...  ├── train.txt  └── valid.txt |

#### Labels

Label 파일에는 object 정보가 저장되며 object 정보는 한 줄에 하나씩 아래와 같은 양식으로 구성된다.

|  |
| --- |
| <object-class> <x> <y> <width> <height> |

{image\_name}.txtlabels 폴더 하위에 존재하는 텍스트 파일에는 해당 이미지에 대해서 학습할 object정보가 저장된다. 해당 데이터는 다음과 같은 규칙으로 하나의 파일에 0개 이상의 object 정보가 포함되며, box 정보는 이미지를 0~1로 정규화 했을때의 좌표이다.

|  |
| --- |
| 45 0.479492 0.688771 0.955609 0.595500  45 0.736516 0.247188 0.498875 0.476417  50 0.637063 0.732938 0.494125 0.510583  45 0.339438 0.418896 0.678875 0.781500  49 0.646836 0.132552 0.118047 0.096937  49 0.773148 0.129802 0.090734 0.097229  49 0.668297 0.226906 0.131281 0.146896  49 0.642859 0.079219 0.148063 0.148062 |

#### train.txt

train.txt 파일에는 아래와 같이 학습시 사용할 이미지의 절대경로 리스트

|  |
| --- |
| /home/user/data/coco/images/train2014/COCO\_train2014\_000000000009.jpg  /home/user/data/coco/images/train2014/COCO\_train2014\_000000000025.jpg  ... |

#### valid.txt

valid.txt 파일에는 아래와 같이 validation에 사용할 이미지의 절대경로 리스트가 저장 되어야 한다.

|  |
| --- |
| /home/user/data/coco/images/val2014/COCO\_val2014\_000000000164.jpg  /home/user/data/coco/images/val2014/COCO\_val2014\_000000000192.jpg  ... |

#### .names

.names 파일에는 학습하고자 하는 class 들의 이름을 한줄에 하나씩 저장

|  |
| --- |
| person  bicycle  car  motorbike  ... |

#### .data

.data 파일에는 위의 정보들을 하나의 파일에 요약하여 저장되며 파일은 아래와 같이 구성된다.

|  |
| --- |
| classes= 80  train = <path-to-coco>/train.txt  valid = <path-to-coco>/valid.txt  names = data/coco.names  backup = backup |

### Model

testworks-v1-c3-256.cfg

|  |
| --- |
| [net]  ######## Testing  # batch=1  # subdivisions=1  ######## Training  batch=64  subdivisions=1  width=256  height=256  channels=3  momentum=0.9  decay=0.0005  angle=0  saturation = 1.5  exposure = 1.5  hue=.1  learning\_rate=0.001  policy=steps  burn\_in=1000  max\_batches = 80200  steps=64000,72000  scales=.1,.1  [convolutional]  batch\_normalize=1  filters=8  size=3  stride=1  pad=1  activation=leaky  [maxpool]  size=2  stride=2  [convolutional]  batch\_normalize=1  filters=16  size=3  stride=1  pad=1  activation=leaky  [maxpool]  size=2  stride=2  [convolutional]  batch\_normalize=1  filters=32  size=3  stride=1  pad=1  activation=leaky  [maxpool]  size=2  stride=2  [convolutional]  batch\_normalize=1  filters=64  size=3  stride=1  pad=1  activation=leaky  [maxpool]  size=2  stride=2  [convolutional]  batch\_normalize=1  filters=128  size=3  stride=1  pad=1  activation=leaky  [maxpool]  size=2  stride=2  [convolutional]  batch\_normalize=1  filters=256  size=3  stride=1  pad=1  activation=leaky  [maxpool]  size=2  stride=1  [convolutional]  batch\_normalize=1  filters=512  size=3  stride=1  pad=1  activation=leaky  ###########  [convolutional]  batch\_normalize=1  filters=128  size=1  stride=1  pad=1  activation=leaky  [convolutional]  batch\_normalize=1  filters=64  size=1  stride=1  pad=1  activation=leaky  [upsample]  stride=2  [route]  layers = -1, 8  [convolutional]  batch\_normalize=1  filters=128  size=3  stride=1  pad=1  activation=leaky  [convolutional]  size=1  stride=1  pad=1  filters=24  activation=linear  [yolo]  mask = 0,1,2  anchors = 21, 20, 50, 47, 115, 107  classes=3  num=3  jitter=.3  ignore\_thresh = .5  truth\_thresh = 1  random=1 |

batch: 한번의 학습에 사용할 image의 수로 GPU 메모리에 맞게 설정 가능

width/height: 모델의 입력 크기로 일반적으로 같은 값을 사용하며 32의 배수로 설정

max\_batches: 트레이닝 횟수를 의미하며 batch 처리를 몇 번 반복할 지를 의미, 일반적으로 200 epochs(모든 이미지가 200회 학습될) 정도의 반복이 필요함.   
(max\_batches = num\_images / batch \* 200)

learning\_rate: 시작 learning rate

policy: learning rate policy, step decay 방식을 사용하도록 설정 됨

steps: step decay가 적용 될 batch 번호, yolov3 에서는 80%, 90%에 해당하는 batch로 설정 함

scales: step decay가 적용 될 때 learning rate에 반영 될 배수

yolo/anchors: anchor에서 사용될 (width, height) 값의 조합, {w1, h1, w2, h2, w3, h3} 와 같이 anchor의 개수 만큼 정의가 필요하다. 해당 값은 이미지상의 모든 object의 가로,세로 크기에 대하여 k-means 연산을 수행해 얻어진다(k = anchor의 개수이며 Yolo v3 논문 참조). 값의 범위는 network input 사이즈를 기준으로 한다.

yolo바로 위 convolution/filter size 설정:

filters = 3 \* (classes + 5)

### Training

아래의 명령어로 모델 학습

|  |
| --- |
| ./darknet detector train <.data> <.cfg> <.weights> -clear |

<.weights>파일은 정확히 같은 모델의 다른 데이터셋에 사용할 수 있으면 더 좋지만, 모델의 구조가 다르더라도 기존에 학습된 weights를 입력해 줌으로 weights 초기화 문제를 해결하고 빠른 수렴에 도움을 준다. 학습이 진행되면서 중간/최종 학습된 weights 파일이 .data 파일에서 설정한 backup 폴더에 저장된다. 모델이 학습되는 도중 특정 시점 마다 .backup 파일에 저장되며, 학습이 중간에 취소 된 경우 이 파일을 pre-trained-model로 설정하면 마지막 저장된 시점부터 다시 학습한다.

|  |
| --- |
| ./darknet detector train <.data> <.cfg> <.backup> |

* + 1. Etc.

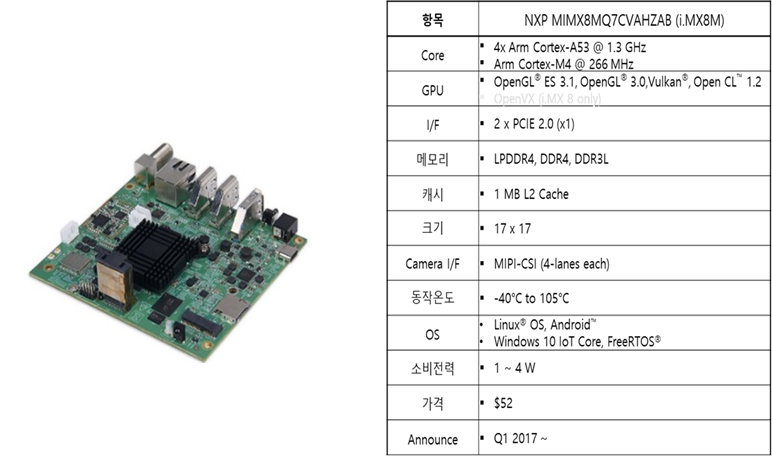
### Network resize

Darknet에서는 .cfg 파일에 random=1 로 설정되어 있을 경우 학습도중 10 batch 마다 network resize를 특정 값들 중 임의의 값으로 수행한다. 기본 소스코드의 설정은 resize 되는 입력 사이즈는 32의 배수인 {320, 352, ..., 608} 중 하나로 결정된다(Yolo v2 논문 참조). 이 값들은 416x416 입력에 대한 설정이기 때문에 256 과 같은 네트워크의 값에는 {192, 224, …, 384} 정도의 값으로 설정하는 것이 도움이 되는데 이를 위해서는 아래의 소스코드 수정이 필요하다.

|  |
| --- |
| # darknet/examples/detectors.c  void train\_detector(...){  ...  while(get\_current\_batch(net) < net->max\_batches){  if(l.random && count++%10 == 0){  printf("Resizing\n");  **int dim = (rand() % 10 + 10) \* 32;**  ... |

1. 엣지 탑재 경량 인식 SW
   1. 엣지 시스템 H/W 사양

경량 딥러닝 인퍼런스 알고리즘 탑재를 위한 엣지 시스템의 H/W 스펙은 아래와 같다. Arm 기반의 CPU를 가지는 저비용 저발열의 엣지 보드를 대상으로 경량 인퍼런스 시스템을 탑재한다.

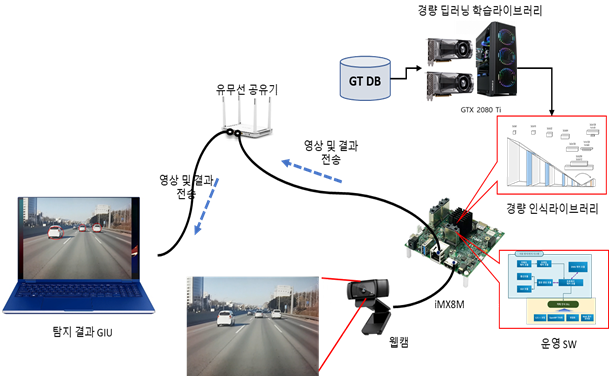


**<엣지 컴퓨팅 H/W 스펙>**

* 1. 엣지 탑재 실시간 객체 탐지 인식 SW

엣지 컴퓨터에 탑재된 실시간 객체 인식 SW 의 물리적 구조는 아래와 같다. 개발된 시스템은 iMx8M 보드와 유무선 공유기 그리고 영상 입력장치, 웹캠으로 구성된다.

영상 입력 장치나 웹캠으로 입력되는 영상에 대해서 iMx8M 보드에 탑재된 경량 인퍼런스 엔진을 통해서 객체의 종류를 인식하고 인식된 결과는 다시 영상 입력장치에 전달하는 구조로 설계되어있다.

<엣지 컴퓨팅 탑재 경량 딥러닝 시스템 구조>

1. 경량 딥러닝 모델 및 인식기 가속화

GPU 가 없는 보드상에서 실시간 딥러닝 인식을 위해서는 다양한 방법의 경량화된 딥러닝 엔진 개발이 필요하다. 이에 기 보유하고 있는 경량 딥러닝 엔진을 이용하여 CPU 상에서 300ms 이하의 속도로 객체를 탐지/인식하는 엔진을 탐재하고 이를 다양한 종류의 자동차를 탐지/인식 할 수 있도록 경량 딥러닝 모델을 개발한다. 다음은 CPU 만을 갖는 온디바이스 보드에서 실시간 성능을 갖는 딥러닝 모델이다**.**

1. OpenMP를 이용한 CPU 가속 연산

OpneMP는 C/C++ 에서 병렬프로그램을 가능하게 해주는 API로써, 내부적으로 Multi-Thread 및 공유메모리를 사용하게 해주며, 대부분의 컴파일러에서 구현되어 있어서 다른 병렬 프로그래밍 방식에 비해서 효과적으로 활용할 수 있다. 다음은 OpenMP의 개략적인 특징이다.

● 분산 메모리에서는 사용할 수 없다

● 구현은 컴파일러마다 차이가 있을 수 있다

● 최상의 공유메모리 사용 패턴을 보장하지 않는다.

● Data Dependency, Data Race, Deadlock 검사는 프로그래머가 해야한다.

● 컴파일러가 알아서 프로그램을 병렬로 변환해 주지 않는다. 어느 부분을 어떻게 병렬화 할지를 프로그래머가 지정해 주어야 한다.

● 입출력의 동기화는 프로그래머의 몫이다.

위와 같은 특징을 고려하여 엣지 탑재 경량 딥러닝 엔진에는 다중 for 문에서 openMP 가속을 적용하여 실시간 적용을 위한 추가적인 속도 향상이 가능하도록 시스템을 개발하였다. 아래는 OpenMP를 적용한 인퍼런스 엔진의 예이다.

|  |
| --- |
| #ifdef YOLO\_OMP\_IMG\_CONVERSION  #pragma omp parallel for schedule(static) num\_threads(2)  #endif  for (int row = 0; row < pDst->height; ++row) {  for (int nCh = 0; nCh < pDst->channel; ++nCh) {  for (int col = 0; col < pDst->width; ++col) {  im.data[nCh \* pDst->width \* pDst->height + row \* pDst->width + col]  = ((unsigned char)pSrc->pData[(row + clipPosRow) \* pSrc->widthStep + (col + clipPosCol) \* pDst->channel + nCh]) / 255.0f;  }  }  }  #else |

1. 성능 분석 결과

◦ 성능 지표 정의

- 다양한 야외 환경에서 수집된 영상에 대해서 Ground Truth 정보를 수작업으로 기록하고 이를 대상으로 AI 솔루션이 객체 탐지/인식 결과와 비교하여 성능을 검증함

◦ 평가방법

- AI 솔루션의 객체 탐지/인식 결과와 Ground Truth 간의 검출 및 인식률 측정

◦ 결과

- 5개의 클래스(boxcar, car, person, remicon, truck)에 대해서 F1-score 로 0.81로 인식됨

