Install and Test Darknet on Raspberry Pi

Aug. 2019

Ando Ki, Ph.D.
adki@future-ds.com

Contents

- Prerequisites
- What is Darknet
- Building Darknet
- Testing Darknet using Tiny-YOLO
- Darknet profiling
- How to deal with ‘gemm_nn‘
- Darknet using OpenMP
- Running Tiny-YOLO with USB-CAM
- Running Tiny-YOLO with video stream

Prerequisites

- OpenCV
- OpenMP
- OpenBLAS

What is Darknet

- Darknet is an open source neural network framework written in C and CUDA (C ompute Unified Device Architecture) supporting CPU (Central Processing Unit) and GPU (Graphical Processing Unit) computation.
  - ▶ Site: https://pjreddie.com/darknet/
  - ▶ GitHub : https://github.com/pjreddie/darknet
    - ➲ This version may cause error on Rasbperry Pi while running.

- Alexey's version
  - ▶ https://github.com/AlexeyAB/darknet

"Darknet: Open Source Neural Networks in C", Joseph Redmon, http://pjreddie.com/darknet, 2013-2016.

Building Darknet

- Visit
  - https://github.com/AlexeyAB/darknet
- Download
  - make a directory
  - $ mkdir work && cd work
  - $ git clone https://github.com/AlexeyAB/darknet.git
  - $ mv darknet darknet-alexey
- Modify 'Makefile'
  - cd darknet-alexey
  - $ vi Makefile
    - set 1 for OpenCV if you installed it.
- Compile
  - $ make
- At last
  - '**darknet**': executable

```
GPU=0
CUDNN=0
CUDNN_HALF=0
OPENCV=1
AVX=0
OPENMP=0
LIBSO=0
ZED_CAMERA=0
```

- Nvidia CUDA related
  - GPU, CUDNN, CUDNN_HALF
- x86 Vector related
  - AVX
- Multi-core/computer related
  - OpenMP
- Shared library
  - LIBSO
- 3D camera
  - ZED_CAMERA

Darknet usage

- $ ./darknet function [function_arguments]

- functions
  - look 'darknet/examples/darknet.c' file and its related C files.

  - detect [cfg_file] [weights_file] [options]
  - detector [train/test/valid] [data_cfg] [cfg_file] [weights_file] [options]
  - yolo [train/test/valid] [cfg_file] [weights_file] [options]
  - cifar [train/test/valid] [cfg_file] [weights_file] [options]

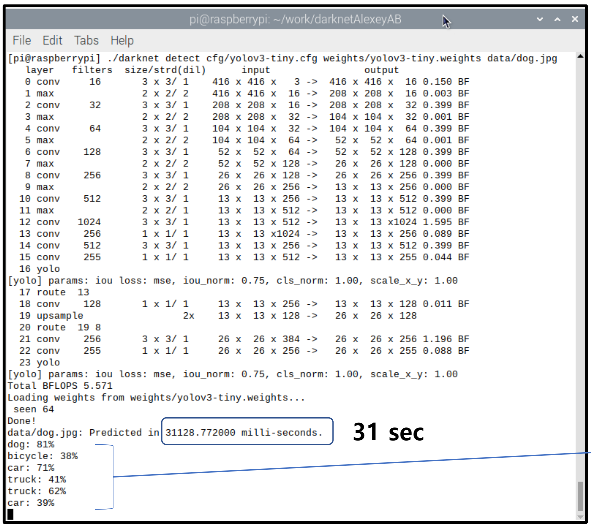
```
$ ./darknet detect cfg/yolov3.cfg weights/yolov3.weights data/dog.jpg
$ ./darknet detector test cfg/coco.data cfg/yolov3.cfg weights/yolov3.weights data/dog.jpg

$ ./darknet detector test cfg/voc.data cfg/yolo.cfg weights/yolo.weights data/dog.jpg
```

Testing Darknet using Tiny-YOLO (1/2)

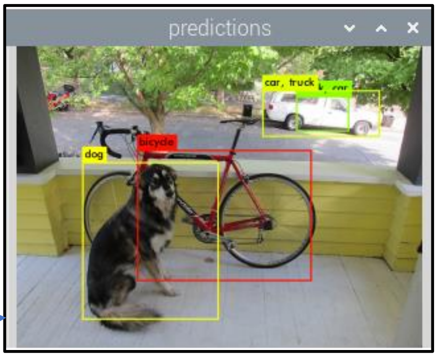
- Download weight file
  - ▶ $ cd ~/work/darknet-alexey
  - ▶ $ mkdir weights && cd weights
  - ▶ $ wget https://pjreddie.com/media/files/yolov3-tiny.weights
- Run Tiny-YOLO
  - ▶ $ cd ~/work/darknet-alexey
  - ▶ $ ./darknet detect cfg/yolov3-tiny.cfg weights/yolov3-tiny.weights data/dog.jpg

Testing Darknet using Tiny-YOLO (2/2)



```
[pi@raspberrypi] ./darknet detect cfg/yolov3-tiny.cfg weights/yolov3-tiny.weights data/dog.jpg
   layer   filters  size/strd(dil)      input                output
   0 conv     16       3 x 3/ 1    416 x 416 x   3 ->  416 x 416 x  16 0.150 BF
   1 max               2 x 2/ 2    416 x 416 x  16 ->  208 x 208 x  16 0.003 BF
   2 conv     32       3 x 3/ 1    208 x 208 x  16 ->  208 x 208 x  32 0.399 BF
   3 max               2 x 2/ 2    208 x 208 x  32 ->  104 x 104 x  32 0.001 BF
   4 conv     64       3 x 3/ 1    104 x 104 x  32 ->  104 x 104 x  64 0.399 BF
   5 max               2 x 2/ 2    104 x 104 x  64 ->   52 x  52 x  64 0.001 BF
   6 conv    128       3 x 3/ 1     52 x  52 x  64 ->   52 x  52 x 128 0.399 BF
   7 max               2 x 2/ 2     52 x  52 x 128 ->   26 x  26 x 128 0.000 BF
   8 conv    256       3 x 3/ 1     26 x  26 x 128 ->   26 x  26 x 256 0.399 BF
   9 max               2 x 2/ 2     26 x  26 x 256 ->   13 x  13 x 256 0.000 BF
  10 conv    512       3 x 3/ 1     13 x  13 x 256 ->   13 x  13 x 512 0.399 BF
  11 max               2 x 2/ 1     13 x  13 x 512 ->   13 x  13 x 512 0.000 BF
  12 conv   1024       3 x 3/ 1     13 x  13 x 512 ->   13 x  13 x1024 1.595 BF
  13 conv    256       1 x 1/ 1     13 x  13 x1024 ->   13 x  13 x 256 0.089 BF
  14 conv    512       3 x 3/ 1     13 x  13 x 256 ->   13 x  13 x 512 0.399 BF
  15 conv    255       1 x 1/ 1     13 x  13 x 512 ->   13 x  13 x 255 0.044 BF
  16 yolo
[yolo] params: iou loss: mse, iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00
  17 route  13
  18 conv    128       1 x 1/ 1     13 x  13 x 256 ->   13 x  13 x 128 0.011 BF
  19 upsample                 2x    13 x  13 x 128 ->   26 x  26 x 128
  20 route  19 8
  21 conv    256       3 x 3/ 1     26 x  26 x 384 ->   26 x  26 x 256 1.196 BF
  22 conv    255       1 x 1/ 1     26 x  26 x 256 ->   26 x  26 x 255 0.088 BF
  23 yolo
[yolo] params: iou loss: mse, iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00
Total BFLOPS 5.571
Loading weights from weights/yolov3-tiny.weights...
 seen 64
Done!
data/dog.jpg: Predicted in 31128.772000 milli-seconds.
dog: 81%
bicycle: 38%
car: 71%
truck: 41%
truck: 62%
car: 39%
```

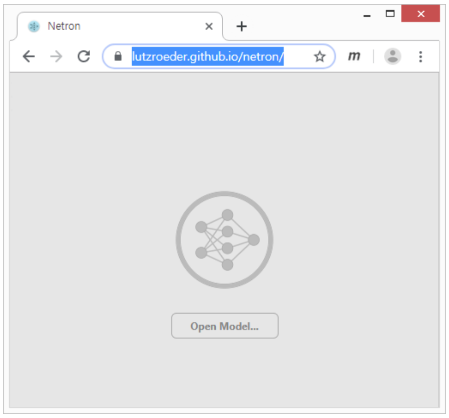
31 sec

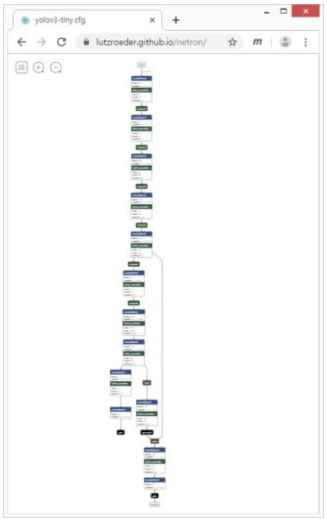


Type ‘q’ on the picture in order to quit.

Network visualizer

- https://github.com/lutzroeder/netron
- https://lutzroeder.github.io/netron/





Contents

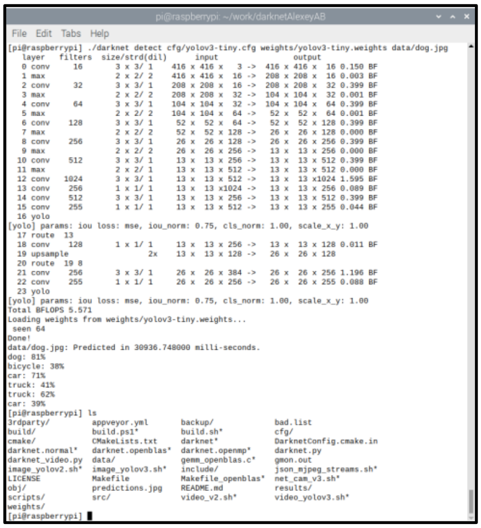
- Prerequisites
- What is Darknet
- Building Darknet
- Testing Darknet using Tiny-YOLO
- Darknet profiling
- How to deal with ‘gemm_nn‘
- Darknet using OpenMP
- Running Tiny-YOLO with USB-CAM
- Running Tiny-YOLO with video stream

Darknet profiling (1/2)

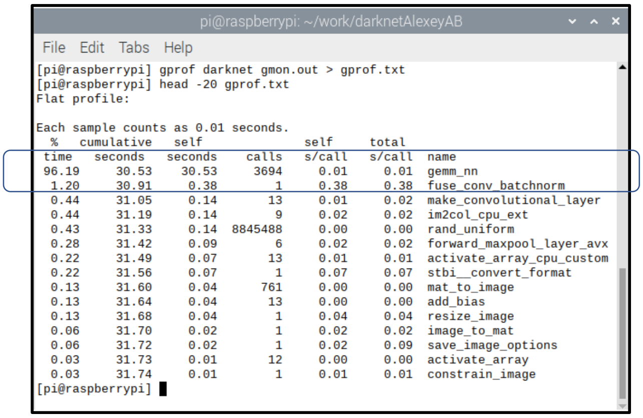
- ‘gprof’ 사용
  - 컴파일 단계에서 ‘-pg’ 선택자 사용
  - 프로그램 수행 결과로 ‘gmon.out’ 파일 생성
  - ‘gprof’ 프로그램으로 분석
- 1) ‘Makefile’의 ‘CFLAGS’에 ‘-pg’ 추가
  - CFLGAS+=-pg
- 2) ‘make’ 실행
  - $ make clean && make GPROF=1
- 3) run
  - $ ./darknet detect cfg/yolov3-tiny.cfg weights/yolov3-tiny.weights data/dog.jpg
- 4) ‘gprof’ 실행
  - $ gprof darknet gmon.out > gprof.txt
- 5) ‘gprof.txt’ 파일 검토
  - $ head -20 gprof.txt

```
GPU=0
CUDNN=0
CUDNN_HALF=0
OPENCV=1
AVX=0
OPENMP=0
LIBSO=0
ZED_CAMERA=0
GPROF=0
....
ifeq ($(GPROF), 1)
CFLAGS+=-pg
endif
...
```

Darknet profiling (2/2)



```
[pi@raspberrypi] ./darknet detect cfg/yolov3-tiny.cfg weights/yolov3-tiny.weights data/dog.jpg
   layer   filters  size/strd(dil)      input                output
   0 conv     16       3 x 3/ 1    416 x 416 x   3 ->  416 x 416 x  16 0.150 BF
   1 max               2 x 2/ 2    416 x 416 x  16 ->  208 x 208 x  16 0.003 BF
   2 conv     32       3 x 3/ 1    208 x 208 x  16 ->  208 x 208 x  32 0.399 BF
   3 max               2 x 2/ 2    208 x 208 x  32 ->  104 x 104 x  32 0.001 BF
   4 conv     64       3 x 3/ 1    104 x 104 x  32 ->  104 x 104 x  64 0.399 BF
   5 max               2 x 2/ 2    104 x 104 x  64 ->   52 x  52 x  64 0.001 BF
   6 conv    128       3 x 3/ 1     52 x  52 x  64 ->   52 x  52 x 128 0.399 BF
   7 max               2 x 2/ 2     52 x  52 x 128 ->   26 x  26 x 128 0.000 BF
   8 conv    256       3 x 3/ 1     26 x  26 x 128 ->   26 x  26 x 256 0.399 BF
   9 max               2 x 2/ 2     26 x  26 x 256 ->   13 x  13 x 256 0.000 BF
  10 conv    512       3 x 3/ 1     13 x  13 x 256 ->   13 x  13 x 512 0.399 BF
  11 max               2 x 2/ 1     13 x  13 x 512 ->   13 x  13 x 512 0.000 BF
  12 conv   1024       3 x 3/ 1     13 x  13 x 512 ->   13 x  13 x1024 1.595 BF
  13 conv    256       1 x 1/ 1     13 x  13 x1024 ->   13 x  13 x 256 0.089 BF
  14 conv    512       3 x 3/ 1     13 x  13 x 256 ->   13 x  13 x 512 0.399 BF
  15 conv    255       1 x 1/ 1     13 x  13 x 512 ->   13 x  13 x 255 0.044 BF
  16 yolo
[yolo] params: iou loss: mse, iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00
  17 route  13
  18 conv    128       1 x 1/ 1     13 x  13 x 256 ->   13 x  13 x 128 0.011 BF
  19 upsample                 2x    13 x  13 x 128 ->   26 x  26 x 128
  20 route  19 8
  21 conv    256       3 x 3/ 1     26 x  26 x 384 ->   26 x  26 x 256 1.196 BF
  22 conv    255       1 x 1/ 1     26 x  26 x 256 ->   26 x  26 x 255 0.088 BF
  23 yolo
[yolo] params: iou loss: mse, iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00
Total BFLOPS 5.571
Loading weights from weights/yolov3-tiny.weights...
 seen 64
Done!
data/dog.jpg: Predicted in 30936.748000 milli-seconds.
dog: 81%
bicycle: 38%
car: 71%
truck: 41%
truck: 62%
car: 39%
[pi@raspberrypi] ls
3rdparty/         appveyor.yml      backup/             bad.list
build/            build.ps1*        build.sh*           cfg/
cmake/            CMakeLists.txt    darknet*            DarknetConfig.cmake.in
darknet.normal*   darknet.openblas* darknet.openmp*     darknet.py
darknet_video.py  data/             gemm_openblas.c*    gmon.out
image_yolov2.sh*  image_yolov3.sh*  include/            json_mjpeg_streams.sh*
LICENSE           Makefile          Makefile_openblas*  net_cam_v3.sh*
obj/              predictions.jpg   README.md           results/
scripts/          src/              video_v2.sh*        video_yolov3.sh*
weights/
[pi@raspberrypi] █
```



```
[pi@raspberrypi] gprof darknet gmon.out > gprof.txt
[pi@raspberrypi] head -20 gprof.txt
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 96.19     30.53    30.53     3694     0.01     0.01  gemm_nn
  1.20     30.91     0.38        1     0.38     0.38  fuse_conv_batchnorm
  0.44     31.05     0.14       13     0.01     0.02  make_convolutional_layer
  0.44     31.19     0.14        9     0.02     0.02  im2col_cpu_ext
  0.43     31.33     0.14  8845488     0.00     0.00  rand_uniform
  0.28     31.42     0.09        6     0.02     0.02  forward_maxpool_layer_avx
  0.22     31.49     0.07       13     0.01     0.01  activate_array_cpu_custom
  0.22     31.56     0.07        1     0.07     0.07  stbi__convert_format
  0.13     31.60     0.04      761     0.00     0.00  mat_to_image
  0.13     31.64     0.04       13     0.00     0.00  add_bias
  0.13     31.68     0.04        1     0.04     0.04  resize_image
  0.06     31.70     0.02        1     0.02     0.02  image_to_mat
  0.06     31.72     0.02        1     0.02     0.09  save_image_options
  0.03     31.73     0.01       12     0.00     0.00  activate_array
  0.03     31.74     0.01        1     0.01     0.01  constrain_image
[pi@raspberrypi] █
```

How to deal with ‘gemm_nn’(1/3)

src/gemm.c

```
#ifdef OPENBLAS
#include <cblas.h>

void gemm( int    TA
         , int    TB
         , int    M
         , int    N
         , int    K
         , float  ALPHA
         , float *A
         , int    lda
         , float *B
         , int    ldb
         , float  BETA
         , float *C
         , int    ldc)
{
    cblas_sgemm(CblasRowMajor
                , (TA==0) ? CblasNoTrans : CblasTrans
                , (TB==0) ? CblasNoTrans : CblasTrans
                ,M               // OPENBLAS_CONST blasint M
                ,N               // OPENBLAS_CONST blasint N
                ,K               // OPENBLAS_CONST blasint K
                ,ALPHA           // OPENBLAS_CONST float alpha
                ,A               // OPENBLAS_CONST float *A
                ,lda             // OPENBLAS_CONST blasint lda
                ,B               // OPENBLAS_CONST float *B
                ,ldb             // OPENBLAS_CONST blasint ldb
                ,BETA            // OPENBLAS_CONST float beta
                ,C               // float *C
                ,ldc             // OPENBLAS_CONST blasint ldc
                );
}
```

```
#else
void gemm(int TA, int TB, int M, int N, int K, float ALPHA,
        float *A, int lda,
        float *B, int ldb,
        float BETA,
        float *C, int ldc)
{
    gemm_cpu( TA,  TB,  M, N, K, ALPHA,A,lda, B, ldb,BETA,C,ldc);
}
#endif
```

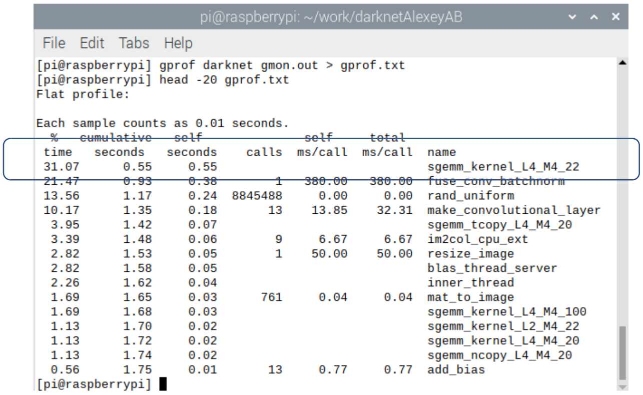
- Use optimized GEMM in ‘src/gemm.c’
- Change ‘Makefile’

Makefile

```
…
OPENBLAS=0
….
ifeq ($(OPENBLAS), 1)
COMMON+=-DOPENBLAS
CFLAGS+=-DOPENBLAS –I/opt/OpenBLAS/include
LDFLAGS+=/opt/OpenBLAS/lib/libopenblas.a
endif
…
```

How to deal with ‘gemm_nn’(2/3)

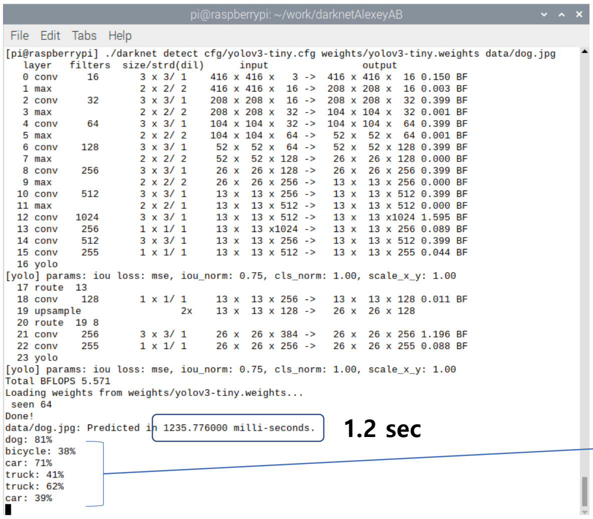
- Install OpenBLAS
  - $ sudo apt-get install libopenblas-dev
- $ make clean && make OPENBLAS=1 \
- GPROF=1
- $ ./darknet detect cfg/yolov3-tiny.cfg \
- weights/yolov3-tiny.weights data/dog.jpg
- $ gprof darknet gmon.out > gprof.txt
- $ head -20 gprof.txt



```
[pi@raspberrypi] gprof darknet gmon.out > gprof.txt
[pi@raspberrypi] head -20 gprof.txt
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 31.07      0.55     0.55                             sgemm_kernel_L4_M4_22
 21.47      0.93     0.38        1   380.00   380.00  fuse_conv_batchnorm
 13.56      1.17     0.24  8845488     0.00     0.00  rand_uniform
 10.17      1.35     0.18       13    13.85    32.31  make_convolutional_layer
  3.95      1.42     0.07                             sgemm_tcopy_L4_M4_20
  3.39      1.48     0.06        9     6.67     6.67  im2col_cpu_ext
  2.82      1.53     0.05        1    50.00    50.00  resize_image
  2.82      1.58     0.05                             blas_thread_server
  2.26      1.62     0.04                             inner_thread
  1.69      1.65     0.03      761     0.04     0.04  mat_to_image
  1.69      1.68     0.03                             sgemm_kernel_L4_M4_100
  1.13      1.70     0.02                             sgemm_kernel_L2_M4_22
  1.13      1.72     0.02                             sgemm_kernel_L4_M4_20
  1.13      1.74     0.02                             sgemm_ncopy_L4_M4_20
  0.56      1.75     0.01       13     0.77     0.77  add_bias
[pi@raspberrypi] █
```

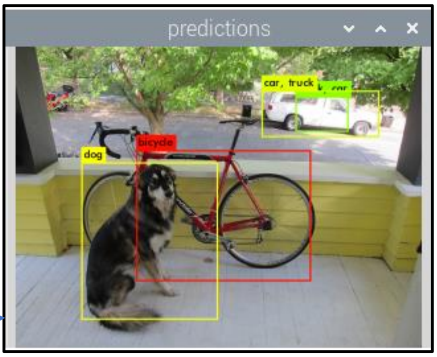
How to deal with ‘gemm_nn’(3/3)



```
[pi@raspberrypi] ./darknet detect cfg/yolov3-tiny.cfg weights/yolov3-tiny.weights data/dog.jpg
   layer   filters  size/strd(dil)      input                output
   0 conv     16       3 x 3/ 1    416 x 416 x   3 ->  416 x 416 x  16 0.150 BF
   1 max               2 x 2/ 2    416 x 416 x  16 ->  208 x 208 x  16 0.003 BF
   2 conv     32       3 x 3/ 1    208 x 208 x  16 ->  208 x 208 x  32 0.399 BF
   3 max               2 x 2/ 2    208 x 208 x  32 ->  104 x 104 x  32 0.001 BF
   4 conv     64       3 x 3/ 1    104 x 104 x  32 ->  104 x 104 x  64 0.399 BF
   5 max               2 x 2/ 2    104 x 104 x  64 ->   52 x  52 x  64 0.001 BF
   6 conv    128       3 x 3/ 1     52 x  52 x  64 ->   52 x  52 x 128 0.399 BF
   7 max               2 x 2/ 2     52 x  52 x 128 ->   26 x  26 x 128 0.000 BF
   8 conv    256       3 x 3/ 1     26 x  26 x 128 ->   26 x  26 x 256 0.399 BF
   9 max               2 x 2/ 2     26 x  26 x 256 ->   13 x  13 x 256 0.000 BF
  10 conv    512       3 x 3/ 1     13 x  13 x 256 ->   13 x  13 x 512 0.399 BF
  11 max               2 x 2/ 1     13 x  13 x 512 ->   13 x  13 x 512 0.000 BF
  12 conv   1024       3 x 3/ 1     13 x  13 x 512 ->   13 x  13 x1024 1.595 BF
  13 conv    256       1 x 1/ 1     13 x  13 x1024 ->   13 x  13 x 256 0.089 BF
  14 conv    512       3 x 3/ 1     13 x  13 x 256 ->   13 x  13 x 512 0.399 BF
  15 conv    255       1 x 1/ 1     13 x  13 x 512 ->   13 x  13 x 255 0.044 BF
  16 yolo
[yolo] params: iou loss: mse, iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00
  17 route  13
  18 conv    128       1 x 1/ 1     13 x  13 x 256 ->   13 x  13 x 128 0.011 BF
  19 upsample                 2x    13 x  13 x 128 ->   26 x  26 x 128
  20 route  19 8
  21 conv    256       3 x 3/ 1     26 x  26 x 384 ->   26 x  26 x 256 1.196 BF
  22 conv    255       1 x 1/ 1     26 x  26 x 256 ->   26 x  26 x 255 0.088 BF
  23 yolo
[yolo] params: iou loss: mse, iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00
Total BFLOPS 5.571
Loading weights from weights/yolov3-tiny.weights...
 seen 64
Done!
data/dog.jpg: Predicted in 1235.776000 milli-seconds.
dog: 81%
bicycle: 38%
car: 71%
truck: 41%
truck: 62%
car: 39%
```

1.2 sec

- use ‘fim’ to see the result
- $ fim predect.png



Make a long story short

- Get Darknet-AlexeyAB version and modify
  - ▶ $ cd ~/work/codes/darknet-projects
  - ▶ $ git clone https://github.com/AlexeyAB/darknet.git
  - ▶ $ mv darknet darknet-alexey-blas
  - ▶ $ cd darknet-alexey-blas
  - ▶ $ patch Makefile < ../patch_Makefile.txt
  - ▶ $ patch src/gemm.c < ../patch_gemm.txt
  - ▶ $ make
  - ▶ $ ./darknet detect cfg/yolov3-tiny.cfg weights/yolov3-tiny.weights data/dog.jpg
  - ▶ $ fim predect.png

Darknet using OpenMP

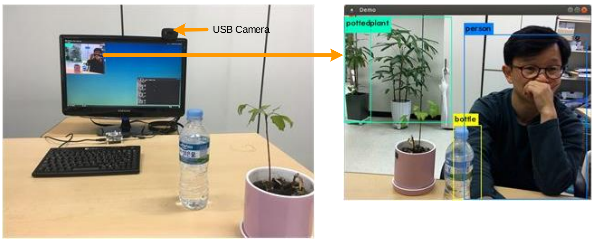
- OpenMP will use multi-thread
  - Install OpenMP
    - $ sudo apt-get update
    - $ sudo apt-get install libomp-dev

```
GPU=0
CUDNN=0
CUDNN_HALF=0
OPENCV=1
AVX=0
OPENMP=1
LIBSO=0
ZED_CAMERA=0
...
```

- Simply set ‘OPENMP’ 1
- It can be run along with other options.
  - OPENCV
  - OPENBLAS
  - GPROF

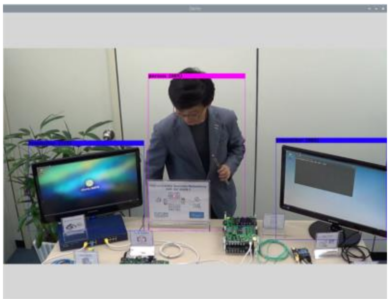
Running Tiny-YOLO with USB-CAM

- $ ./darknet detector demo cfg/coco.data cfg/yolov3-tiny.cfg weights/yolov3-tiny.weights –c 0



Running Tiny-YOLO with video stream

```
$ ./darknet detector demo cfg/coco.data cfg/yolov3-tiny.cfg weights/yolov3-tiny.weights video.mp4
```



㈜퓨쳐디자인시스템
34051 대전광역시 유성구 문지로 193, KAIST 문지캠퍼스, F723호
(042) 864-0211~0212 / contact@future-ds.com / www.future-ds.com

Future Design Systems, Inc.
Faculty Wing F723, KAIST Munji Campus, 193 Munji-ro, Yuseong-gu, Daejeon 34051, Korea
+82-042-864-0211~0212 / contact@future-ds.com / www.future-ds.com

FUTURE
Design Systems