

# Darknet

## - LeNet and YOLO -

2019 - 2020

Ando Ki, Ph.D.

[adki@future-ds.com](mailto:adki@future-ds.com)

## Contents

- What is Darknet
- Building Darknet
- Testing Darknet using LeNet
- 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

## What is Darknet

- Darknet is an open source neural network framework written in C and CUDA (Compute 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 Raspberry 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.

3

## Building Darknet

- |  |   |
|--|---|
| <ul style="list-style-type: none"> <li>■ Visit           <ul style="list-style-type: none"> <li>▶ <a href="https://github.com/AlexeyAB/darknet">https://github.com/AlexeyAB/darknet</a></li> </ul> </li> <li>■ Download           <ul style="list-style-type: none"> <li>▶ make a directory</li> <li>▶ \$ mkdir work &amp;&amp; cd work</li> <li>▶ \$ git clone <a href="https://github.com/AlexeyAB/darknet.git">https://github.com/AlexeyAB/darknet.git</a></li> <li>▶ \$ mv darknet darknet-alexey</li> </ul> </li> <li>■ Modify 'Makefile'           <ul style="list-style-type: none"> <li>▶ cd darknet-alexey</li> <li>▶ \$ vi Makefile               <ul style="list-style-type: none"> <li>➡ set 1 for OpenCV if you installed it.</li> </ul> </li> </ul> </li> <li>■ Compile           <ul style="list-style-type: none"> <li>▶ \$ make</li> </ul> </li> <li>■ At last           <ul style="list-style-type: none"> <li>▶ '<b>darknet</b>': executable</li> </ul> </li> </ul> | <ul style="list-style-type: none"> <li>■ Nvidia CUDA related           <ul style="list-style-type: none"> <li>▶ GPU, CUDNN, CUDNN_HALF</li> </ul> </li> <li>■ x86 Vector related           <ul style="list-style-type: none"> <li>▶ AVX</li> </ul> </li> <li>■ Multi-core/computer related           <ul style="list-style-type: none"> <li>▶ OpenMP</li> </ul> </li> <li>■ Shared library           <ul style="list-style-type: none"> <li>▶ LIBSO</li> </ul> </li> <li>■ 3D camera           <ul style="list-style-type: none"> <li>▶ ZED_CAMERA</li> </ul> </li> </ul> |
|--|---|

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

4

## 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`

5

## Darknet LeNet (1/2)

■ 1. Get Darknet and build Darknet

■ `$ git clone https://github.com/pjreddie/darknet.git`

■ `$ cd darknet`

■ `$ make`

■ `$ cd ..`

■ 2. Get project

■ `$ git clone https://github.com/ashitani/darknet_mnist.git`

■ 3. Get MNIST database

■ `$ darknet_mnist/data/mnist`

■ `$ python download_and_convert_mnist.py`

■ `$ cd ../../`

■ `<now darknet_mnist>`

6

## Darknet LeNet (2/2)

- 4. Train at 'darknet\_mnist' directory
  - \* change './darknet' in 'train.sh' to './darknet/darknet'
  - \* modify 'cfg/mnist.dataset' to change backup directory.
  - 
  - \$ mkdir backup
  - \$ sh ./train.sh
- 5. Inference at 'darknet\_mnist' directory
  - \* change './darknet' in 'predict.sh' to './darknet/darknet'
- \$ cp backup/mnist\_lenet.weights .
- \$ sh ./predict.sh

7

## 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

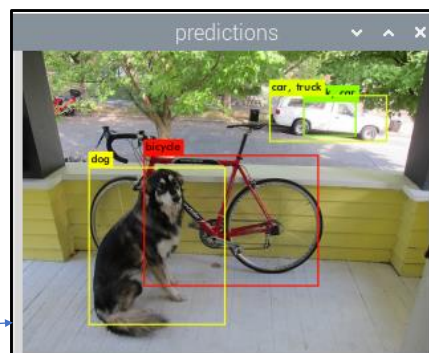
8

## Testing Darknet using Tiny-YOLO (2/2)

```

pi@raspberrypi: ~/work/darknet/AlexeyAB
File Edit Tabs Help
[pi@raspberrypi] ./darknet detect cfg/yolov3-tiny.cfg weights/yolov3-tiny.weights data/dog.jpg
layer  filters  size/stride  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          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          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          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          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          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          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 x 1024 1.595 BF
13 conv 256         1 x 1/ 1    13 x 13 x 1024 -> 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. 31 sec
dog: 81%
bicycle: 38%
car: 71%
truck: 41%
truck: 62%
car: 39%

```



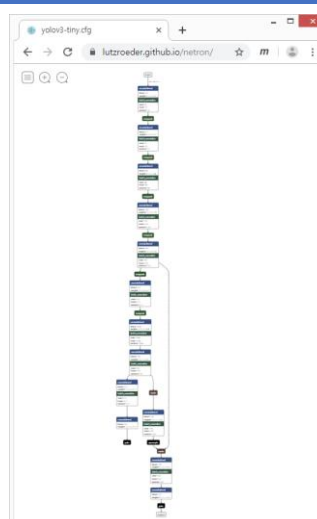
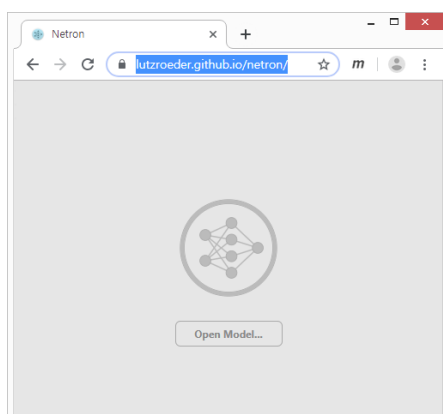
Type 'q' on the picture in order to quit.

9

## Network visualizer

■ <https://github.com/lutroeder/netron>

■ <https://lutroeder.github.io/netron/>



10

## Contents

- 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

11

## 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
  - ▶ CFLAG+=-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
...
```

12

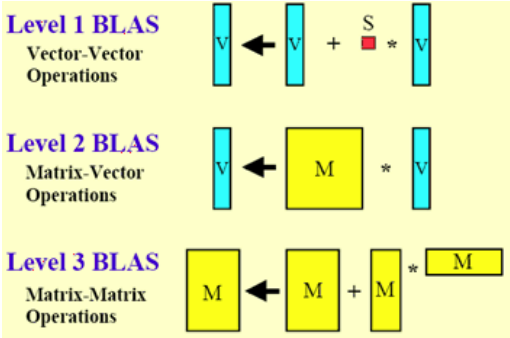
# Darknet profiling (2/2)

```
File Edit Tabs Help
[pi@raspberrypi: ~/work/darknet/AlexeyAB]
[pi@raspberrypi] ./darknet detect cfg/yolov3-tiny.cfg weights/yolov3-tiny.weights data/dog.jpg
layer filters size/stride input output
0 conv 16 3 x 3/ 1 416 x 416 x 3 -> 208 x 208 x 16 0.150 BF
1 max 2 x 2/ 2 208 x 208 x 16 -> 104 x 104 x 16 0.003 BF
2 conv 32 3 x 3/ 1 208 x 208 x 16 -> 104 x 104 x 32 0.399 BF
3 max 2 x 2/ 2 104 x 104 x 32 -> 52 x 52 x 32 0.001 BF
4 conv 64 3 x 3/ 1 104 x 104 x 32 -> 52 x 52 x 64 0.399 BF
5 max 2 x 2/ 2 52 x 52 x 64 -> 26 x 26 x 64 0.001 BF
6 conv 128 3 x 3/ 1 52 x 52 x 64 -> 26 x 26 x 128 0.399 BF
7 max 2 x 2/ 2 26 x 26 x 128 -> 13 x 13 x 128 0.000 BF
8 conv 256 3 x 3/ 1 26 x 26 x 128 -> 13 x 13 x 256 0.399 BF
9 max 2 x 2/ 2 13 x 13 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 x 1024 1.595 BF
13 conv 256 1 x 1/ 1 13 x 13 x 1024 -> 13 x 13 x 256 0.088 BF
14 conv 512 3 x 3/ 1 13 x 13 x 256 -> 13 x 13 x 512 0.399 BF
15 conv 256 1 x 1/ 1 13 x 13 x 512 -> 13 x 13 x 256 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 256 1 x 1/ 1 26 x 26 x 256 -> 26 x 26 x 256 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/ gmm_openblas.c* gmon.out
image_yolov3.sh* include/ json_njpeg_streams.sh*
LICENSE Makefile Makefile_openblas* net_cam_v3.sh*
obj/ predictions.jpg README.md results/
scripts/ video_v2.sh* video_yolov3.sh*
weights/
[pi@raspberrypi]
```

```
File Edit Tabs Help
[pi@raspberrypi: ~/work/darknet/AlexeyAB]
[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]
```

# BLAS: Basic Linear Algebra Subprograms

- Level1: vector-vector operations
  - $V = V + s \times V$
- Level2: matrix-vector operations
  - $V = V + A \times V$
- Level3: matrix-matrix operations
  - $C = C + A \times B$



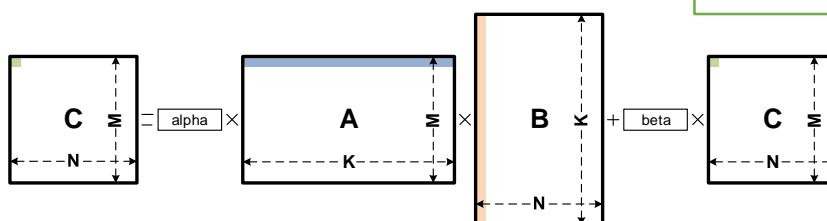
Level	Data Move ment	Floating-Point Ope rations	Example
Level 1	$O(N)$	$O(N)$	DDOT
Level 2	$O(N^2)$	$O(N^2)$	DGEMV
Level 3	$O(N^2)$	$O(N^3)$	DGEMM

# GEMM of BLAS

## ■ GEMM

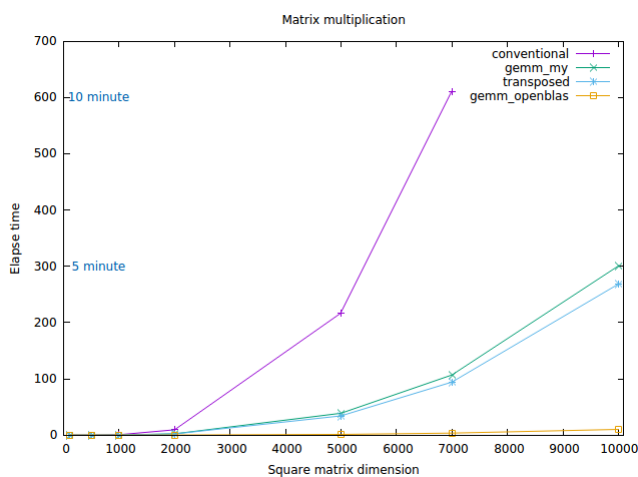
- ▶ General Matrix Multiplication
- ▶  $C = \text{BETA} \times C + \text{ALPHA} \times A \times B$
- ▶ BLAS Level 3
  - SGEMM: single precision
  - DGEMM: double precision
  - CGEMM: single precision complex
  - ZGEMM: double precision complex

```
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);
```



15

## Square matrix multiplication



Ubuntu 16.04 / 64-bit on x86\_64 / Intel Core i7-3770 CPU @ 3.4GHz x 8 / 16GigaByte

16

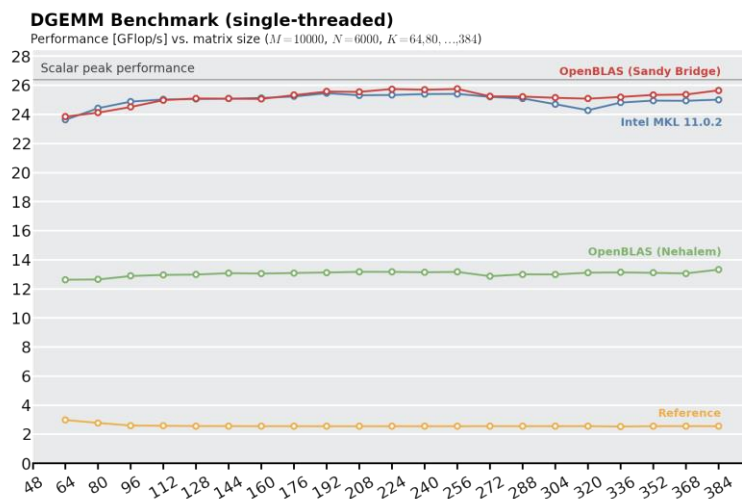


## BLAS packages

- OpenBLAS
  - ▶ [www.openblas.net](http://www.openblas.net)
- Intel MKL (Math Kernel Library)
  - ▶ Commercial and optimized for Intel CPU
    - ➔ `$ source /opt/intel/mkl/bin/mklvars.sh intel64`
- ATLAS

17

## Performance comparison



18

## How to deal with 'gemm\_nn'(1/3)

```
src/gemm.c
#ifdef OPENBLAS
#include <cbblas.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
               , N
               , K
               , ALPHA
               , A
               , lda
               , B
               , ldb
               , BETA
               , C
               , ldc
               // OPENBLAS_CONST blasint ldc
    );
}
```

- 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
...
```

```
#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
```

19

## How to deal with 'gemm\_nn'(1/3)

```
src/gemm.c
#ifdef OPENBLAS
#include <cbblas.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
               , N
               , K
               , ALPHA
               , A
               , lda
               , B
               , ldb
               , BETA
               , C
               , ldc
               // OPENBLAS_CONST blasint ldc
    );
}
```

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

```
Makefile
...
OPENBLAS=1
...
ifeq ($(OPENBLAS), 1)
CFLAGS+=-DOPENBLAS
LDFLAGS+=-lopenblas
endif
...
```

```
#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
```

20

## 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: ~/work/darknet/AlexeyAB
File Edit Tabs Help
[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      total
time  seconds  seconds  calls  ms/call  ms/call  name
21.47    0.93    0.38      1    388.00    388.00  sgemm_kernel_L4_M4_22
13.56    1.17    0.24   8845488    0.00     0.00  fuse_conv_batchnorm
10.17    1.35    0.18      13    13.85    32.31  make_convolutional_layer
3.95     1.42    0.07      13    13.85    32.31  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      1    50.00    50.00  blas_thread_server
2.26     1.62    0.04      13    13.85    32.31  inner_thread
1.69     1.65    0.03     761     0.04     0.04  mat_to_image
1.69     1.68    0.03      13    13.85    32.31  sgemm_kernel_L4_M4_100
1.13     1.70    0.02      13    13.85    32.31  sgemm_kernel_L2_M4_22
1.13     1.72    0.02      13    13.85    32.31  sgemm_kernel_L4_M4_20
1.13     1.74    0.02      13    13.85    32.31  sgemm_ncopy_L4_M4_20
0.56     1.75    0.01      13     0.77     0.77  add_bias

```

21

## How to deal with 'gemm\_nn'(3/3)

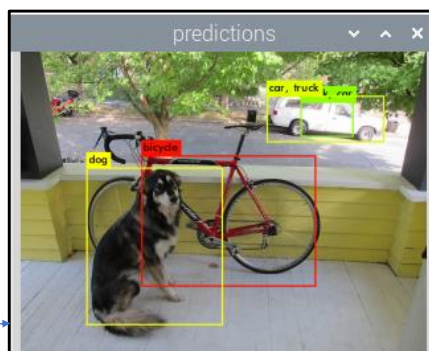
```

pi@raspberrypi: ~/work/darknet/AlexeyAB
File Edit Tabs Help
[pi@raspberrypi] ./darknet detect cfg/yolov3-tiny.cfg weights/yolov3-tiny.weights data/dog.jpg
layer  filters  size/stride(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           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   3           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           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           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           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  13          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 x 1024 1.595 BF
13 conv 256         1 x 1/ 1          13 x 13 x 1024 -> 13 x 13 x 256 0.080 BF
14 conv 512         3 x 3/ 1          13 x 13 x 256 -> 13 x 13 x 512 0.399 BF
15 conv 256         1 x 1/ 1          13 x 13 x 512 -> 13 x 13 x 256 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...
Done!
data/dog.jpg: Predicted in 1235.776000 milli-seconds.
dog: 81%
bicycle: 38%
car: 71%
truck: 41%
car: 62%
car: 39%

```

1.2 sec

- use 'fim' to see the result
- \$ fim predict.png



22

## Make a long story short

### ■ Get Darknet-AlexeyAB version and modify (for Raspberry Pi Raspbian case)

- ▶ \$ 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 predefect.png

23

## Darknet using OpenMP

### ■ OpenMP will use multi-thread

- ▶ Install OpenMP
  - ➔ \$ sudo apt-get update
  - ➔ \$ sudo apt-get install libomp-dev

### ■ Simply set 'OPENMP' 1

### ■ It can be run along with other options.

- ▶ OPENCV
- ▶ OPENBLAS
- ▶ GPROF

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

24

## What is OpenCV

- OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library.
  - ▶ <https://opencv.org>
- What OpenCV can do :
  - ▶ 1. Read and Write Images.
  - ▶ 2. Detection of faces and its features.
  - ▶ 3. Detection of shapes like Circle, rectangle etc in a image.
  - ▶ 4. Text recognition in images. (number of car license plate)
  - ▶ 5. Modifying image quality and colors
  - ▶ 6. Developing Augmented reality apps.
  - ▶ 7. Controlling camera
- Which Language it supports :
  - ▶ 1. C++
  - ▶ 2. Android SDK
  - ▶ 3. Java
  - ▶ 4. Python
  - ▶ 5. C (Not recommended)



25

## Installing OpenCV on Ubuntu

- If OpenCV is not installed yet, do as follows. (You need root password.)
  - ▶ \$ sudo apt-get install libopencv-dev python-opencv ffmpeg
- Set 'OPENCV' macro '1' in Makefile
  - ▶ do not forget to run 'make clean' in order to remove old files.

Makefile

```
GPU=0
CUDNN=0
OPENCV=1
OPENMP=0
DEBUG=0

# This is what I use, uncomment if you know your arch and want to specify
ARCH= -gencode arch=compute_52,code=compute_52

VPATH=./src/./examples
SLIB=libdarknet.so
ALIB=libdarknet.a
EXEC=darknet
OBJDIR=./obj/

CC=gcc
NVCC=nvcc
...
```

Set 'OPENCV' 1

26

## Run YOLO with OpenCV

- Run darknet with yolo configuration with OpenCV.
  - \$ ./darknet detect cfg/yolov3.cfg weights/yolov3.weights data/horses.jpg

```

[adk@gadki-ubuntu] ./darknet detect cfg/yolo.cfg weights/yolo.weights data/horses.jpg
layer  filters  size  input              output
0 conv  32  3 x 3 / 1  608 x 608 x 3  ->  608 x 608 x 32
1 max   2  2 x 2 / 2  608 x 608 x 32  ->  304 x 304 x 32
2 conv  64  3 x 3 / 1  304 x 304 x 32  ->  304 x 304 x 64
3 max   2  2 x 2 / 2  304 x 304 x 64  ->  152 x 152 x 64
4 conv  128  3 x 3 / 1  152 x 152 x 64  ->  152 x 152 x 128
5 conv  64  1 x 1 / 1  152 x 152 x 128  ->  152 x 152 x 64
6 conv  128  3 x 3 / 1  152 x 152 x 64  ->  152 x 152 x 128
7 max   2  2 x 2 / 2  152 x 152 x 128  ->  76 x 76 x 128
8 conv  256  3 x 3 / 1  76 x 76 x 128  ->  76 x 76 x 256
9 conv  128  1 x 1 / 1  76 x 76 x 256  ->  76 x 76 x 128
10 conv 256  3 x 3 / 1  76 x 76 x 128  ->  76 x 76 x 256
11 max   2  2 x 2 / 2  76 x 76 x 256  ->  38 x 38 x 256
12 conv 512  3 x 3 / 1  38 x 38 x 256  ->  38 x 38 x 512
13 conv 256  1 x 1 / 1  38 x 38 x 512  ->  38 x 38 x 256
14 conv 512  3 x 3 / 1  38 x 38 x 256  ->  38 x 38 x 512
15 conv 256  1 x 1 / 1  38 x 38 x 512  ->  38 x 38 x 256
16 conv 512  3 x 3 / 1  38 x 38 x 256  ->  38 x 38 x 512
17 max   2  2 x 2 / 2  38 x 38 x 512  ->  19 x 19 x 512
18 conv 1024 3 x 3 / 1  19 x 19 x 512  ->  19 x 19 x 1024
19 conv 512  1 x 1 / 1  19 x 19 x 1024  ->  19 x 19 x 512
20 conv 1024 3 x 3 / 1  19 x 19 x 512  ->  19 x 19 x 1024
21 conv 512  1 x 1 / 1  19 x 19 x 1024  ->  19 x 19 x 512
22 conv 1024 3 x 3 / 1  19 x 19 x 512  ->  19 x 19 x 1024
23 conv 1024 3 x 3 / 1  19 x 19 x 1024  ->  19 x 19 x 1024
24 conv 1024 3 x 3 / 1  19 x 19 x 1024  ->  19 x 19 x 1024
25 route 16
26 conv 64  1 x 1 / 1  38 x 38 x 512  ->  38 x 38 x 64
27 reorg  / 2  38 x 38 x 64  ->  19 x 19 x 256
28 route 27,24
29 conv 1024 3 x 3 / 1  19 x 19 x 256  ->  19 x 19 x 1024
30 conv 425 1 x 1 / 1  19 x 19 x 1024  ->  19 x 19 x 425
31 detection
mask scale: using default '1.000000'
Loading weights from weights/yolo.weights...done!
data/horses.jpg: Predicted in 10.762035 seconds.
horse: 46%
horse: 59%
cow: 26%
horse: 91%

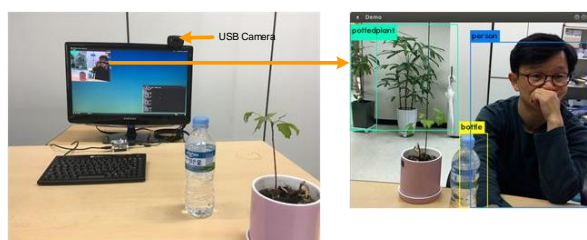
```

Type 'q' on the picture in order to quit.

27

## Running Tiny-YOLO with USB-CAM

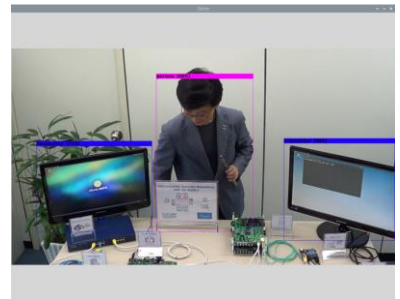
- \$ ./darknet detector demo cfg/coco.data cfg/yolov3-tiny.cfg weights/yolov3-tiny.weights -c 0



28

## Running Tiny-YOLO with video stream

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



29

(주)퓨처디자인시스템

34051 대전광역시 유성구 문지로 193, KAIST 문지캠퍼스, F723호  
(042) 864-0211~0212 / [contact@future-ds.com](mailto:contact@future-ds.com) / [www.future-ds.com](http://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](mailto:contact@future-ds.com) / [www.future-ds.com](http://www.future-ds.com)



**FUTURE**  
Design Systems