

LeNet-5 using DLR

- running model and profiling -

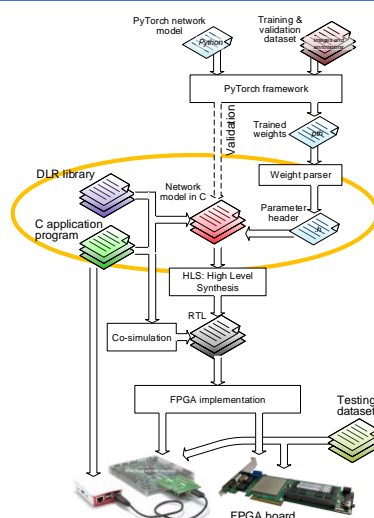
2021

Ando Ki, Ph.D.

adki@future-ds.com

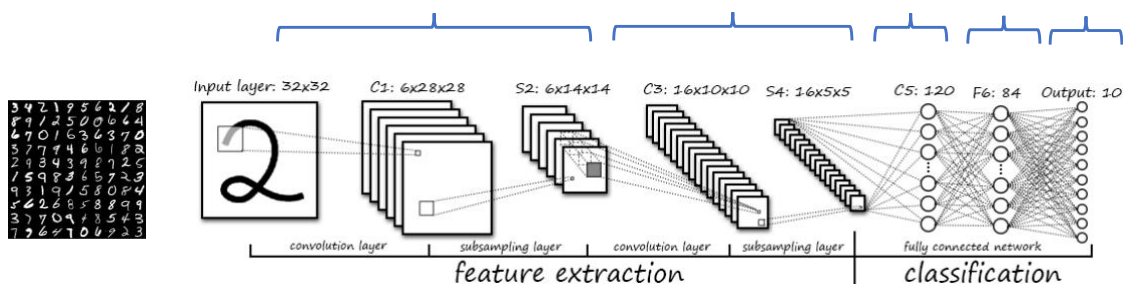
Table of contents

- LeNet-5
- Preparing DLR
- Building LeNet-5 model in C
- Compiling and running
- Profiling



LeNet-5 for MNIST

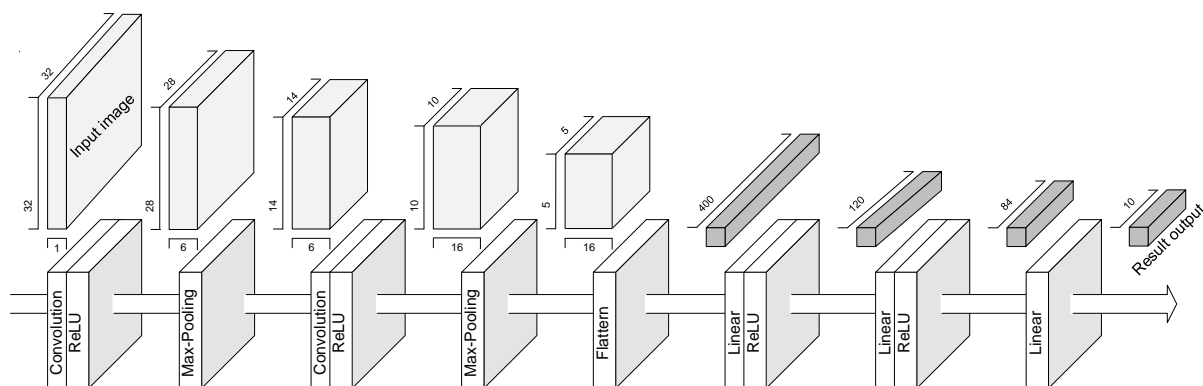
- LeNet is one of the popular convolutional networks, and works well on digit classification tasks.
 - 1024 (32x32) inputs of black and white → converted to floating number 0.0 ~ 1.0
 - 10 outputs representing digit 0 to 9



Copyright (c) Ando Ki

3

LeNet-5 network



Copyright (c) Ando Ki

4

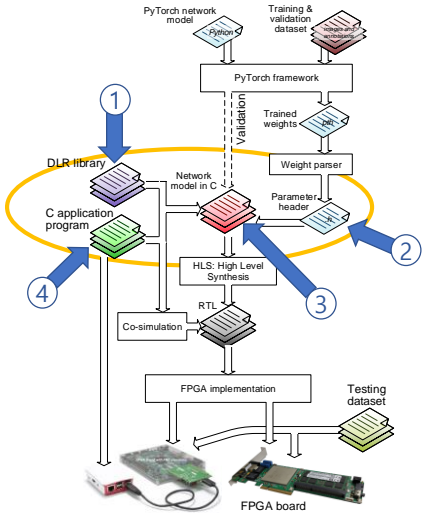
Modified LeNet-5 network

Layer		feature map (channel x W x H)		kernel size	stride	padding	Activation	Parameters (bias+weight)	Feature map
input	image	1	32x32	1,024
1	CONV	6	28x28	5x5	1	0	ReLU	6+150	4,704
2	MaxPOOL	6	14x14	2x2	2	0	.	0	1,176
3	CONV	16	10x10	5x5	1	0	ReLU	16+2,400	1,600
4	MaxPOOL	16	5x5	2x2	2	0	.	0	400
5	FC	.	120	.	.	.	ReLU	120+48,000	120
6	FC	.	84	.	.	.	ReLU	84+10,080	84
7	FC	.	10	.	.	.	softmax	10+840	10
								61,706	9,118

bias = # channel
weight for convolution= # in_channel x # out_channel x # kernel_size
feature map = # channel x # in_feature_map
weight for FC = # in_feature x # out_feature

What are required

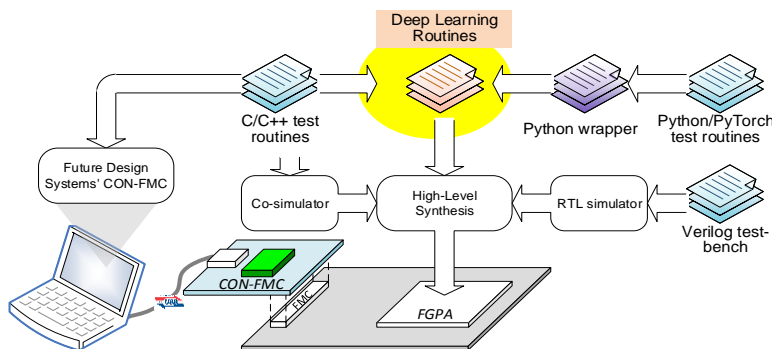
- (1) Deep Learning Routines
- (2) LeNet-5 trained parameters (weights)
- (3) LeNet-5 model in C/C++
- (4) Program to run LeNet-5 model



What is DLR

- 'DLR (Deep Learning Routines)' as a part of DPU (Deep Learning Processing Unit) is a collection of high-level synthesizable C/C++ routines for deep learning inference network.

► https://github.com/github-fds/Deep_Learning_Routines



Copyright (c) Ando Ki

7

How to prepare DLR library

- Get a clone
 - \$ git clone https://github.com/github-fds/Deep_Learning_Routines.git
- Go to 'src' directory
 - \$ cd Deep_Learning_Routines/v1.3/src
- Compile and install (mind '--std=c++11')
 - \$ make
 - \$ make install
- See 'include' and 'lib' directories
 - \$ cd ..
 - \$ ls include lib

Copyright (c) Ando Ki

8

How to prepare and run the project

- Get a clone
 - ▶ \$ git clone https://github.com/adki/DLR_Projects.git
- Go to 'LeNet-5/LeNet-5.dlr/native.cpp' directory
 - ▶ \$ cd DLR_Projects/LeNet-5/LeNet-5.dlr/native.cpp
- Compile and install
 - ▶ (do not forget to set 'DLR_HOME' macro in 'Makefile')
 - ▶ \$ make
 - ▶ \$ make run

```

LeNet-5
|-- LeNet-5.dlr
|   |-- native.cpp
|   |-- src
|-- LeNet-5.dlr.fpga
|   |-- hw
|   |-- hls
|   |   |-- tcl.float
|   |   |-- impl
|   |       |-- vivado.zed.confmc.float
|   |       |-- xdc
|   |-- iplib
|   |-- bfm_axi
|   |-- sw.native
|   |   |-- lenet.confmc
|   |   |-- src
|-- LeNet-5.pytorch
|   |-- samples
|   |-- src
|-- LeNet-5.pytorch.dlr
|   |-- src

```

Where the trained parameter resides.

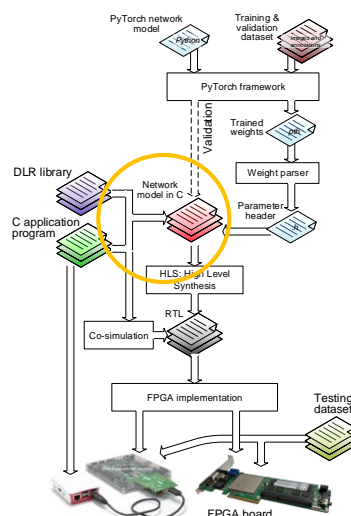
Table of contents

- LeNet-5
- Preparing DLR
- Building LeNet-5 model in C
- Compiling and running
- Profiling

```

LeNet-5
|-- LeNet-5.dlr
|   |-- native.cpp
|   |-- LeNet-5.dlr.fpga
|   |-- LeNet-5.pytorch
|   |-- LeNet-5.pytorch.dlr

```



lenet5.cpp

```
#include "dlr.hpp"
#include "lenet5_params.h" // come from LeNet-5.pytorch

void lenet5(      DTYPE classes[10]
               , const DTYPE image [32][32]
               , #if !defined(__SYNTHESIS__)
               , const int rigor
               , const int verbose
               #endif
               )
{
    DTYPE  c1_out_data[6][28][28];
    const DTYPE (*c1_in_data)[32][32]=(DTYPE (*)[32][32])image; // [1][32][32]
    const DTYPE (*c1_kernel)[1][5][5]=(DTYPE (*)[1][5][5])conv1_weight; // [6][1][5][5]
    const DTYPE (*c1_bias)=conv1_bias; // [6]
    const uint16_t c1_out_size=28;
    const uint16_t c1_in_size=32;
    const uint8_t c1_kernel_size=5;
    const uint16_t c1_bias_size=6;
    const uint16_t c1_in_channel=1;
    const uint16_t c1_out_channel=6;
    const uint8_t c1_stride=1;
    const uint8_t c1_padding=0;
```

```
Convolution2d<DTYPE> (
    (DTYPE *)c1_out_data
    , (DTYPE *)c1_in_data
    , (DTYPE *)c1_kernel
    , (DTYPE *)c1_bias
    , c1_out_size
    , c1_in_size
    , c1_kernel_size
    , c1_bias_size
    , c1_in_channel
    , c1_out_channel
    , c1_stride
    , c1_padding
    , #if !defined(__SYNTHESIS__)
    , rigor
    , verbose
    #endif
);
```

Copyright (c) Ando Ki

11

main.cpp

```
int main( int argc, char* argv[]) {
    int num_images = arg_parser(argc, argv);

    int idx;
    for (idx=0; idx<num_images; idx++) {
        float classes[NUM_CLASSES];
        float image[IMAGE_HEIGHT][IMAGE_WIDTH];
        get_image_data( image_files[idx], (float *)image, 0, 2);
        lenet5( classes
              , image
              , #if !defined(__SYNTHESIS__)
              , rigor
              , verbose
              #endif
              );
        float classes_softmax[NUM_CLASSES];
        softmax(classes_softmax, classes);
        printf("[%s]\n", image_files[idx]);
        int idy;
        for (idy=0; idy<NUM_CLASSES; idy++) {
            printf("%t[%d]: %f:%8.3f\n", idy, classes_softmax[idy], classes[idy]);
        }
    }
    return 0;
}
```

Copyright (c) Ando Ki

12

Compile and run

- For g++ 5.x, mind '--std=c++11'

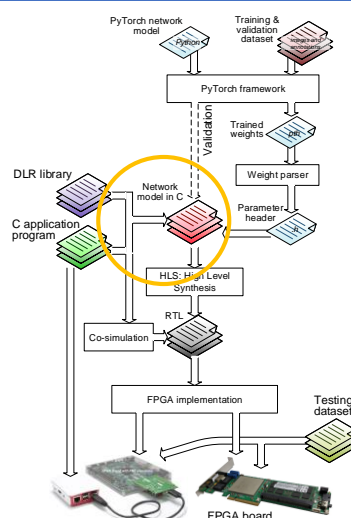
- ▶ \$ make
- ▶ \$ make run
- ▶ or
- ▶ \$ make run.all

```
adki@ando-pc: ~/work/projects/DPULR_Projects.master/LeNet-5/...
File Edit View Search Terminal Help
(base) [adki@ando-pc] make run
./lenet5 --rigor ../LeNet-5.pytorch/samples/t_00_c4.png
[.../LeNet-5.pytorch/samples/t_00_c4.png]
[0]: 0.000000: -13.836
[1]: 0.000000: -14.100
[2]: 0.000000: -0.527
[3]: 0.000000: -19.281
[4]: 1.000000: 40.089
[5]: 0.000000: -2.063
[6]: 0.000000: -4.218
[7]: 0.000000: 1.709
[8]: 0.000000: -1.059
[9]: 0.000000: 3.265
(base) [adki@ando-pc]
```

Table of contents

- LeNet-5
- Preparing DLR
- Building LeNet-5 model in C
- Compiling and running
- Profiling

```
LeNet-5
|-- LeNet-5.dlr
|   |-- native.cpp
|   |-- LeNet-5.dlr.fpga
|   |-- LeNet-5.pytorch
|   +-- LeNet-5.pytorch.dlr
```



Compile and run

■ For g++ 5.x, mind '--std=c++11'

- ▶ \$ make run.profile

■ GNU GCC와 gprof 사용

- ▶ 컴파일 단계에서 '-pg' 선택자 사용
- ▶ 프로그램 수행 결과로 'gmon.out' 파일 생성
- ▶ 'gprof' 프로그램으로 분석

■ 1. compile and link using '-pg' option

■ 2. run as normal

■ 3. run using 'gprof'

- ▶ \$ gprof lenet5 gmon.out > profile.txt

Compile and run

```

adki@ando-pc: ~/work/projects/DPU/DLR_Projects.master/LeNet-5/LeNet-5.dlr/native.cpp
File Edit View Search Terminal Help
[adki@ando-pc] pwd
/home/adki/work/projects/DPU/DLR_Projects.master/LeNet-5/LeNet-5.dlr/native.cpp
[adki@ando-pc] make clean
/bin/rm -f *.o
/bin/rm -fr obj
/bin/rm -f resized.png reverted.png
/bin/rm -f gmon.out profile.txt
[adki@ando-pc] make run.profile
g++ -c -pg -Isrc -I../Deep_Learning_Routines.master/v1.3/include -I../LeNet-5.pytorch -DDEMBED_ReLU=1 -o obj/lenet5.o src/lenet5.cpp
g++ -pg -o lenet5 obj/lenet5.o obj/main.o -Wl,-Bstatic -L../LeNet-5.pytorch -DEMBED_ReLU=1 -o obj/main.o src/main.cpp
./lenet5 ../LeNet-5.pytorch/samples/t_00_c4.png
[../LeNet-5.pytorch/samples/t_00_c4.png]
[0]: 0.000000: -14.162
[1]: 0.000000: -6.505
[2]: 0.000000: -22.498
[3]: 0.000000: -20.053
[4]: 1.000000: 51.848
[5]: 0.000000: -3.359
[6]: 0.000000: -5.028
[7]: 0.000000: 0.822
[8]: 0.000000: -7.590
[9]: 0.000000: 10.531
gprof lenet5 gmon.out > profile.txt
[adki@ando-pc] more profile.txt

```


Profiling results

```
adki@ando-pc: ~/work/projects/DPU/DLR_Projects.master/LeNet-5/dlr/native.cpp
File Edit View Search Terminal Help
Flat profile:
Each sample counts as 0.01 seconds.
no time accumulated

% cumulative self self total
time seconds seconds calls Ts/call Ts/call name
0.00 0.00 0.00 2 0.00 0.00 void Pooling2dMax(float, 1, 0, 1036831949u>{float*, float const*, unsigned short, unsigned
short, unsigned char, unsigned short, unsigned char, unsigned char, int, int, int)
0.00 0.00 0.00 2 0.00 0.00 void Convolution2d(float>{float*, float const*, float const*, float const*, unsigned short
, unsigned short, unsigned char, unsigned short, unsigned short, unsigned short, unsigned char, int, int)
0.00 0.00 0.00 2 0.00 0.00 void LinearId(float, 1, 0, 1036831949u>{float*, float const*, float const*, float const*,
unsigned short, unsigned short, unsigned short, int, int)
0.00 0.00 0.00 1 0.00 0.00 lenet5(float*, float const (*) [32], int, int)
0.00 0.00 0.00 1 0.00 0.00 void LinearId(float, 0, 0, 1036831949u>{float*, float const*, float const*, float const*,
unsigned short, unsigned short, unsigned short, int, int)

% the percentage of the total running time of the
time program used by this function.

cumulative a running sum of the number of seconds accounted
seconds for by this function and those listed above it.

self the number of seconds accounted for by this
More--(15%)
```