GUINNESS Tutorial (Version 0.2)

Hiroki Nakahara, Tokyo Institute of Technology, Japan

1. Introduction

1.1 What is the “GUINNESS”?

The GUINNESS (GUI based a binarized neural network synthesizer for an FPGA) is a GUI based framework includes both a training on a GPU, and a bitstream generation for an FPGA using the Xilinx Inc. SDSoC. This tool uses the Chainer deep learning framework to train a binarized CNN. Also, it uses optimization techniques for an FPGA implementation. Details are shown in following papers:

[Yonekawa IPDPSW2017] H. Yonekawa and H. Nakahara, "On-Chip Memory Based Binarized Convolutional Deep Neural Network Applying Batch Normalization Free Technique on an FPGA," IPDPS Workshops, 2017, pp. 98-105.
[Nakahara FPL2017] H. Nakahara et al., "A Fully Connected Layer Elimination for a Binarized Convolutional Neural Network on an FPGA", FPL, 2017, (to appear).

1.2 Requirements

Ubuntu 14.04 or 16.04 (Choose appropriate version for the SDSoC)
Python 2.7.6+
CUDA 8.0 (+GPU), not necessary to install a cuDNN library
Chainer 1.23.0 or 1.24.0

SDSoC 2016.4 (or 2017.1)
FPGA board: Xilinx ZC702, ZCU102, Digilent Zedboard, Zybo
(In the near future, I will support the PYNQ board)

PyQt4, matplotlib, python-opencv2, numpy, scipy,

2. Setup

Install the following python libraries:

The Chainer, which is a deep learning framework, is necessary, however, the current version (2.0) is not supported by the GUINNESS. So, install the previous version (1.24.0) following command.

```
$sudo pip install chainer==1.24.0
```

Note: Due to a conflict between SDSoC's clang and Chainer's one, please disable the PATH of the Xilinx tool. We are investigating the reason, and will respond ASAP.

Next, since the GUINNESS is based on a GUI, install PyQt4 (not PyQt5!) following command.

```
$sudo apt-get install python-qt4 pyqt4-dev-tools
```

3. Binarized CNN design in the GUINNESS

This tutorial implements an image classification on the Digilent Inc. Zedboard using the Xilinx SDSoC. It also uses a web camera to capture the image.

3.1 Dataset Generation

The GUINNESS uses a training dataset and its label one. We provide the dataset generator using given images. This tutorial uses the following dataset.

[Cars Dataset] Jonathan Krause, Michael Stark, Jia Deng, Li Fei-Fei, "3D Object Representations for Fine-Grained Categorization," 4th IEEE Workshop on 3D Representation and Recognition, at ICCV 2013 (3dRR-13). Sydney, Australia. Dec. 8, 2013.
http://ai.stanford.edu/~jkrause/cars/car_dataset.html
[Pet Dataset] O. M. Parkhi, A. Vedaldi, A. Zisserman, C. V. Jawahar, "Cats and Dogs," IEEE Conference on Computer Vision and Pattern Recognition, 2012.
http://www.robots.ox.ac.uk/~vgg/data/pets/
[Airplane Dataset] L. Fei-Fei, R. Fergus and P. Perona, "Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories," IEEE. CVPR 2004, Workshop on Generative-Model Based Vision. 2004.
http://www.vision.caltech.edu/Image_Datasets/Caltech101/

We corrected above images into three folders. Download it from
https://www.dropbox.com/s/59fx0r5fuqmuo3j/class3_images.zip?dl=0 (about 100MB zip file)
Then, unzip it to the same location of the GUINNESS (guinness.py). Then type the following command to generate training dataset.

```
$ python gen_training_data.py --pathfile list.txt --dataset class3 --size 48 --keepaspect yes
```

It takes a few minutes, after that, following dataset are generated.

class3_dataset.pkl … it is compressed image set to train the CNN
class3_label.pkl … it is assigned labels (number) to the training image set for the CNN
class3_tag.txt … it contains the strings assigned to label dataset

"--pathfile" option specifies the file which indicates the image folder to be assigned the image class.

“--dataset” option specifies the dataset name for above dataset.
“--size” option specifies the pixel size for the CNN input image.
“--keepaspect” option requires keeping aspect ratio for the image set.

```
$ cat list.txt
./class3_images/airplane800 airplane
./class3_images/pets800 pets
./class3_images/car800 car
```

In the path file, each row corresponds to the assigned label number (0,1,2,...). The first term in the line specifies the path for the image file folder and the second one denotes the assigned tag to be used for the verification.

3.2 Run the GUINNESS GUI

After the datasets are generated, then, run the GUINNESS GUI with the following command.

```
$ python guinness.py
```

The GUINNESS GUI appears on the desktop. It consists of a project setting part, a target CNN specification part, a training parameter setup part, and a code generation part for an FPGA.

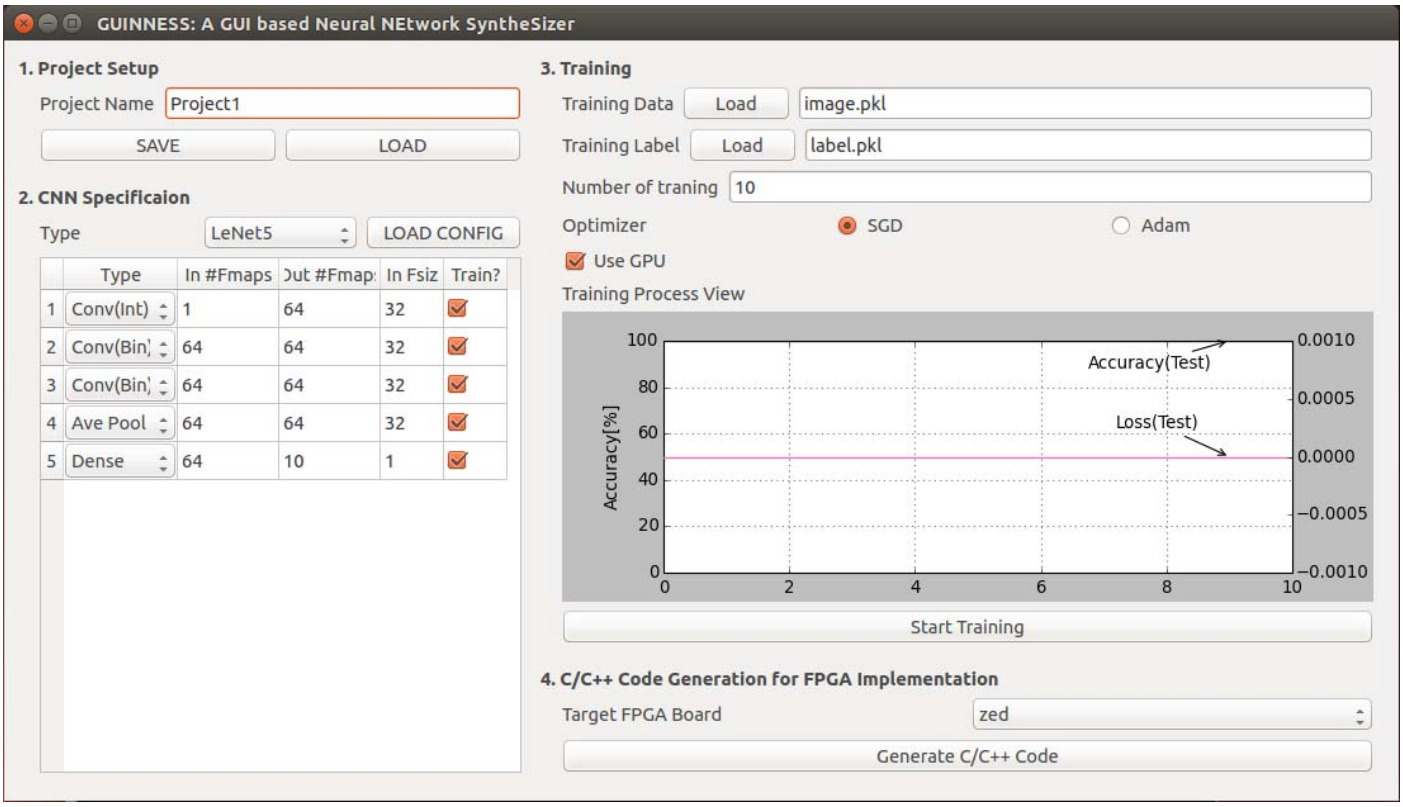


Fig.3.1: GUINNESS GUI

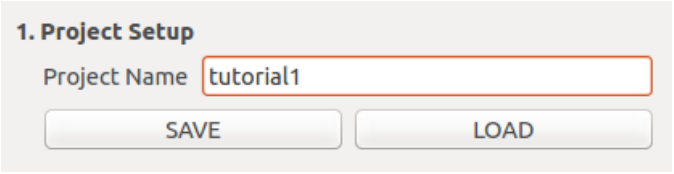


Fig.3.2: Specify the project name

First, you specify the project name as shown in Fig.3.2. In the tutorial, type “tutorial1” to the project setup label.

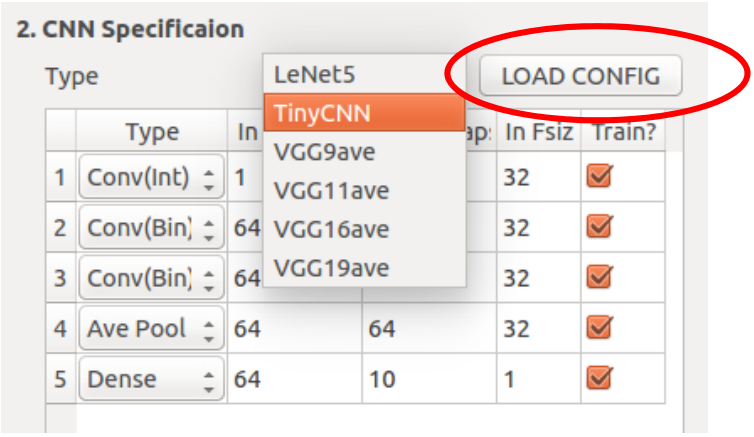


Fig.3.3: Load a preset binarized CNN parameters

Second, you specify the target CNN parameters including the number of layers, layer types, a feature map size. This part takes a long time since there are many combinations of parameters. Fortunately, the GUINNESS provides these parameters as shown in Fig.3.3. Specify a “TinyCNN” preset, then click a “LOAD CONFIG” button. Then, the CNN specification is loaded to the GUI as shown in Fig.3.4.

In that case, you can change the number of layers by right click on the GUI as shown in Fig.3.5, and change operation in the layer as shown in Fig.3.6. Also, you can change the number of feature maps (In #Fmaps/Out #Fmaps) by directly changing in each cell. Note that, you should not specify the “In Fsiz”, which is the feature map size, since the GUINNESS GUI automatically calculate these size by loading the dataset generated in the previous section.

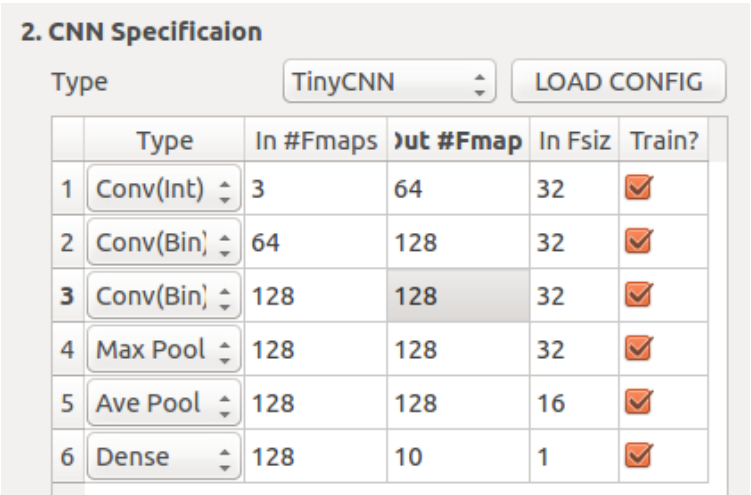


Fig.3.4: TinyCNN preset parameter is loaded.

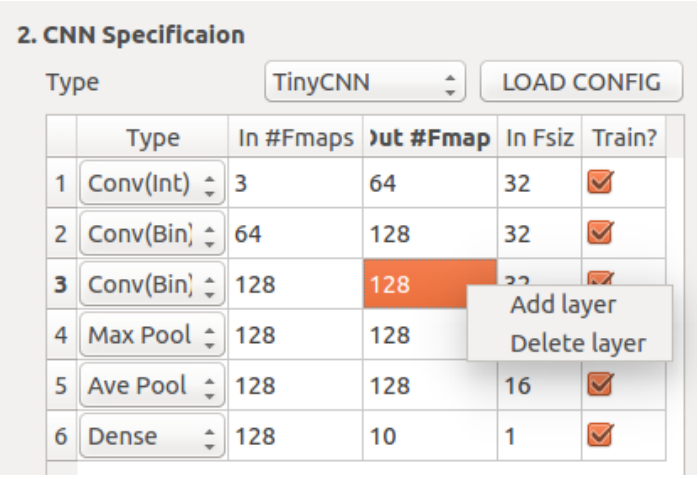


Fig.3.5: Add/Delete layer (optional)

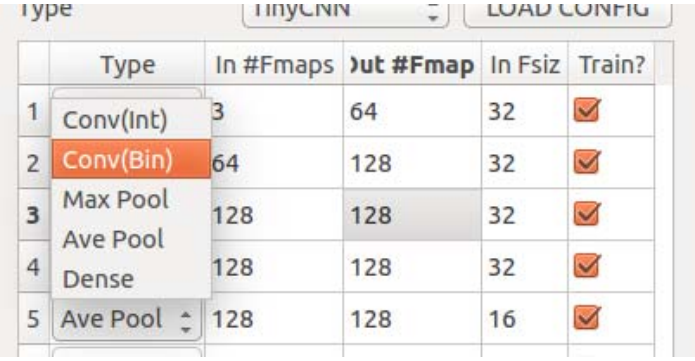


Fig.3.6 Change operation in the layer (optional)

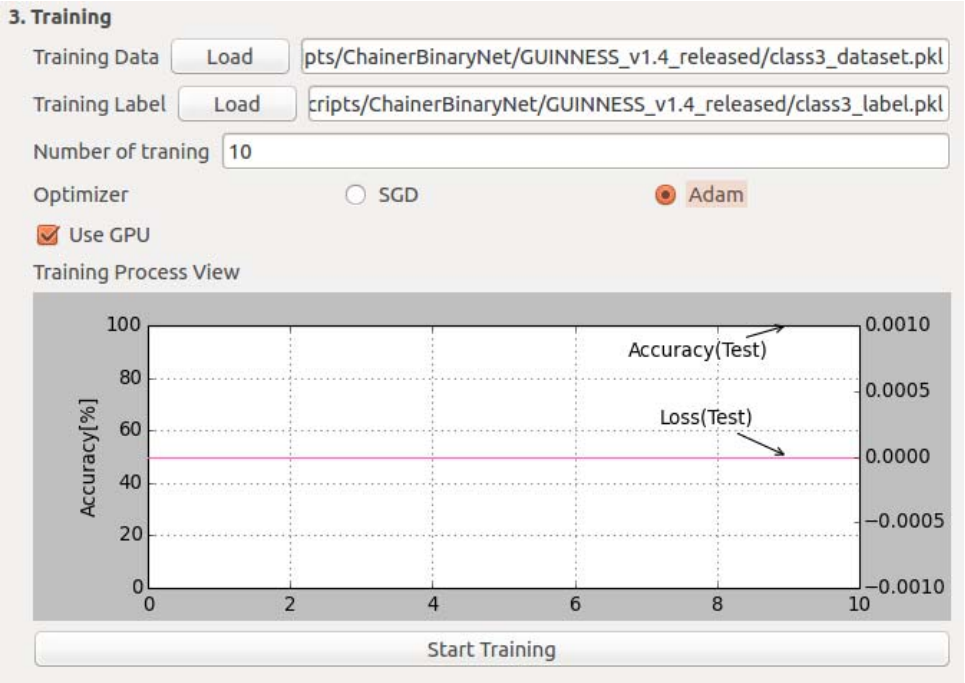


Fig.3.7: Setup training parameters

Third, setup training parameters as shown in Fig.3.7. Click “Load” button for both “Training Data” and “Training Label”. Training and label data are generated by “gen_training_data.py” script. Next, change the training algorithm to “Adam”, and make sure the “Use GPU” button is checked.

Then, click “Start Training” button. The GUI generates the scripts to operate training for your CNN. You can check the training process both “Terminal” and “Training Process View”. This process takes a long time to complete. After finish the training, you can generate bitstream or continue training (this button appear after finish the training instead of “Start Training”). If you want to increase recognition accuracy, then change the training parameters and click “Continue Training” button.

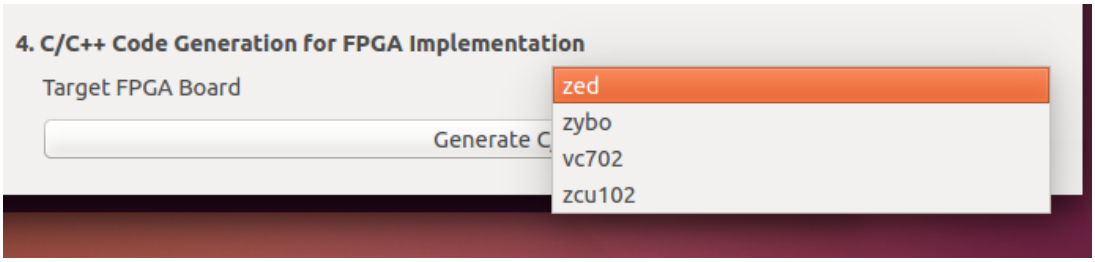
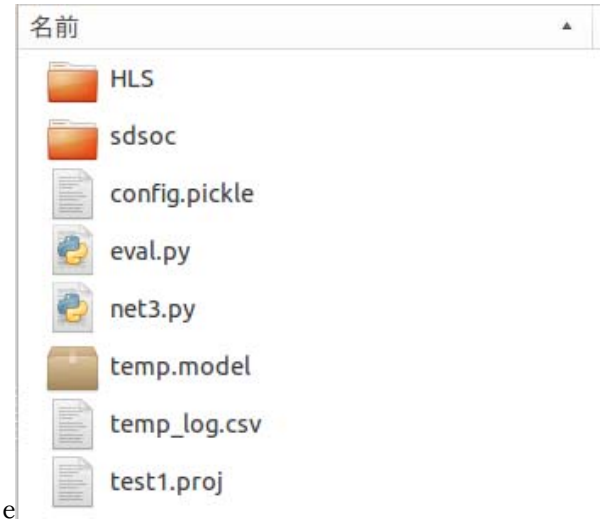


Fig.3.8: Specify target FPGA board

Finally, generate files to synthesise your CNN for an FPGA. Since the GUINNESS intends to use the Xilinx Inc. SDSoC tool, the target FPGA board must be specified. In the tutorial, we specify “Zed” board as shown in Fig.3.8. However, the target “TinyCNN” is a very small CNN, the Zybo board may be able to realize the CNN. Then, click “Generate C/C++ code”. Many codes are generated under your project folder. Click “SAVE” button in “Project Setup” part. So, you can restore your design in the GUINNESS.



e

Fig.3.9: (Project name) folder

Under the (project name) folder as shown in Fig.3.9, following files/folders are generated.

HLS folder ... csv file and header file to your project (not used in the FPGA synthesis)
sdsoc folder ... used in the SDSoC synthesis.
config.pickle ... configuration file to maintain your CNN.
eval.py ... verification script used in the next section
net3.py ... Your CNN specification used in the verification (not used in the SDSoC)
temp.model ... trained parameters for the Chainer framework. Note that, it is a numpy dataset, you can access raw weight and bias. However, the GUINNESS has already such an access tool.

temp_log.csv ... temporary training accuracy and loss to illustrate the training view.
test1.proj ... configuration file to maintain the project. You can restore the project by clicking project "LOAD" button, and specify this file.

4 Verification (Optional)

4.1 Python code verification

4.2 C++ simulation by Xilinx Inc. Vivado HLS

5 FPGA Synthesis using Xilinx Inc. SDSoC

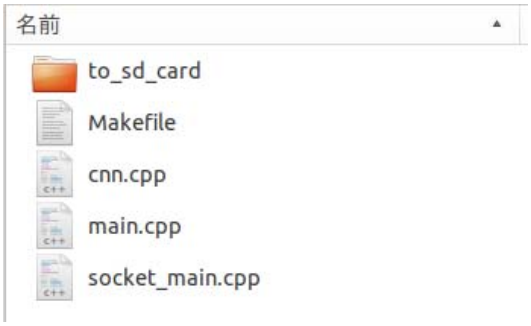


Fig.5.1: "(project name)/sdsoc" folder

Now, you can synthesize your CNN to the target FPGA board. First, you set the PATH environment variable with the following command.

```
# source /opt/Xilinx/SDx/2016.4/settings64.sh
```

Note that, you must change to the super user since the SDSoC requires it. After that, you can synthesis in the GUI mode of the SDSoC. In that case, you launch the SDSoC GUI, then, load "cnn.cpp" and "socket_main.cpp" shown in Fig.5.1. In the tutorial, we set the project name "guinness_tut1_v1".

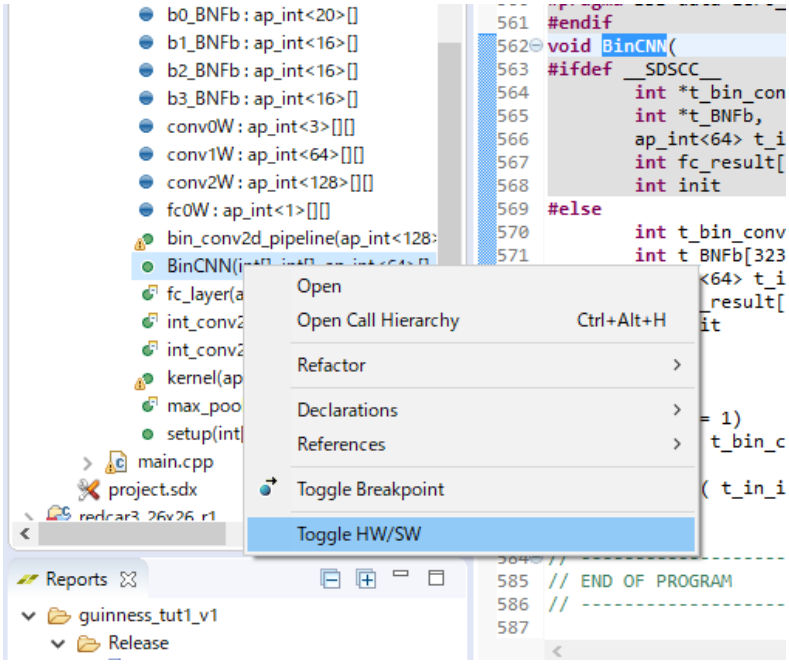


Fig. 5.2: Specify “BinCNN” as a HW function

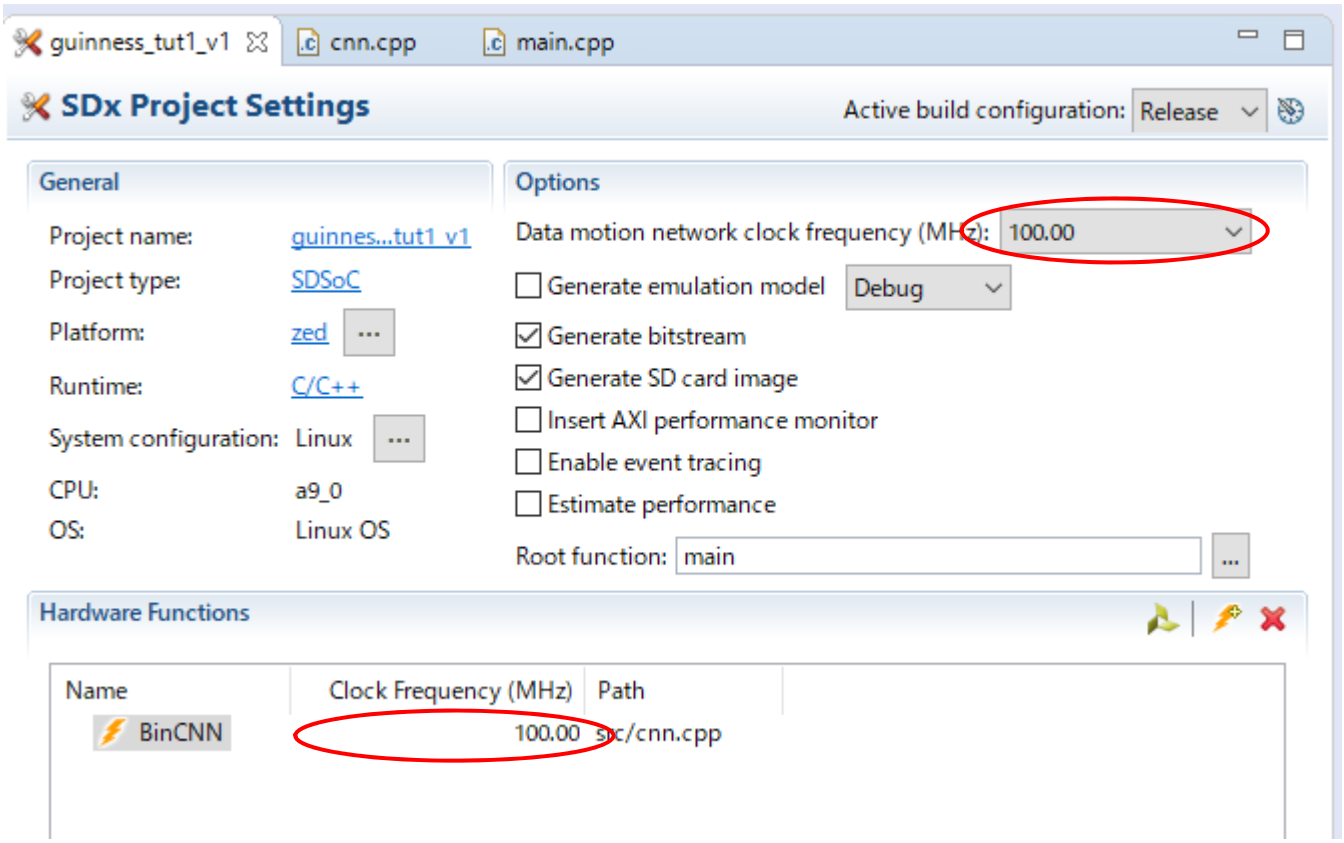


Fig.5.3: Setup target clock frequency as 100.0 MHz

Before the HW/SW co-synthesis, specify the HW function as shown in Fig.5.2, and setup target clock frequency as 100.0 MHz. Then, build the project.

This process requires about 40 minutes for my environment.

6 Run the image classification using a web camera

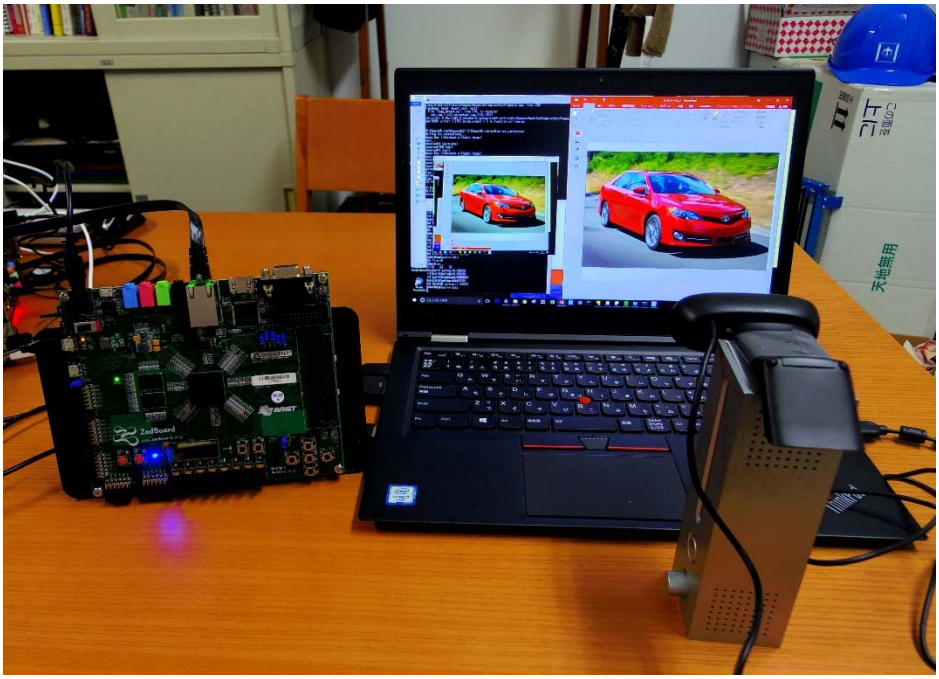


Fig.6.1: Image classification demo using a web camera

After finish the SDSoC compilation, you store weight and bias files under “(project name)/sdsoc/to_sd_card/” folder to the SD card. Also, generated files from the SDSoC are also transferred to the SD card. Connect the Zed board to the host PC via the Ethernet as shown in Fig.6.1. Make sure, both platforms (zedboard and host PC) can communicate by using “ping” command. If you have a router, then it is a simple way to connect these ones.

In the Zedboard, move to “/mnt” folder,

```
(zedboard) # cd /mnt
```

then, run your synthesized CNN on the Zedboard by the following command.

```
(zedboard) # ./guinness_tut1_v1.elf (host PC IP address) 10050
```

“10050” denotes the port number used in this application. In the host PC, download python script from https://github.com/HirokiNakahara/GUINNESS-Tutorial/blob/master/cnn_capture.py

Then, run the tutorial script on the host PC with the following command.

```
$ python cnn_capture.py --ip (host PC IP address) --size (image size) --tag (tag file name)
```

In the tutorial, the image size is “48” and the tag file name is “class3_tag.txt”. After type above command, an image view window appears as shown in Fig.6.2. You type “space” key to perform classification at once, and the classification result appears in the window. Also, type “-” key, then, a continue classification starts. Type “-” again, then, terminate classification. “ESC” key finish the application.

Fig.6.2: Image classification view

7 Re-training your CNN to improve recognition accuracy

The GUINNESS supports a re-training your CNN. In the case, you must not re-synthesize your CNN hardware, just only re-loading your re-trained weights and bias. First, run the GUINNESS GUI with the following command,

```
$ python guinness.py
```

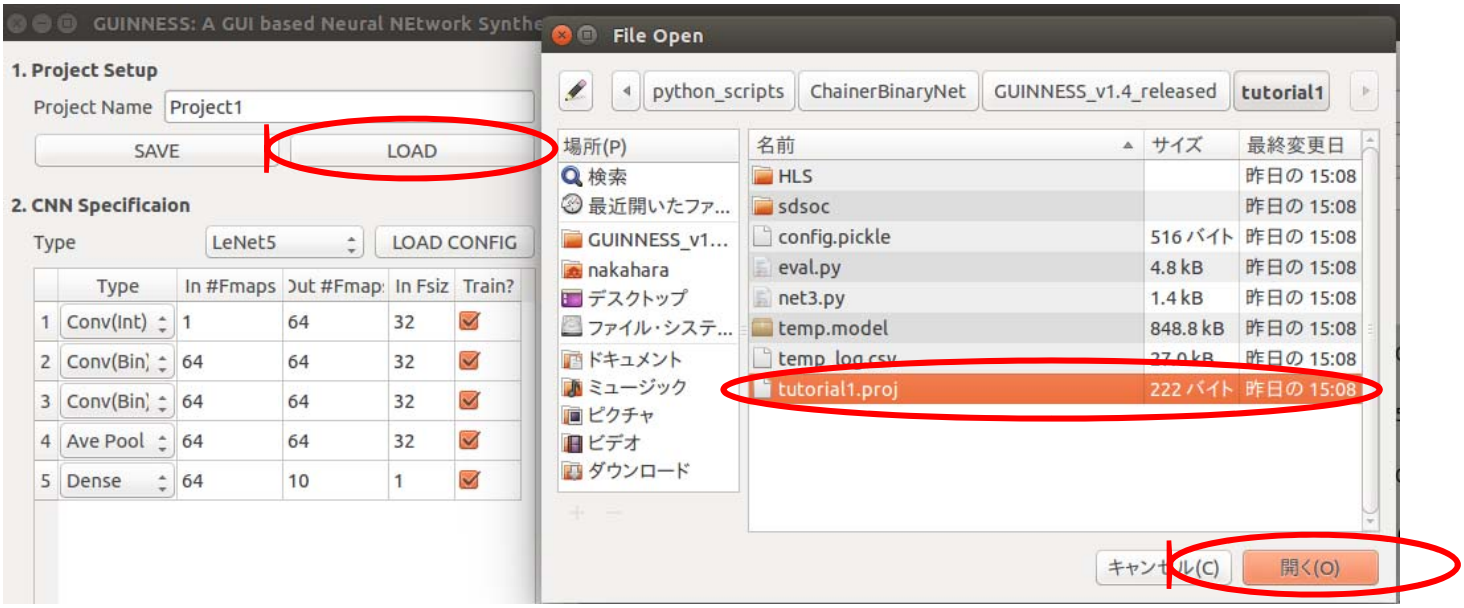


Fig. 7.1: Load the past project (*.proj) file

Then, the GUI appears as shown in Fig. 3.1. Restore your past project by clicking “LOAD” project button, then specify the project name (in the tutorial, /tutorial1/tutorial1.proj) as shown in Fig.7.1. You can execute more training to the pre-trained CNN. Set the “Number of training” value to “300”. Click “Continue Training” button, then training starts with a long time (about 20 min by using GTX1060 GPU). After finish the re-training, recognition accuracy is improved as shown in Fig. 7.2. Then, re-generate weights and bias by clicking “Generate

C/C++ Code". Transfer the weight files (located in /tutorial1/sdsoc/to_sd_card folder) to the SD card. You can see the recognition accuracy is improved!

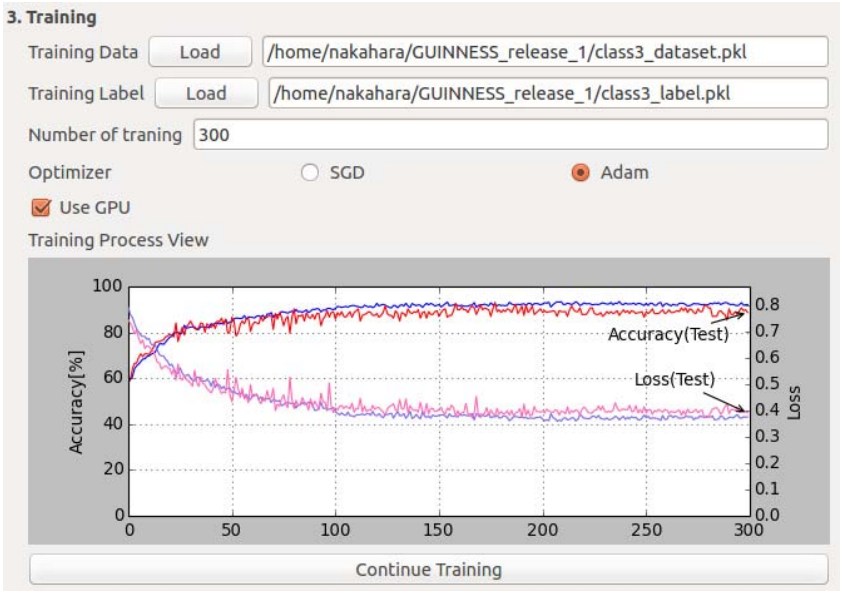


Fig. 7.2: Recognition accuracy achieved to around 90% after 300 epochs

8 Version History

03/Aug./2017 Released version (0.1)
06/Aug./2017 Released version (0.2): Adding the re-training (Chapter 7)