ECP5 Object Counting Using Resnet Quick Start Guide

Application Note

FPGA-AN-02021-1.0

May 2020

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS and with all faults, and all risk associated with such information is entirely with Buyer. Buyer shall not rely on any data and performance specifications or parameters provided herein. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. No Lattice products should be used in conjunction with mission- or safety-critical or any other application in which the failure of Lattice's product could create a situation where personal injury, death, severe property or environmental damage may occur. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Contents

Figures

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
FPGA	Field-Programmable Gate Array
VIP	Video Interface Platform

1. Introduction

This document provides a quick guide on how to train a machine and create a frozen file for the Lattice Machine Learning development using the Lattice's Embedded Vision Development Kit. It assumes that you are familiar with the basic Lattice FPGA design flow and mainly focuses on the Machine Learning part of the overall development process. For detailed instructions of the design flow described in this document, refer to the Object Counting Using Resnet CNN Accelerator IP (FPGA-RD-02195).

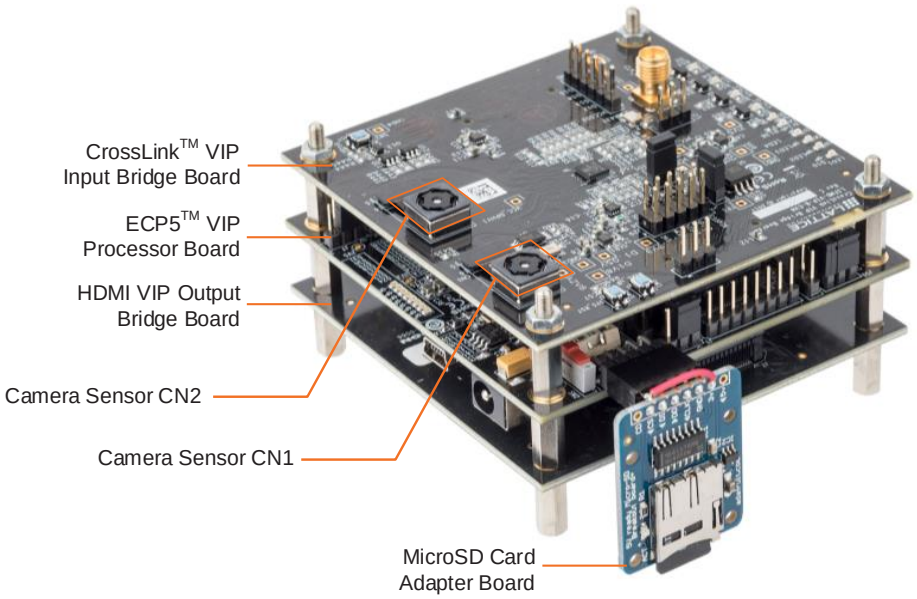

Figure 1.1. Lattice EVDK with MicroSD Card Adapter Board

1.1. Design Process Overview

The design process involves the following steps:

1. Setting up the basic environment
2. Preparing the dataset
3. Training the machine
4. Creating the frozen file (*.pb)
5. Creating the binary file with Lattice sensAI™ 3.0 program
6. Programming the binary and bitstream files to VIP board and SD card

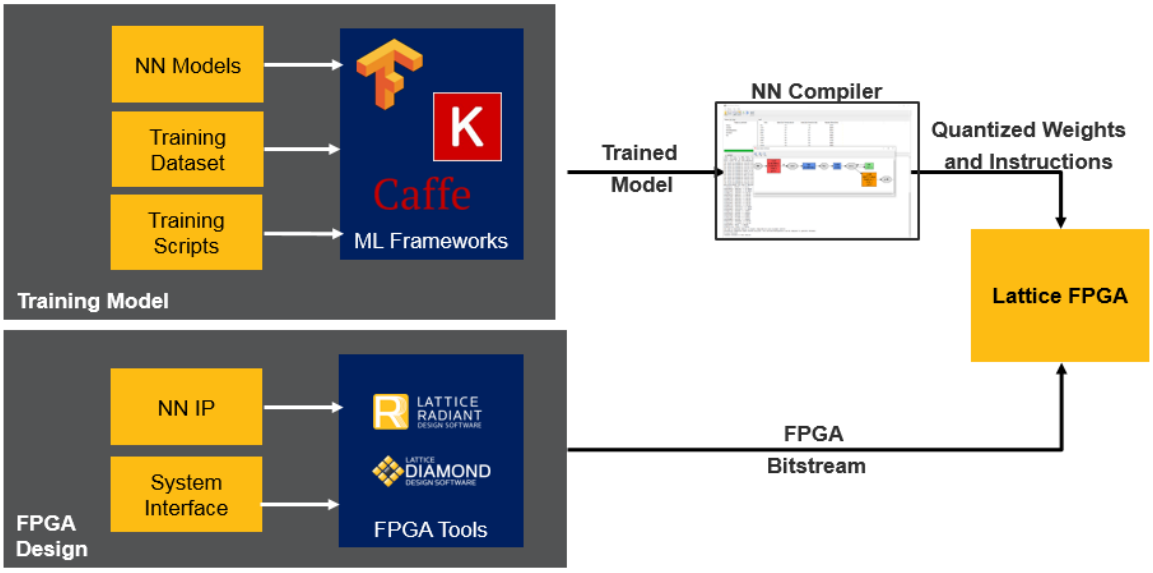

Figure 1.2. Lattice Machine Learning Design Flow

2. Machine Training and Creating Frozen File

2.1. Verifying TensorFlow and Tool Environment

Check if TensorFlow and your tool environment are installed correctly. For the detailed procedure in creating the basic environment on PC, refer to the Setting up the Basic Environment section in Object Counting Using Resnet CNN Accelerator IP (FPGA-RD-02195).

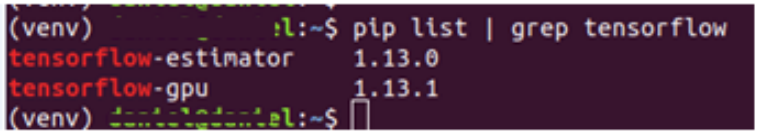

Figure 2.1. TensorFlow Installation Check

2.2. Preparing the Dataset

Prepare the image and label data (KITTI format). The image size for this design is 224 x 224 pixels.

For the detailed procedure in preparing the dataset, refer to the Preparing the Dataset section in Object Counting Using Resnet CNN Accelerator IP (FPGA-RD-02195).

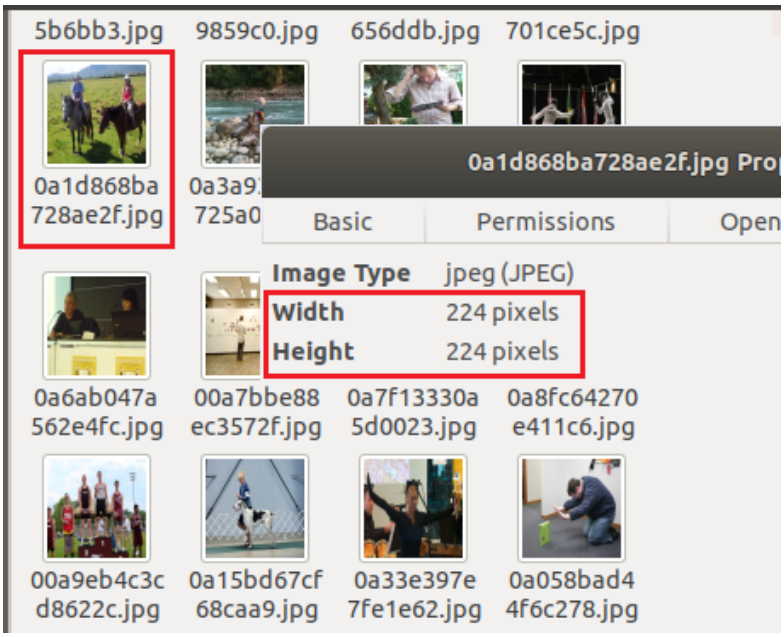

Figure 2.2. Dataset Image Size Check

2.3. Training the Machine

For the detailed procedure in machine training, refer to the Training the Machine section in Object Counting Using Resnet CNN Accelerator IP (FPGA-RD-02195).

To train the machine:

1. Check the training dataset path in the training script file *train.sh*.

```
export GPUID=1
export NET="squeezeDet"
export TRAIN_DIR="./logs"

export TRAIN_DATA_DIR="/data/humancnt/"
```

Figure 2.3. Dataset Folder Path Check

The subdataset path can be set in the training code *@src/dataset/kitti.py* and can be used in combination with the *--data_path* option while triggering training using *train.py* to get the desired path. For example, you can have *<data_path>/training/images* and *<data_path>/training/labels*.

```
def __init__(self, image_set, data_path, mc):
  imdb.__init__(self, 'kitti_'+image_set, mc)
  self._image_set = image_set
  self._data_root_path = data_path
  self._image_path = os.path.join(self._data_root_path, 'training', 'images')
  self._label_path = os.path.join(self._data_root_path, 'training', 'labels')
  self._classes = self.mc.CLASS_NAMES
```

Figure 2.4. Dataset List, Image, and Label Data Path

2. Create a train.txt file.

```
$ cd data/humancnt/
$ python dataset_create.py
```

```
k$ python dataset_create.py
k$ _
```

Figure 2.5. Create a Label File

3. Run machine training. In the command prompt, execute *./run* command.

```
earth:$ ./run
self.preds: Tensor("conv12/bias_add:0", shape=(20, 14, 14, 42), dtype=float32, device=/device:GPU:0)
ANCHOR_PER_GRID: 7
CLASSES: 1
ANCHORS: 1372
max person: 22
Model statistics saved to ./logs/humancnt/train/model_metrics.txt.
name: GeForce RTX 2080 Ti major: 7 minor: 5 memoryClockRate(GHz): 1.755
pciBusID: 0000:01:00.0
totalMemory: 10.73GiB freeMemory: 10.34GiB
2019-09-16 14:55:32.019017: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1512] Adding visible gpu devices: 0
2019-09-16 14:55:35.451832: I tensorflow/stream_executor/dso_loader.cc:152] successfully opened CUDA library libcublas.so.10.0 locally
conf_loss: 26.6821231842041, bbox_loss: 12.065683364868164, class_loss: 0.0
2019-09-16 14:55:36.940385: step 0, loss = 38.75 (5.8 images/sec; 3.466 sec/batch)
2019-09-16 14:55:39.423756: step 10, loss = 7.28 (240.5 images/sec; 0.083 sec/batch)
2019-09-16 14:55:40.269647: step 20, loss = 2.74 (241.6 images/sec; 0.083 sec/batch)
2019-09-16 14:55:41.106887: step 30, loss = 2.11 (239.2 images/sec; 0.084 sec/batch)
2019-09-16 14:55:41.949579: step 40, loss = 2.21 (241.7 images/sec; 0.083 sec/batch)
2019-09-16 14:55:42.786778: step 50, loss = 1.87 (241.1 images/sec; 0.083 sec/batch)
2019-09-16 14:55:43.613277: step 60, loss = 2.03 (241.3 images/sec; 0.083 sec/batch)
2019-09-16 14:55:44.448544: step 70, loss = 1.96 (240.3 images/sec; 0.083 sec/batch)
2019-09-16 14:55:45.289586: step 80, loss = 1.93 (240.5 images/sec; 0.083 sec/batch)
2019-09-16 14:55:46.123831: step 90, loss = 2.19 (242.3 images/sec; 0.083 sec/batch)
conf_loss: 1.1395012140274048, bbox_loss: 1.0199577808380127, class_loss: 0.0
```

Figure 2.6. Execute the Script

4. Start TensorBoard.

```
$ tensorboard –logdir=<log directory of training>
```

For example: tensorboard –logdir='./logs/'

5. Open the local host port on your web browser.

```
earth:$ tensorboard --logdir logs/humancnt/train
TensorBoard 1.12.0 at http://earth:6006 (Press CTRL+C to quit)
```

Figure 2.7. TensorBoard – Generated Link

6. Check the training status on TensorBoard.

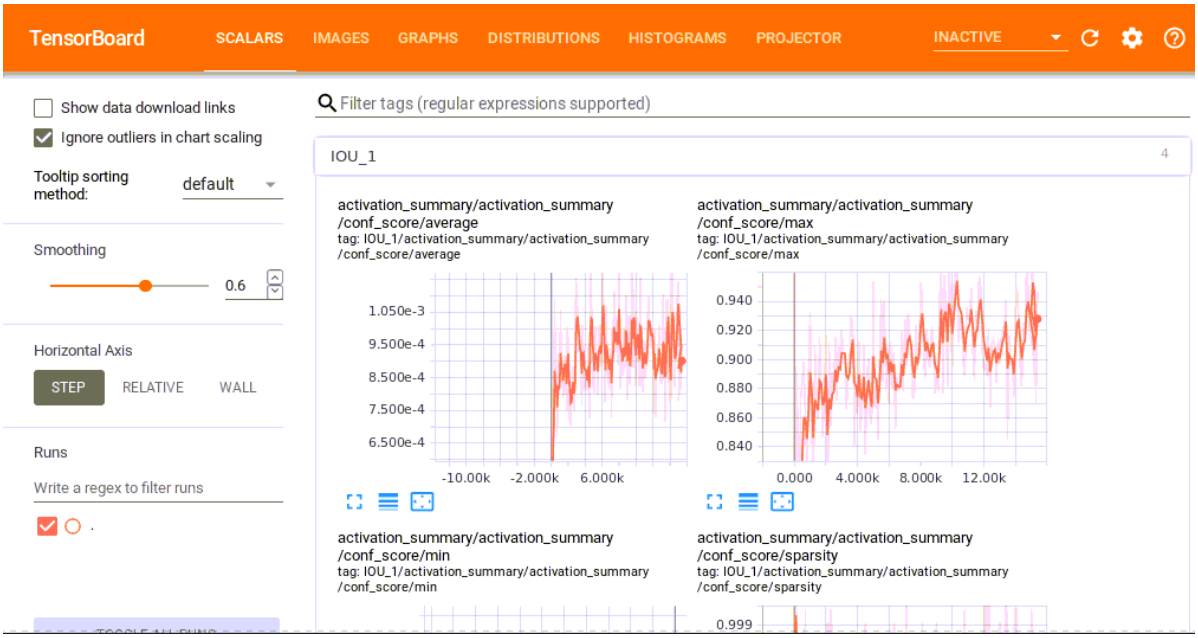

Figure 2.8. Training Status

Figure 2.9 shows the image menu of TensorBoard.

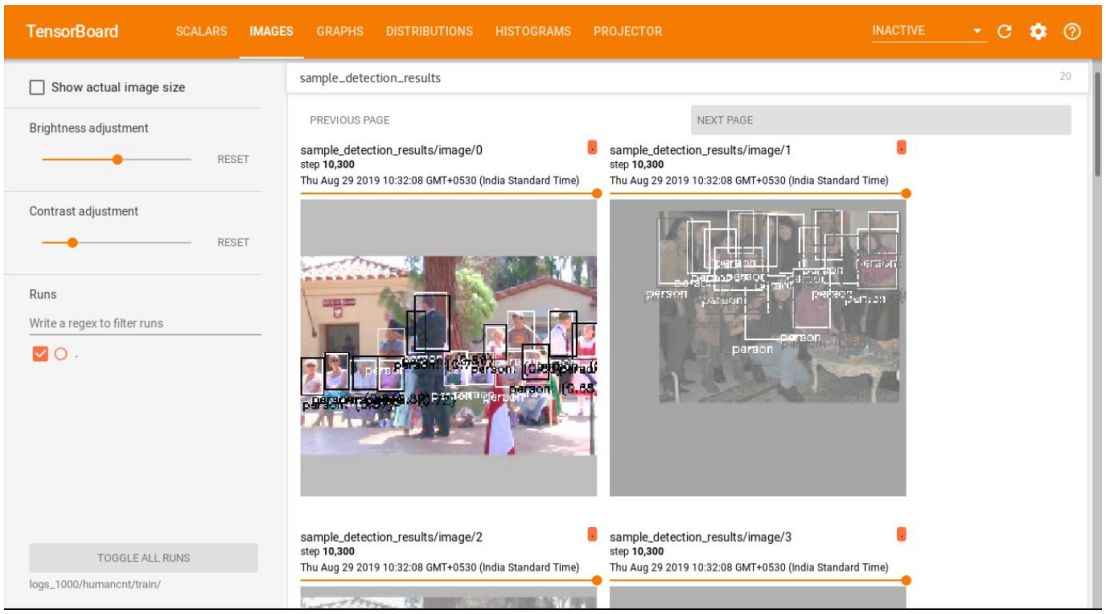

Figure 2.9. Image Menu

7. Check if the checkpoint, data, meta, index, and events (if using TensorBoard) files are created at the log directory. These files are used for creating the frozen file (*.pb).

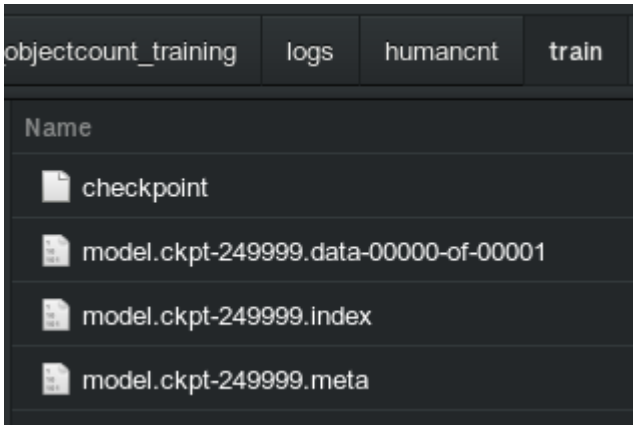

Figure 2.10. Checkpoint Data Files at Log Folder

2.4. Generating Frozen (*.pb) File

To generate frozen file (*.pb):

1. Run genpb.py.

   $ python src/genpb.py –ckpt_dir <log directory> --freeze

   For example, python src/genpb.py –ckpt_dir logs/humancnt/train --freeze

```
earth:$ python src/genpb.py --ckpt_dir logs/humancnt/train/ --freeze
genrating pbtxt
self.preds: Tensor("conv12/bias_add:0", shape=(20, 14, 14, 42), dtype=float32, device=/device:GPU:0)
ANCHOR_PER_GRID: 7
CLASSES: 1
ANCHORS: 1372
Using checkpoint: ./model.ckpt-249999
saved pbtxt at checkpoint direcory Path
inputShape shape [1, 224, 224, 3]
```

Figure 2.11. Create *.pbtxt File

2. Verify the generated frozen file.

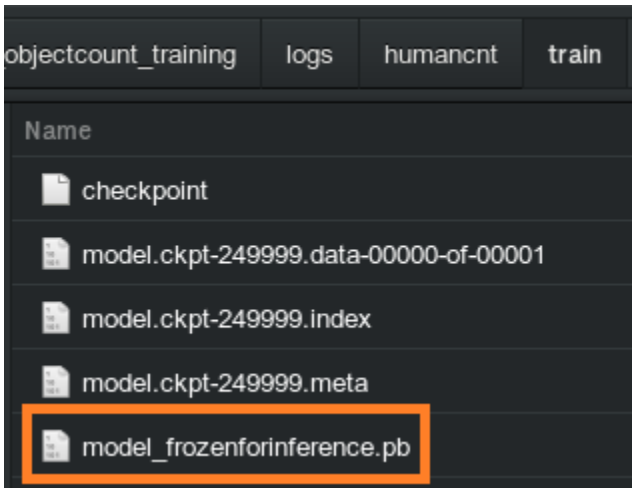

Figure 2.12. Check Frozen File

3. Generating the Binary File

For the detailed procedure in creating the binary file, refer to the Creating Binary File with sensAI section in Object Counting Using Resnet CNN Accelerator IP (FPGA-RD-02195).

4. Programming the Bitstream and Binary files to VIP Board and SD Card

For the detailed procedure in flashing the bitstream file to the VIP board and flashing the binary file to the SD card, refer to Object Counting Using Resnet CNN Accelerator IP (FPGA-RD-02195).

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

Revision History

Revision 1.0, May 2020

Section	Change Summary
All	Initial release.

www.latticesemi.com