ECP5 Face Identification

Reference Design

FPGA-RD-02062-1.0

August 2019

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

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
CKPT	Checkpoint
CNN	Convolutional Neural Network
EVDK	Embedded Vision Development Kit
FPGA	Field-Programmable Gate Array
LED	Light-emitting diode
ML	Machine Learning
MLE	Machine Learning Engine
NN	Neural Network
NNC	Neural Network Compiler
SD	Secure Digital
SDHC	Secure Digital High Capacity
SDXC	Secure Digital eXtended Capacity
SPI	Serial Peripheral Interface
VIP	Video Interface Platform
USB	Universal Serial Bus

1. Introduction

The Lattice ECP5 Face Identification document describes the Face Identification design process using the ECP5 EVDK FPGA platform.

1.1. Design Process Overview

1.1.1. Training Model

- Setting up the basic environment
- Preparing the dataset
    - Preparing 90 × 90 Image
    - Dataset augmentation
- Training the machine
    - Training the machine and creating the checkpoint or caffedata file
- Creating Frozen file (*.pb)

1.1.2. Neural Network Compiler

- Creating Binary file with Lattice sensAI2.0 program

1.1.3. FPGA Design

- Creating FPGA Bitstream file

1.1.4. FPGA Bitstream and Quantized Weights and Instructions

- Flashing Binary and Bitstream files
    - Binary File to MicroSD
    - Bitstream to Flash Memory on VIP Board

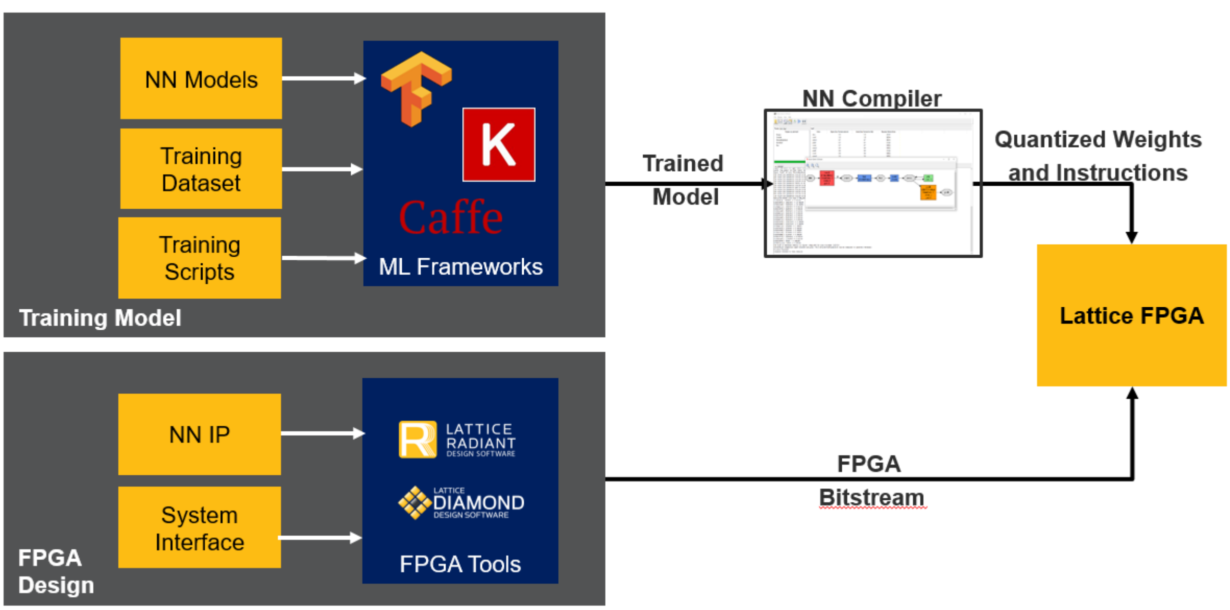


Figure 1.1. Lattice Machine Learning Design Flow

2. Setting Up the Basic Environment

2.1. Tools and Hardware Requirements

This section describes the required tools and environment setup for FPGA Bitstream and Flashing.

2.1.1. Lattice Tools

- Lattice Diamond® Tool – Refer to http://www.latticesemi.com/latticediamond.
- Lattice Diamond Programmer – Refer to http://www.latticesemi.com/programmer.
- Lattice SensAI Compiler v2.0 – Refer to https://www.latticesemi.com/Products/DesignSoftwareAndIP/AIML/NeuralNetworkCompiler.

2.1.2. Win32 MicroSD Disk Imager

Refer to https://sourceforge.net/projects/win32diskimager/.

2.1.3. Hardware

- ECP5 FPGA VIP Board – Refer to http://www.latticesemi.com/Solutions/Solutions/SolutionsDetails02/VIP.
- 1 × 4 membrane keypad (https://www.adafruit.com/product/1332)
- Hardware Rework

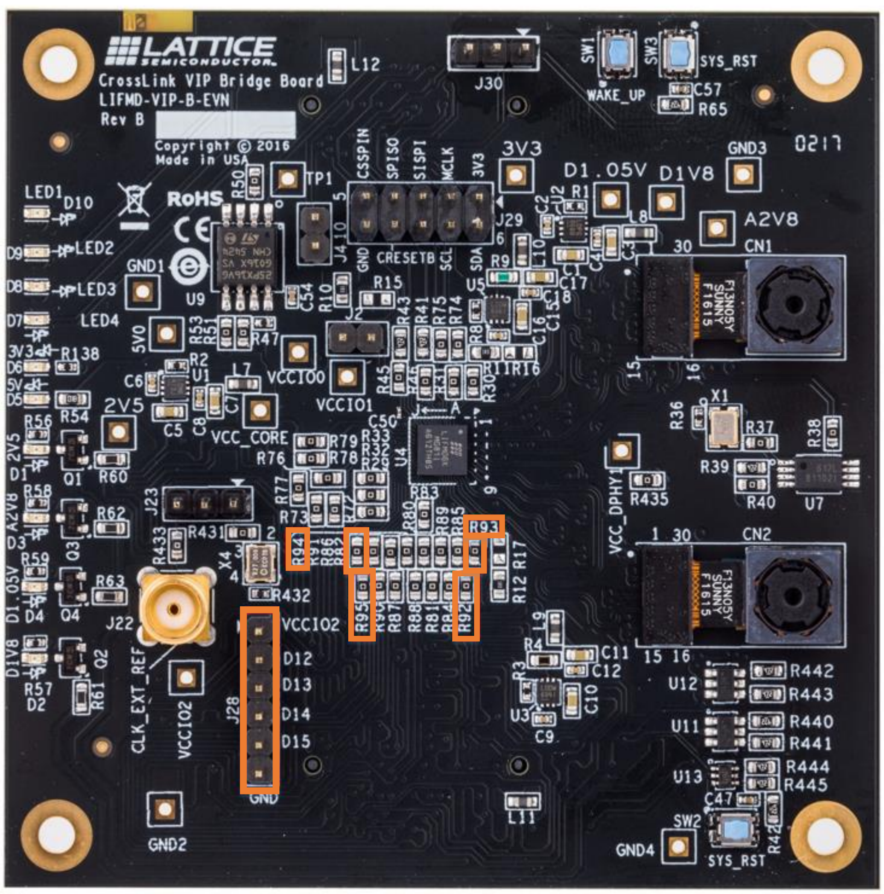


Figure 2.1. CrossLink VIP Input Bridge Board Hardware Rework

- Remove four zero-ohm resisters R92, R93, R94, and R95 and connect 1 × 4 keypad to J28.

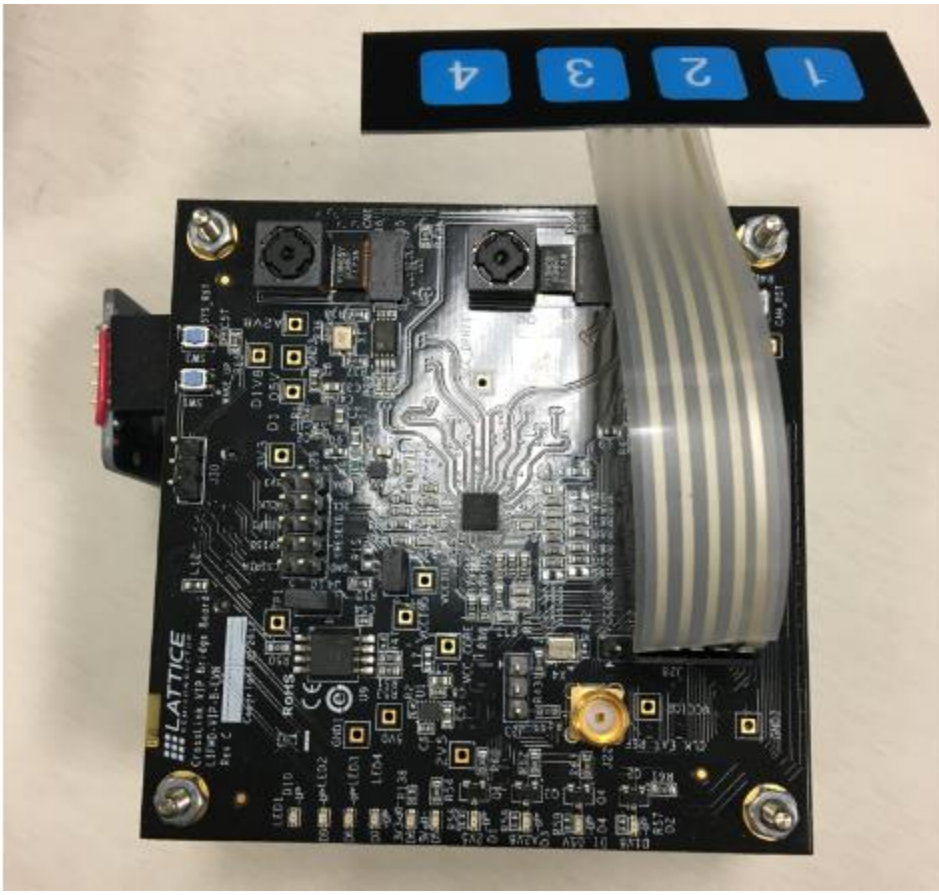


Figure 2.2. CrossLink VIP Input Bridge Board with 1 × 4 Keypad

3. Preparing the Dataset

This chapter describes how to create a dataset for training the Face Identification model.

3.1. Downloading the Dataset

To download the dataset:

1. Use the links below to download the training and testing VGGFace2 dataset:
    - Training dataset – http://zeus.robots.ox.ac.uk/vgg_face2/get_file?fname=vggface2_train.tar.gz
    - Testing dataset – http://zeus.robots.ox.ac.uk/vgg_face2/get_file?fname=vggface2_test.tar.gz
2. To access the dataset, you need to login. You have to sign up if you do not have an account by filling the required details on the given site.

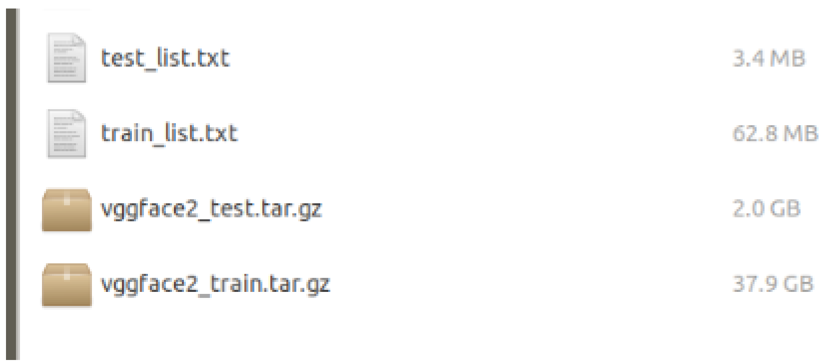


Figure 3.1. VGGFace2 Dataset

3.2. Augmenting the Dataset

To augment the dataset:

1. Run the dataset augmentation script file *augmentation_crop_and_canvas.py*.
    - Command help option:
        - -h 'python augmentation_crop_and_canvas.py –h'
    - Argumentation options:
        - -i – Input directory location
        - -o – Output directory location
        - -op (operation)
            - Canvas – Create canvas and place image in it.
            - Crop – Crop image and make it square of size minimum of height and width.
            - Both – To apply both operations, canvas and crop on 50% images each.
        - -cw (canvas width) – If given, canvas is generated of given width. Otherwise, ignored if -cr argument is passed.
        - -ch (canvas height) – If given, canvas is generated of given height. Otherwise, ignored if -cr argument is passed.
        - -cr (canvas ratio) – If given, canvas is generated of given % of maximum out of height and width.
        - -cl: (canvas location) – If ture, image is placed at random location in canvas.
        - -ns: (name suffix) – If given, name suffix is added in generated images.

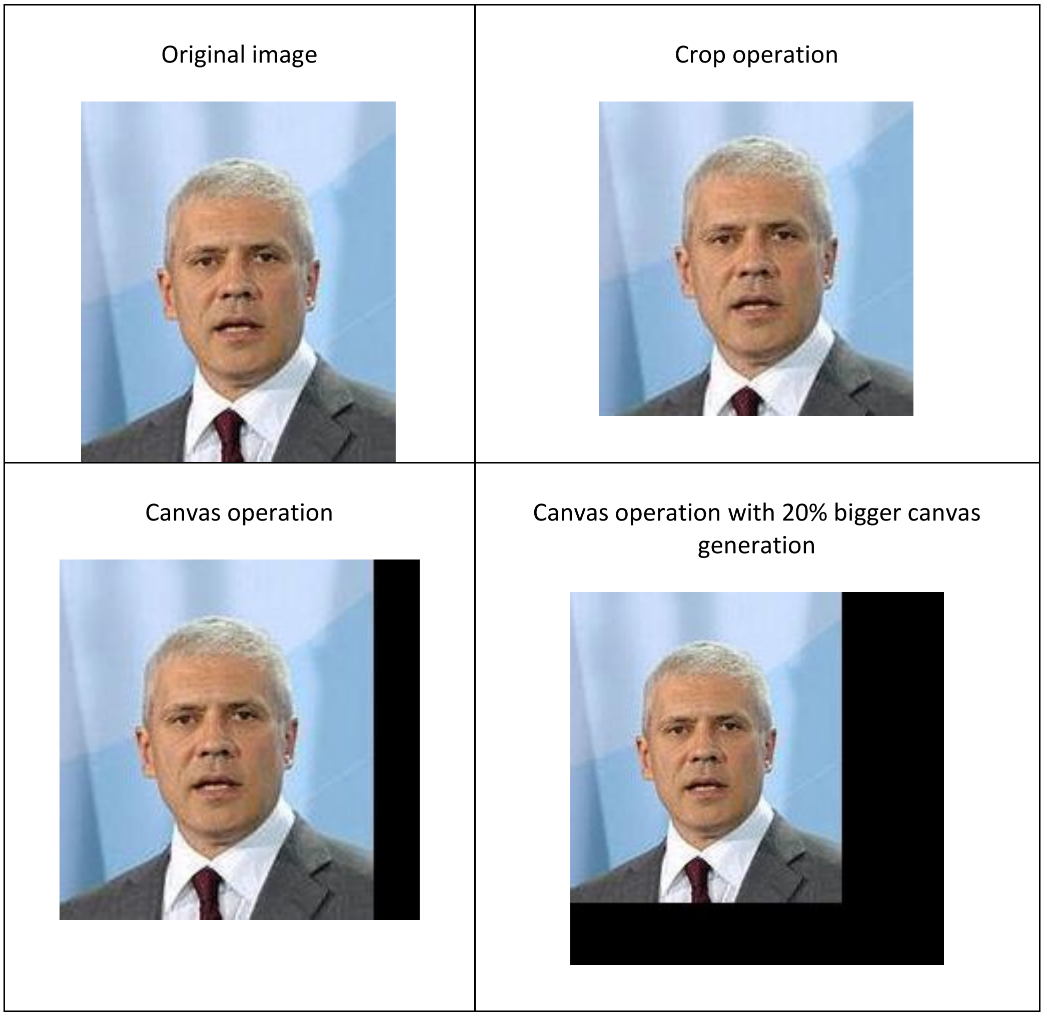


Figure 3.2. Dataset Augmentation Example

2. Generate square images out of original training and testing dataset by crop and pad operation for alternate classes by running the scripts below:

```
$ python augmentation_crop_and_canvas.py -
i ./dataset/vggface2_train/train/o ./dataset/vggface2_train/train/ -op both

$ python augmentation_crop_and_canvas.py -i ./dataset/vggface2_test/test/ -
o ./dataset/vggface2_test/test/ -op both
```

3. Generate additional images using canvas augmentation with 20% larger canvas and random positioning of the original image with image name being post fixed with *canvas* by running the scripts below:

```
$ python augmentation_crop_and_canvas.py -i ./dataset/vggface2_train/train/ -
o ./dataset/vggface2_train/train/ -op canvas -ns canvas -cr 20 -cl true
$ python augmentation_crop_and_canvas.py -i ./dataset/vggface2_test/test/ -
o ./dataset/vggface2_test/test/ -op canvas -ns canvas -cr 20 -cl true
```

3.3. Annotations and Adding Class-Number to Annotation File

1. Run the scripts below to generate the annotation script file.

   Note: This *generate_train_test_list.py* script creates two files: *train.txt* for training dataset and *test.txt* for testing dataset, where training dataset is 80% of the total images and testing dataset is 20% of the images.

```
$python generate_train_test_list.py --input_dir=/dataset/vggface2_train/train

$python generate_train_test_list.py --input_dir=/dataset/vggface2_test/test
```

2. Run the scripts below to add class number.

```
$python modify-vgg2-face-label.py --input_file=train.txt
```

   Note: The above script generates the *train_new.txt* which can be used as source of training dataset in proto file. Note that text file is renamed to match the name entry in proto file. If you want to use your own file name, you need to update the proto file accordingly.

```
$python modify-vgg2-face-label.py --input_file=test.txt
```

   Note: The above script generates the *test_new.txt* which can be used as source of testing dataset in proto file. Note that text file is renamed to match the name entry in proto file. If you want to use your own file name, you need to update the proto file accordingly.

4. Training the Machine

4.1. Training the Machine with Caffe Machine Learning Platform

4.1.1. Caffe Code Structure

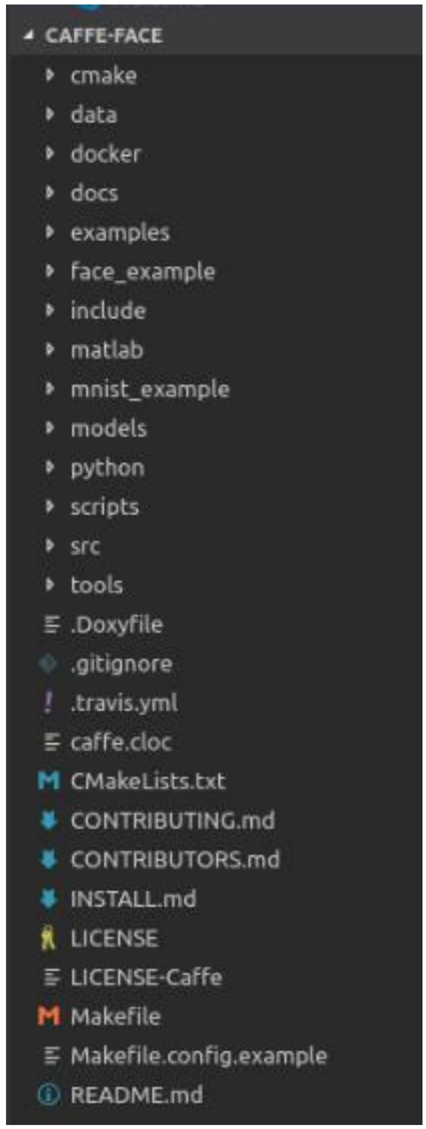


Figure 4.1 Caffe Code Directory Structure

4.1.2. Installing and Compiling the Caffe OpenSource

For instructions on how to install the Caffe Open Source, refer to https://caffe.berkeleyvision.org/installation.html.

4.1.3. Dataset Augmentation and Generating Label Data with VGGFace2 Images

To augment dataset and generate label data:

1. Create a Face-Identification dataset folder in the *data* folder.

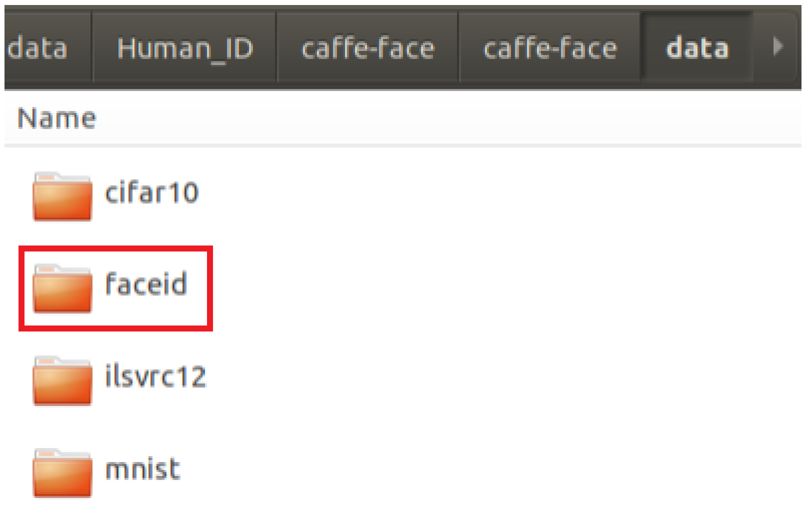


Figure 4.2 Data Set Folder in Caffe Structure

2. Copy the data augmentation and annotation script files to the dataset folder.
3. Copy the VGGFace2 images to the dataset folder.

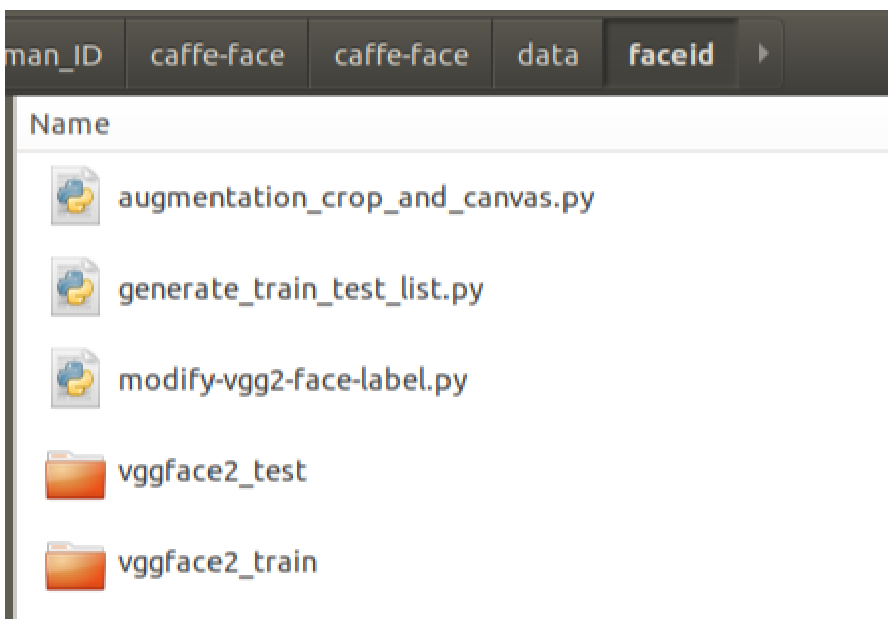


Figure 4.3 Face Identification Dataset Folder Content Example

4. Run the dataset augmentation and annotation script files.

```
daniel@daniel:~/data/Human_ID/caffe-face/caffe-face/data/faceid$
python augmentation_crop_and_canvas.py -i ./vggface2_test/test/ -
o ./vggface2_test/test/ -op both
daniel@daniel:~/data/Human_ID/caffe-face/caffe-face/data/faceid$
python augmentation_crop_and_canvas.py -i ./vggface2_train/train/
 -o ./vggface2_train/train/ -op both
```

Figure 4.4 Generating Square image Dataset Script Command Example

```
daniel@daniel:~/data/Human_ID/caffe-face/caffe-face/data/faceid$
python augmentation_crop_and_canvas.py -i ./vggface2_test/test/ -
o ./vggface2_test/test/ -op canvas -ns canvas -cr 20 -cl true
```

```
daniel@daniel:~/data/Human_ID/caffe-face/caffe-face/data/faceid$
python augmentation_crop_and_canvas.py -i ./vggface2_train/train/
 -o ./vggface2_train/train/ -op canvas -ns canvas -cr 20 -cl true
```

Figure 4.5 Dataset Augmentation Script Command Example

```
daniel@daniel:~/data/Human_ID/caffe-face/caffe-face/data/faceid$ python genera
te_train_test_list.py --input_dir=./vggface2_test/test/
daniel@daniel:~/data/Human_ID/caffe-face/caffe-face/data/faceid$ python genera
te_train_test_list.py --input_dir=./vggface2_train/train/
```

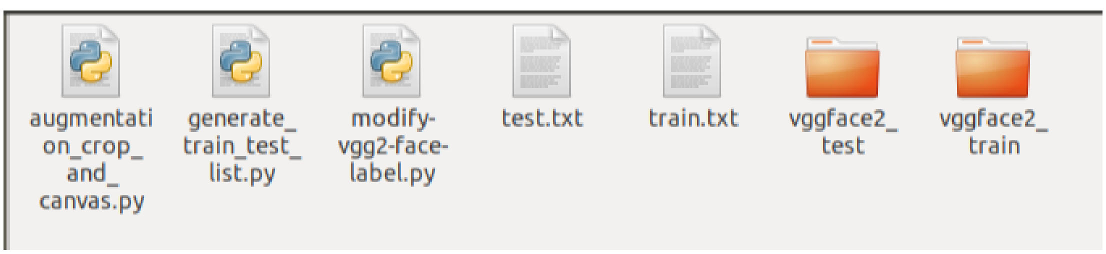


Figure 4.6 Annotation Script Command Example

```
daniel@daniel:~/data/Human_ID/caffe-face/caffe-face/data/faceid$ python modify
-vgg2-face-label.py --input_file=train.txt
Modified file created: train_new.txt
daniel@daniel:~/data/Human_ID/caffe-face/caffe-face/data/faceid$ python modify
-vgg2-face-label.py --input_file=test.txt
Modified file created: test_new.txt
```

Figure 4.7 Label Data Files Example

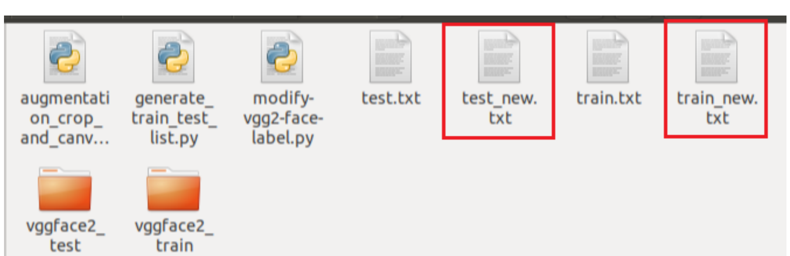


Figure 4.8 Label Script Command Example

4.1.4. Training the Face Identification

To train the face identification:

1. Check three script files in the *train* folder as shown in Figure 4.9.

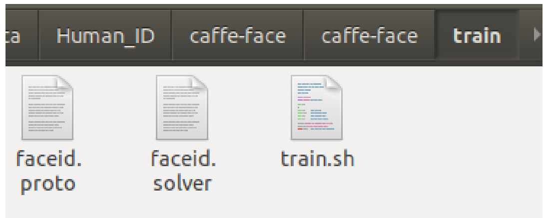


Figure 4.9 Default Content of the Train Folder

2. Copy the label files *train_new.txt* and *test_new.txt* to the *train* folder.

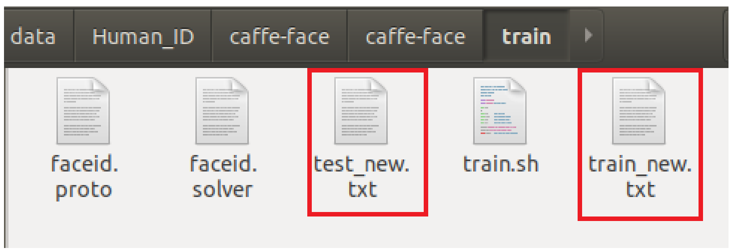


Figure 4.10 Machine Training Contents

3. Update the proto file, *faceid.proto* for Image dataset source.

```
name: "FaceID_vgg8"
layer {
  name: "train_data"
  type: "ImageData"
  top: "data"
  top: "label"
  transform_param {
    mirror: true
    #crop_size: 90
    mean_value: 128
    mean_value: 128
    mean_value: 128
    max_rotation_angle: 20
  }
  image_data_param {
    source: "train_new.txt"
    #batch_size: 128
    #batch_size: 64
    batch_size: 32
    #batch_size: 16
    new_height: 90
    new_width: 90
    shuffle: true
  }
  include { phase: TRAIN }
}
layer {
  name: "test_data"
  type: "ImageData"
  top: "data"
  top: "label"
  transform_param {
    mirror: false
    mean_value: 128
    mean_value: 128
    mean_value: 128
  }
  image_data_param {
    source: "test_new.txt"
    batch_size: 10
```

Figure 4.11 Update Proto File for Dataset Label File Name

4. Set the output number of FC6 layer.

   Set '*num_output*' to the number of dataset class.

   The training code of Face Identification is for the object classification as MNIST application. Therefore, the final output number of Fully-connected model is the number of object class. The default value of FC6 output is 9295.

```
  top: "fc6"
  exclude: { phase: TEST not_stage: "val" }
  param {
    lr_mult: 1.0
    decay_mult: 1.0
  }
  param {
    lr_mult: 2.0
    decay_mult: 0.0
  }
  inner_product_param {
    num_output: 9295
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
      value: 0.1
    }
  }
}
```

The number of dataset class

Figure 4.12 Last Fully-connected Layer Output Number Setting

5. Update the following variables in *train.sh* as per your environment setup.
   - CAFFE_ROOT – with compiled caffe location.
   - OPT – GPU number or comment out to use CPU only.

Open ▾ | train.sh ~/data/Human_ID/caffe-face/caffe-face/train | Save

```
################################################################################
#
################################################################################
SOLVER=faceid.solver
#WEIGHTS=fid_iter_100000.caffemodel
#SNAPSHOT=fid_iter_100000.solverstate

################################################################################
if [ -n "$SNAPSHOT" ]; then
  OPT=--snapshot=$SNAPSHOT
elif [ -n "$WEIGHTS" ]; then
  OPT=--weights=$WEIGHTS
fi
if [ "$CAFFE_ROOT" == "" ]; then
  #CAFFE_ROOT=$HOME/ML/caffe
fi
OPT="$OPT --gpu=0"
PYTHONPATH=$CAFFE_ROOT/python $CAFFE_ROOT/build/tools/caffe train --solver=$SOLVER $OPT |& tee -a
train.log
```

Figure 4.13 train.sh Parameter Setting Example

6. Run the *train.sh* script.

```
daniel@daniel:~/data/Human_ID/caffe-face/caffe-face/train$ ls
faceid.proto  faceid.solver  test_new.txt  train.log  train_new.txt  train.sh
daniel@daniel:~/data/Human_ID/caffe-face/caffe-face/train$ ./train.sh
```

Figure 4.14 Executing train.sh Script Command

```
I0802 09:45:46.532480  9499 net.cpp:228] test_data does not need backward computation.
I0802 09:45:46.532481  9499 net.cpp:270] This network produces output accuracy
I0802 09:45:46.532485  9499 net.cpp:270] This network produces output loss
I0802 09:45:46.532498  9499 net.cpp:283] Network initialization done.
I0802 09:45:46.532567  9499 solver.cpp:60] Solver scaffolding done.
I0802 09:45:46.534166  9499 caffe.cpp:251] Starting Optimization
I0802 09:45:46.534171  9499 solver.cpp:279] Solving FaceID_vgg8
I0802 09:45:46.534173  9499 solver.cpp:280] Learning Rate Policy: multistep
I0802 09:45:46.535856  9499 solver.cpp:337] Iteration 0, Testing net (#0)
I0802 09:45:46.535862  9499 net.cpp:693] Ignoring source layer train_data
I0802 09:45:46.535864  9499 net.cpp:693] Ignoring source layer label_train_data_1_split
I0802 09:45:46.539319  9499 net.cpp:693] Ignoring source layer faceid_drop5_0_split
I0802 09:45:46.540297  9499 net.cpp:693] Ignoring source layer center_loss
I0802 09:45:54.777025  9499 solver.cpp:404]     Test net output #0: accuracy = 0
I0802 09:45:54.777060  9499 solver.cpp:404]     Test net output #1: loss = 87.3362 (* 1 = 87.3362 loss)
I0802 09:45:54.956874  9499 solver.cpp:228] Iteration 0, loss = 12.2869
I0802 09:45:54.956912  9499 solver.cpp:244]     Train net output #0: center_loss = 245.702 (* 0.008 = 1.96561 loss
)
I0802 09:45:54.956918  9499 solver.cpp:244]     Train net output #1: loss = 10.3213 (* 1 = 10.3213 loss)
I0802 09:45:54.956944  9499 sgd_solver.cpp:106] Iteration 0, lr = 0.0001
I0802 09:46:12.322404  9499 solver.cpp:228] Iteration 100, loss = 10.2861
I0802 09:46:12.322432  9499 solver.cpp:244]     Train net output #0: center_loss = 55.6718 (* 0.008 = 0.445375 los
s)
I0802 09:46:12.322438  9499 solver.cpp:244]     Train net output #1: loss = 9.39769 (* 1 = 9.39769 loss)
I0802 09:46:12.322441  9499 sgd_solver.cpp:106] Iteration 100, lr = 0.0001
```

Figure 4.15 Training Status Example

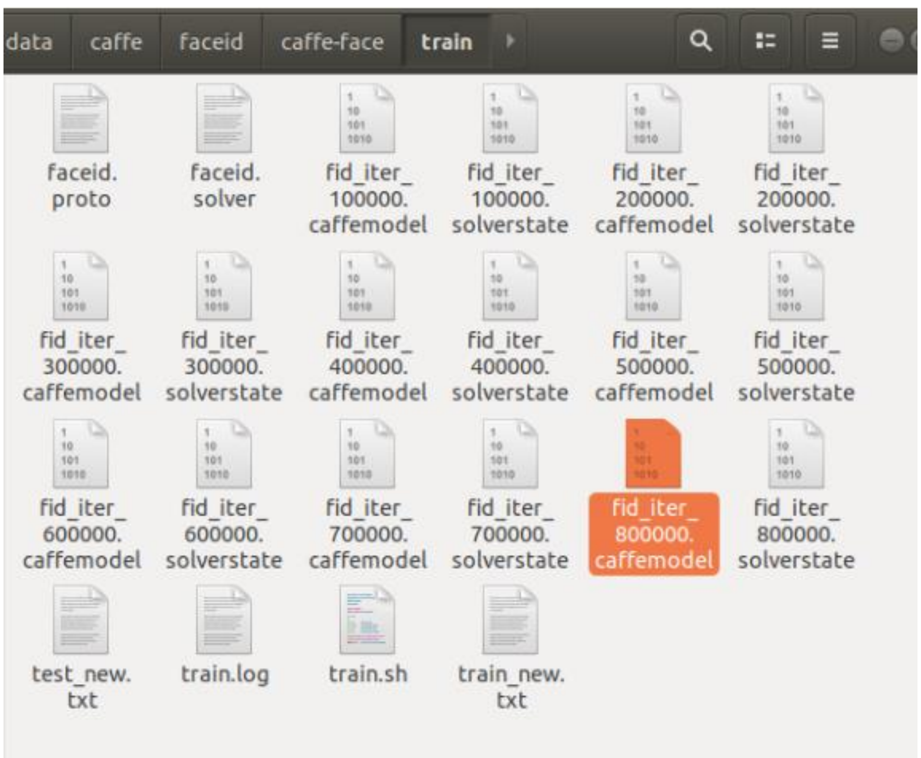


Figure 4.16 Machine Training Output File

4.1.5. Neural Network Architecture

4.1.5.1. Face Identification Training Neural Network

Table 4.1 describes the Face Identification Training Neural Network model.

Table 4.1. Face Identification Training Neural Network Model

<table>
<tr><th colspan="3">Image Input (90 × 90)</th></tr>
<tr><td rowspan="5">Layer 1</td><td>Conv3 - 96</td><td rowspan="38">Conv3 - # where:
• Conv3 – 3 × 3 Convolution filter Kernel size
• # - The number of filter
For example, Conv3 - 96 = 96 3 × 3 convolution filter
BN – Batch Normalization
FC - # where:
• FC – Fully connected layer
• # - The number of output
<Face Class Number>:
The default number of output is 9295 in the proto file.
The number has to be updated according to Dataset class number.</td></tr>
<tr><td>BN</td></tr>
<tr><td>Scale</td></tr>
<tr><td>Relu</td></tr>
<tr><td>Maxpool</td></tr>
<tr><td rowspan="4">Layer 21</td><td>Conv3 - 192</td></tr>
<tr><td>BN</td></tr>
<tr><td>Scale</td></tr>
<tr><td>Relu</td></tr>
<tr><td rowspan="4">Layer 22</td><td>Conv3 - 192</td></tr>
<tr><td>BN</td></tr>
<tr><td>Scale</td></tr>
<tr><td>Relu</td></tr>
<tr><td rowspan="5">Layer 23</td><td>Conv3 - 192</td></tr>
<tr><td>BN</td></tr>
<tr><td>Scale</td></tr>
<tr><td>Relu</td></tr>
<tr><td>Maxpool</td></tr>
<tr><td rowspan="4">Layer 31</td><td>Conv3 - 288</td></tr>
<tr><td>BN</td></tr>
<tr><td>Scale</td></tr>
<tr><td>Relu</td></tr>
<tr><td rowspan="4">Layer 32</td><td>Conv3 - 288</td></tr>
<tr><td>BN</td></tr>
<tr><td>Scale</td></tr>
<tr><td>Relu</td></tr>
<tr><td rowspan="5">Layer 33</td><td>Conv3 - 288</td></tr>
<tr><td>BN</td></tr>
<tr><td>Scale</td></tr>
<tr><td>Relu</td></tr>
<tr><td>Maxpool</td></tr>
<tr><td rowspan="5">Layer 41</td><td>Conv3 - 384</td></tr>
<tr><td>BN</td></tr>
<tr><td>Scale</td></tr>
<tr><td>Relu</td></tr>
<tr><td>Maxpool</td></tr>
<tr><td>FC5</td><td>FC – 256</td></tr>
<tr><td>Drop5</td><td>Dropout</td></tr>
<tr><td>FC6</td><td>FC – <Face ClassNumber></td></tr>
</table>

Notes:

- For the last two layers, drop5 and FC6 are used in the training mode in order to implement the Predefined Object classification design. But these are not used in the inference mode because input image is an unknown Object classification.
- The output of FC6 is the number of classes in dataset and is used to calculate loss.
- The output of FC5 is 256-feature map which is used for Face Identification classification.
- Neural Network Configuration can be changed by modifying proto file.
- Refer to the Caffe tutorial https://caffe.berkeleyvision.org/tutorial/.

4.1.5.2. Face Identification Inference Neural Network

Table 4.2 describes the Face Identification Inference Neural Network model.

Table 4.2. Face Identification Inference Neural Network Model

<table>
<tr><th colspan="3">Image Input (90 × 90)</th></tr>
<tr><td rowspan="5">Layer 1</td><td>Conv3 - 96</td><td rowspan="37">Conv3 - # where:
• Conv3 – 3 × 3 Convolution filter Kernel size
• # - The number of filter

For example, Conv3 - 96 = 96 3 × 3 convolution filter

BN – Batch Normalization

FC - # where:
• FC – Fully connected layer
• # - The number of output</td></tr>
<tr><td>BN</td></tr>
<tr><td>Scale</td></tr>
<tr><td>Relu</td></tr>
<tr><td>Maxpool</td></tr>
<tr><td rowspan="4">Layer 21</td><td>Conv3 - 192</td></tr>
<tr><td>BN</td></tr>
<tr><td>Scale</td></tr>
<tr><td>Relu</td></tr>
<tr><td rowspan="4">Layer 22</td><td>Conv3 - 192</td></tr>
<tr><td>BN</td></tr>
<tr><td>Scale</td></tr>
<tr><td>Relu</td></tr>
<tr><td rowspan="5">Layer 23</td><td>Conv3 - 192</td></tr>
<tr><td>BN</td></tr>
<tr><td>Scale</td></tr>
<tr><td>Relu</td></tr>
<tr><td>Maxpool</td></tr>
<tr><td rowspan="4">Layer 31</td><td>Conv3 - 288</td></tr>
<tr><td>BN</td></tr>
<tr><td>Scale</td></tr>
<tr><td>Relu</td></tr>
<tr><td rowspan="4">Layer 32</td><td>Conv3 - 288</td></tr>
<tr><td>BN</td></tr>
<tr><td>Scale</td></tr>
<tr><td>Relu</td></tr>
<tr><td rowspan="5">Layer 33</td><td>Conv3 - 288</td></tr>
<tr><td>BN</td></tr>
<tr><td>Scale</td></tr>
<tr><td>Relu</td></tr>
<tr><td>Maxpool</td></tr>
<tr><td rowspan="5">Layer 41</td><td>Conv3 - 384</td></tr>
<tr><td>BN</td></tr>
<tr><td>Scale</td></tr>
<tr><td>Relu</td></tr>
<tr><td>Maxpool</td></tr>
<tr><td>FC5</td><td>FC – 256</td></tr>
</table>

Notes:

- The inference network mode is different from the training proto file. Inference model does not use *Drop5* and *FC6* layer.
- Remove train_data, test_data, drop5, loss, accuracy, fc6, and center_loss.
- Lattice provides the reference protofile of Inference model file *faceid_demo.proto*.
- Inference proto file is used in the part of generating binary file.

4.2. Training the Machine with TensorFlow

4.2.1. TensorFlow Code Structure

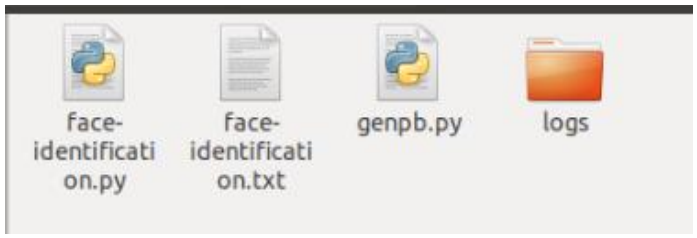


Figure 4.17 TensorFlow Training Script files

4.2.2. Training Code Overview

Training Code is divided into four parts:

- Model Building
- Model Freezing
- Data Preparation
- Training for Overall Execution Flow

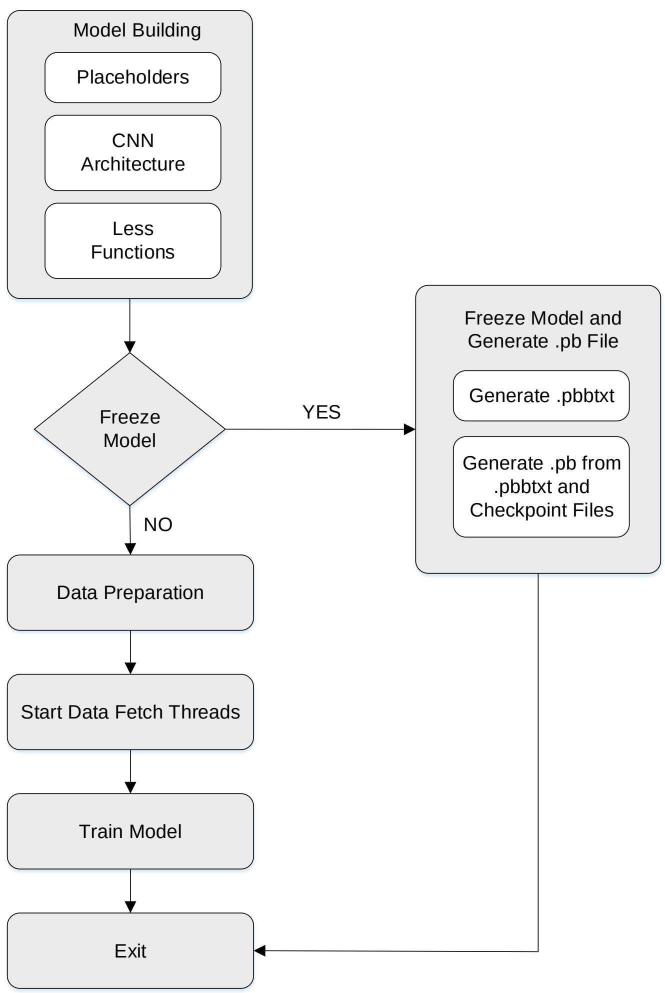


Figure 4.18 Face Identification Training Structure TensorFlow Code

4.2.2.1. Model Building

def get_model(num_classes, is_training) builds the graph which are divided in three sections: Placeholders, CNN Architecture, and Loss Functions.

- **Placeholders**
  - *images*, *labels,* and *learning_rate_placeholder* are used in this model, where *images* and *labels* are used to feed training data.
  - While creating *Images* placeholder, it can modify input dimensions of model by replacing proper values at IMG_WIDTH and IMG_HEIGHT.
  - *learning_rate_placeholder* is used to change learning rate of training, which gives flexibility to change learning rate on the go without stopping the training.

```
images = tf.placeholder(tf.float32, [None, IMG_WIDTH, IMG_HEIGHT, 3], name = 'input')
labels = tf.placeholder(tf.int32, shape = (None), name = 'labels')
global_step = tf.Variable(0, name = 'global_step', trainable = False)
learning_rate_placeholder = tf.placeholder(tf.float32, [], name = 'learning_rate')
```

Figure 4.19 Code Snippet – Placeholder

```
with tf.variable_scope('conv1_1') as scope:
        kernel = tf.Variable(initializer([3, 3, 3, 96]), name = 'weights')
        conv = tf.nn.conv2d(images, kernel, [1, 1, 1, 1], padding = 'SAME')
        out = batch_norm("bn", conv, is_training)
        conv1_1 = tf.nn.relu(out)

pool1 = tf.nn.max_pool(conv1_1,
        ksize = [1, 2, 2, 1], strides = [1, 2, 2, 1], padding='SAME', name = 'pool1')
```

Figure 4.20 Code Snippet – Convolution

- **CNN Architecture**
  - CNN Architecture consist of Convolution, Batch Normalization, Relu and Maxpool layers and ends with fully connected layer based on model design.
  - The second argument of *tf.nn.conv2d* is kernel variable, in convolution kernel variable initializer ([3, 3, 3, 96] in above figure) we can define size of kernel (first two param), input channel to the layer (third param) and number of filters of that layer (forth param).
  - The third argument of *tf.nn.conv2d* is stride, which is currently [1, 1, 1, 1].

```
with tf.variable_scope('fc1') as scope:
        shape = int(np.prod(pool4.get_shape()[1:]))
        pool4_flat = tf.reshape(pool4, [-1, shape])
        features = tf.contrib.layers.fully_connected(pool4_flat, num_outputs = 256, activation_fn = None)

if not is_training:
        return _, _, _, _, _, _, _, _, _

with tf.variable_scope('fc2') as scope:
        fc2 = tf.contrib.layers.fully_connected(features, num_outputs = num_classes, activation_fn = None)
```

Figure 4.21 Code Snippet – FC Layers

  - There are two fully connected layers in this model, out of which the last fully connected layer (fc2) is removed in freezing as it is used for training the model only.
  - The number of outputs of the fully connected layer can be configured by *tf.contrib.layers.fully_connected* API's argument num_outputs. In this model, the first fully connected layer number of outputs are features of face and second fully connected layer number of outputs are number of classes in dataset, which are calculated automatically by code from given dataset directory.

- **Loss Functions**

  The Loss layer consists of two different losses:

  - Softmax Loss – It is mainly classification based.
  - Center Loss – It bounds the features of the same class together.

  These two losses get combined in total loss and is given to optimizer to optimize loss of the whole model. The Center Loss is multiplied by LAMDA and resultant value is used to calculate total loss.

```
with tf.variable_scope('loss'):
        with tf.variable_scope('center_loss'):
                center_loss, centers, centers_batch = get_center_loss(features, labels, num_classes)
        with tf.variable_scope('softmax_loss'):
                softmax_loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(labels = tf.to_float(labels), logits = fc2))
        with tf.variable_scope('total_loss'):
                total_loss = tf.add(softmax_loss, tf.multiply(center_loss,LAMDA))
```

Figure 4.22 Code Snippet – Loss Layers

4.2.2.2. Model Freezing

The Model training graph cannot be used directly for inference, as it contains training specific nodes.

To use it for inference, training specific nodes need to be removed and graph needs to be freezed. Model Freezing code does this operation from the latest checkpoint files and saves the final graph along with weights and biases in form of protobuf (.pb) file in the log directory.

```
if freeze_model:
        file_name = tf.train.latest_checkpoint(log_dir).split(".")[-2].split("/")[-1]
        with tf.Session() as Session:
                tf.train.write_graph(Session.graph_def, log_dir, file_name + '.pbtxt')
        os.chdir(log_dir)
        model_info = [r''+"./"+file_name+'.pbtxt','input',  'fc1/fully_connected/BiasAdd','input',True,[1,90,90,3] ]
        demoCKPT2PB(model_info)
        print('Frozen graph saved at path ' + log_dir)
        #tf.train.write_graph(sess.graph_def, log_dir, MODEL_NAME + '.pb', as_text=False)
        sys.exit()
```

Figure 4.23 Code Snippet – Model Freezing

4.2.2.3. Data Preparation

The *ImageDataGenerator()* API is used to generate the input data tensors from the given directory. The *validation_spli* is the ratio of dividing the dataset in to train set and validation set. *train_it* and *val_it* are iterators which return the next batch.

```
datagen = ImageDataGenerator(validation_split = split_ratio)
train_it = datagen.flow_from_directory(data_dir, target_size = (IMG_WIDTH, IMG_HEIGHT),
        color_mode = 'rgb', class_mode = 'categorical', batch_size = batch_size, shuffle = True, subset = 'training')
val_it = datagen.flow_from_directory(data_dir, target_size = (IMG_WIDTH, IMG_HEIGHT),
        color_mode = 'rgb', class_mode = 'categorical', subset='validation')
```

Figure 4.24 Code Snippet – Dataset Preparing

4.2.2.4. Training

There are threads running in background that continuously fetch and preprocess the data and stores it in a queue in batches. *q.get()* method pops the batch from queue and is fed to the network placeholder using *feed_dict*.

To train the network over a batch, TensorFlow *sess.run* method is used, which returns the computed losses and accuracy on predicted targets by network.

Network is trained in main loop which runs for *'max_steps'*. More configuration of training can be done using arguments while triggering training.

```
iterator_train = q.get()
_,softmax_loss_train,center_loss_train,accuracy_train,summary = sess.run([train, softmax_loss, center_loss,
                                                accuracy, merged_summary_op],
                                                feed_dict = {images:iterator_train[0], labels:iterator_train[1]
                                                ,learning_rate_placeholder:learning_rate})
```

Figure 4.25 Code Snippet – Training

4.2.3. Training from Scratch and/or Transfer Learning

To train the machine:

1. Run the command in the *code root* directory:

```
$ python face-identification.py --data_dir=<path of data directory>
```

```
[earth]$ python3 face-identification.py --data_dir /home/earth/ssd/faceId/dataset/train_aug
Using TensorFlow backend.
Number of clases are  8741
name: GeForce RTX 2080 Ti major: 7 minor: 5 memoryClockRate(GHz): 1.755
pciBusID: 0000:01:00.0
totalMemory: 10.73GiB freeMemory: 4.33GiB
Found 5688962 images belonging to 8741 classes.
Found 628237 images belonging to 8741 classes.
Starting thread 139937718466304
Starting thread 139937710073600
Starting thread 139938246928128
Starting Training...!!!
Starting thread 139938255320832
2019-07-09 11:18:17: Step 0 softmax_loss: 9.1922, Center_loss: 98725.46(* 0.003=296.17638), ACU: 0.0
2019-07-09 11:18:37: Step 100 softmax_loss: 9.0761, Center_loss: 349.0126(* 0.003=1.04704), ACU: 0.0002
2019-07-09 11:18:57: Step 200 softmax_loss: 9.0754, Center_loss: 1.9106(* 0.003=0.00573), ACU: 0.0002
2019-07-09 11:19:19: Step 300 softmax_loss: 9.075, Center_loss: 0.0773(* 0.003=0.00023), ACU: 0.0
2019-07-09 11:19:44: Step 400 softmax_loss: 9.075, Center_loss: 0.026(* 0.003=8e-05), ACU: 0.0
2019-07-09 11:20:09: Step 500 softmax_loss: 9.0746, Center_loss: 0.0148(* 0.003=4e-05), ACU: 0.0002
2019-07-09 11:20:34: Step 600 softmax_loss: 9.0745, Center_loss: 0.0104(* 0.003=3e-05), ACU: 0.0
2019-07-09 11:20:58: Step 700 softmax_loss: 9.0745, Center_loss: 0.0071(* 0.003=2e-05), ACU: 0.0
2019-07-09 11:21:23: Step 800 softmax_loss: 9.0744, Center_loss: 0.0056(* 0.003=2e-05), ACU: 0.0002
2019-07-09 11:21:49: Step 900 softmax_loss: 9.0744, Center_loss: 0.0046(* 0.003=1e-05), ACU: 0.0
2019-07-09 11:22:14: Step 1000 softmax_loss: 9.0743, Center_loss: 0.0039(* 0.003=1e-05), ACU: 0.0
```

Figure 4.26 Training Script Execution for Scratch Training

Notes:

- Training Configurations Parameters:
  - --data_dir – Directory path of dataset
  - --log_dir – Log directory to write summary for TensorBoard and save checkpoint
  - --cpu – Flag to do training on cpu only
  - --batch_size – Batch size
  - --summary_step – Steps to print summary
  - --checkpoint_step – Steps to save checkpoint
  - --val_interval – Training steps after which validation occurs
  - --val_step_size – Number of validation steps to perform
  - --max_steps – Maximum steps to train model
  - --restore_ckpt – Restore Latest checkpoints if flag is present
  - --learning_rate – Initial learning rate of model
  - --data_split_ratio – Fraction to split data into validation set
  - --lr_steps – List of steps where to decay the learning rate
  - --lr_decay – Learning Rate Decay
  - --threads – Number of helper threads to fetch the data
  - --freeze_model – Flag to generate pb file from the latest checkpoint

- Below are the default values of the parameters when training is triggered with only *data_dir*.
    - --log_dir: "./logs"
    - --batch_size: 64
    - --summary_step: 100
    - --checkpoint_step: 10000
    - --val_interval: 10000
    - --val_step_size: 100
    - --max_steps: 1000000
    - --learning_rate: 0.0001
    - --data_split_ratio: 0.1
    - --threads: 4

2. Run the command below to trigger training with *--restore_ckpt* flag and log directory in which checkpoint is restored.

```
$ python face-identification.py --data_dir=<path of data directory> --log_dir=<path to log
directory> --restore_ckpt
```

```
[earth]$ python3 face-identification.py --data_dir /home/earth/ssd/faceId/dataset/train_aug --log_dir ./logs/ --
restore_ckpt
Using TensorFlow backend.
Number of clases are  8741
name: GeForce RTX 2080 Ti major: 7 minor: 5 memoryClockRate(GHz): 1.755
pciBusID: 0000:01:00.0
totalMemory: 10.73GiB freeMemory: 4.48GiB
/home/logs/Face_Identification10000.ckpt-10001
Instructions for updating:
Use standard file APIs to check for files with this prefix.
Found 5688962 images belonging to 8741 classes.
Found 628237 images belonging to 8741 classes.
Starting Training...!!!
2019-07-09 13:00:34: Step 10100 softmax_loss: 9.0507, Center_loss: 2.1549(* 0.003=0.00646), ACU: 0.0002
2019-07-09 13:00:52: Step 10200 softmax_loss: 9.0507, Center_loss: 2.1335(* 0.003=0.0064), ACU: 0.0003
2019-07-09 13:01:10: Step 10300 softmax_loss: 9.0509, Center_loss: 2.1585(* 0.003=0.00648), ACU: 0.0002
2019-07-09 13:01:28: Step 10400 softmax_loss: 9.0502, Center_loss: 2.2132(* 0.003=0.00664), ACU: 0.0
2019-07-09 13:01:47: Step 10500 softmax_loss: 9.0513, Center_loss: 2.1544(* 0.003=0.00646), ACU: 0.0003
```

Figure 4.27 Training Script Execution for Transfer Learning

- Training status can be checked in logs by observing different losses like softmax loss, center loss, and accuracy.

```
2019-07-02 15:29:44: Step 612600 softmax_loss: 1.1536, Center_loss: 313.5066(* 0.003=0.94052), ACU: 0.8445
2019-07-02 15:30:03: Step 612700 softmax_loss: 1.1357, Center_loss: 313.7823(* 0.003=0.94135), ACU: 0.8514
2019-07-02 15:30:21: Step 612800 softmax_loss: 1.1132, Center_loss: 311.508(* 0.003=0.93452), ACU: 0.8541
2019-07-02 15:30:50: Step 612900 softmax_loss: 1.1265, Center_loss: 313.0352(* 0.003=0.93911), ACU: 0.8528
2019-07-02 15:31:14: Step 613000 softmax_loss: 1.1314, Center_loss: 312.85(* 0.003=0.93855), ACU: 0.8504
2019-07-02 15:31:52: Step 613100 softmax_loss: 1.1408, Center_loss: 312.2625(* 0.003=0.93679), ACU: 0.8482
2019-07-02 15:32:29: Step 613200 softmax_loss: 1.1364, Center_loss: 313.0158(* 0.003=0.93905), ACU: 0.8487
2019-07-02 15:33:07: Step 613300 softmax_loss: 1.1458, Center_loss: 313.3766(* 0.003=0.94013), ACU: 0.8445
2019-07-02 15:33:45: Step 613400 softmax_loss: 1.1339, Center_loss: 311.9562(* 0.003=0.93587), ACU: 0.8487
2019-07-02 15:34:23: Step 613500 softmax_loss: 1.1357, Center_loss: 312.1662(* 0.003=0.9365), ACU: 0.8505
```

Figure 4.28 Training Logs

3. Check the training status on TensorBoard by running the command below.

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

```
[earth@bangalore2 Final_code]$ tensorboard --logdir=logs
TensorBoard 1.14.0 at http://bangalore2.sibshivalik.com:6006/ (Press CTRL+C to quit)
I0702 11:07:43.642186 140085573707520 _internal.py:122] ::ffff:192.168.1.8 - - [02/Jul/2019 11:07:43] "GET / HTT
P/1.1" 200 -
I0702 11:07:45.654393 140085573707520 _internal.py:122] ::ffff:192.168.1.8 - - [02/Jul/2019 11:07:45] "GET /font
-roboto/oMMgfZMQthOryQo9n22dcuvvDin1pK8aKteLpeZ5c0A.woff2 HTTP/1.1" 200 -
```

Figure 4.29 TensorBoard – Launch

This command provides the link which needs to be copied and open in any browser like Chrome, Firefox, and others. You can also right-click on the link and click **Open Link**.

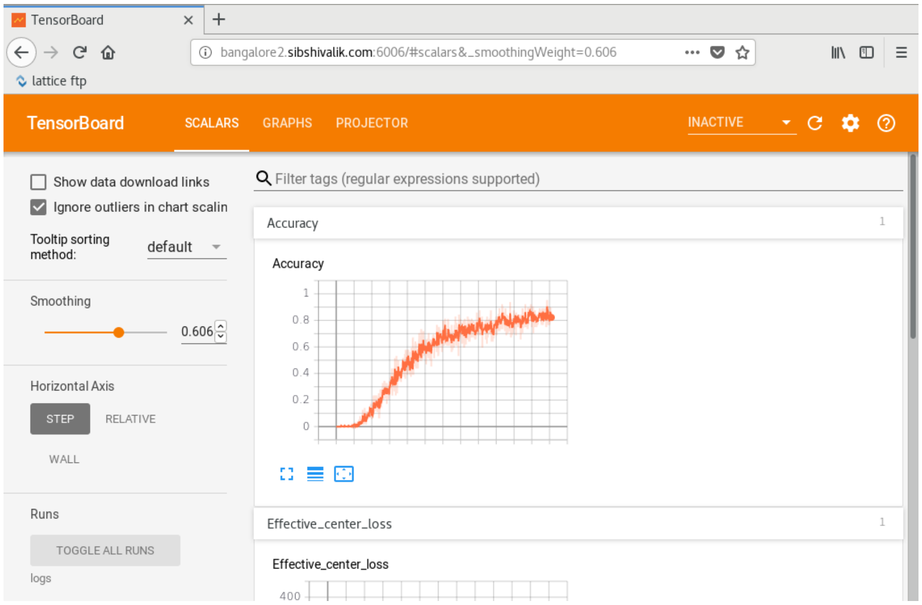


Figure 4.30 TensorBoard – Link Default Output in Browser

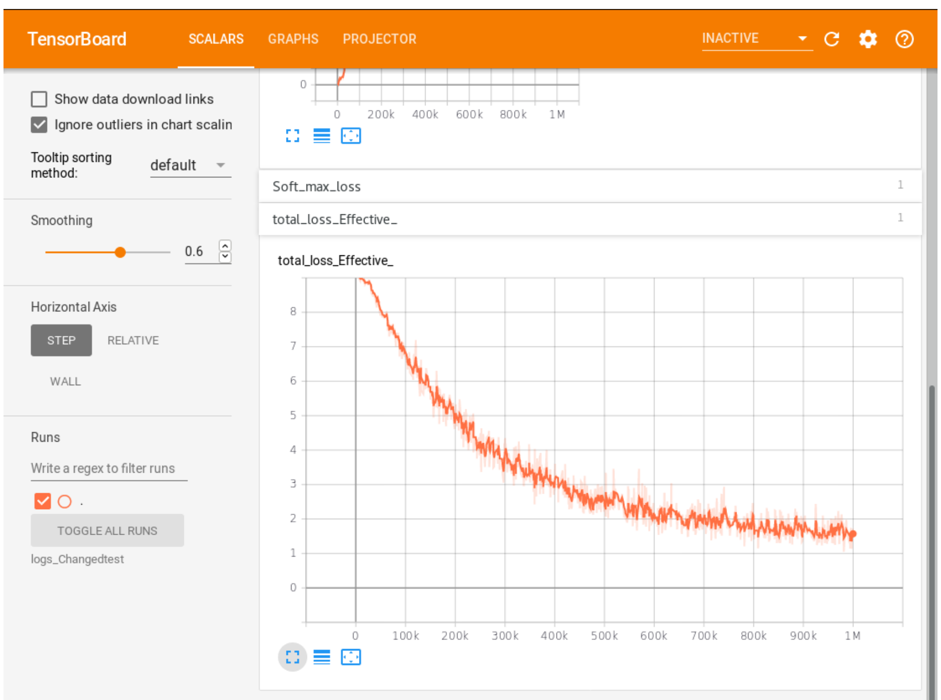


Figure 4.311 TensorBoard – Effective Total Loss Graph

4.2.4. Checkpoint

There are five different types of file as training output:

- A *checkpoint* file
- A *data* file
- A *meta* file
- An *index* file
- If you use TensorBoard, an *events* file

The three checkpoint file types are here to store the compressed data about your models and its weights.

- The **checkpoint** file is just a bookkeeping file that you can use in combination of high-level helper for loading different time saved ckpt files.
- The **.meta** file holds the compressed Protobufs graph of your model and all the metadata associated such as collections, learning rate, and operations.
- The **.index** file holds an immutable key-value table linking a serialised tensor name and where to find its data in the **ckpt.data** files**.**
- The **.data** files hold the data (weights) itself which is usually quite big in size. There can be many data files because they can be shared and/or created on multiple timestamps while training.
- Finally, the **events** file store everything you need to visualise your model and all the data measured while you were training using summaries. This has nothing to do with saving/restoring your models itself.

4.2.5. Freezing the Model

This section describes the how to freeze the model which is aligned with Lattice SensAI tool.

Use the command below to generate the frozen protobuf(.pb) file:

```
$ python face-identification.py --log_dir=<path of log(checkpoint) directory> --freeze_model
```

```
[earth]$ python3 face-identification.py --log_dir /home/earth/ssd/faceId/tensorflow/Final_code/logs --freeze_mod
el
Face_Identification1000000.pbtxt saved at path /home/earth/ssd/faceId/tensorflow/Final_code/logs
/home/earth/ssd/faceId/tensorflow/Final_code/logs
inputShape shape [1, 90, 90, 3]
op: "Placeholder"
```

Figure 4.32 PB File Generation from Checkpoint

The command above creates the frozen graph (.pb) in the log directory which can be used in SensAI tool for firmware generation.

5. Creating Binary File

This section describes the basic Lattice sensAI tool flow starting from the project creation, different projects and/or YML settings, model analysis, and compilation. It also includes optional sensAI simulation steps. The following steps should be followed for sensAI firmware bin file generation.

5.1. Creating Binary File with Caffe

To create a project in sensAI tool with Caffe framework:

Figure 5.1 SensAI Home Screen

1. Click **File > New**.
2. Enter the following settings:
    - Project name
    - Framework – Caffe
    - Class – CNN
    - Device – ECP5

    You can also choose the directory where you want to save the project files.

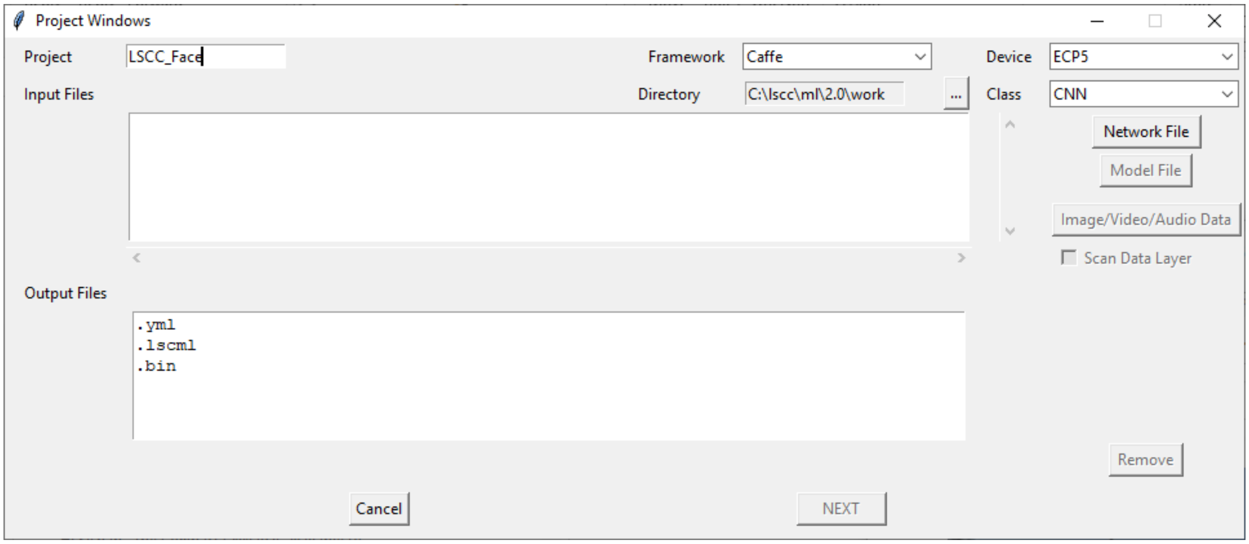


Figure 5.2 SensAI –Project Creation, Framework, and Device Selection

3. Click **Network File** and select the network file.

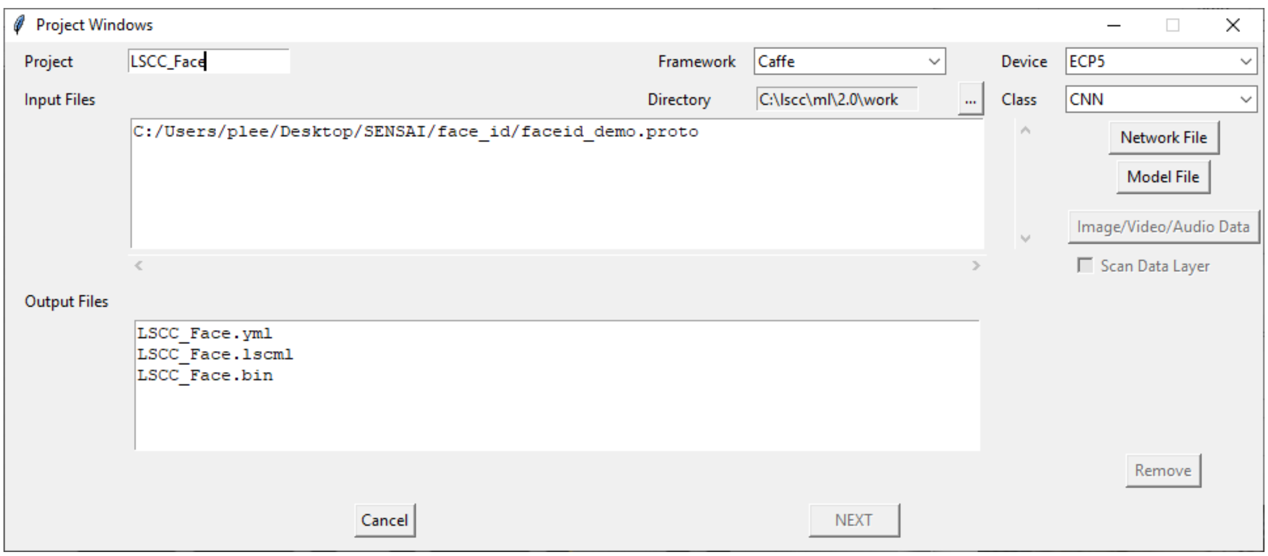


Figure 5.3 SensAI – Network File Selection

4. Click **Model File** and select the model file.

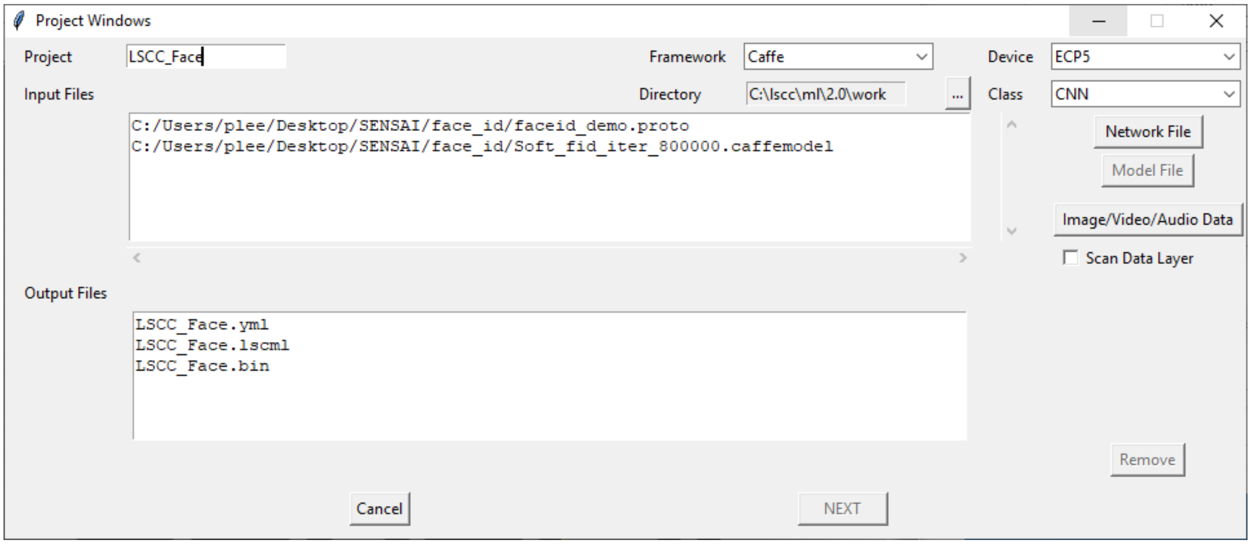


Figure 5.4 SensAI – Model File Selection

5. Click **Image/Video/Audio Data** and select the image input file.

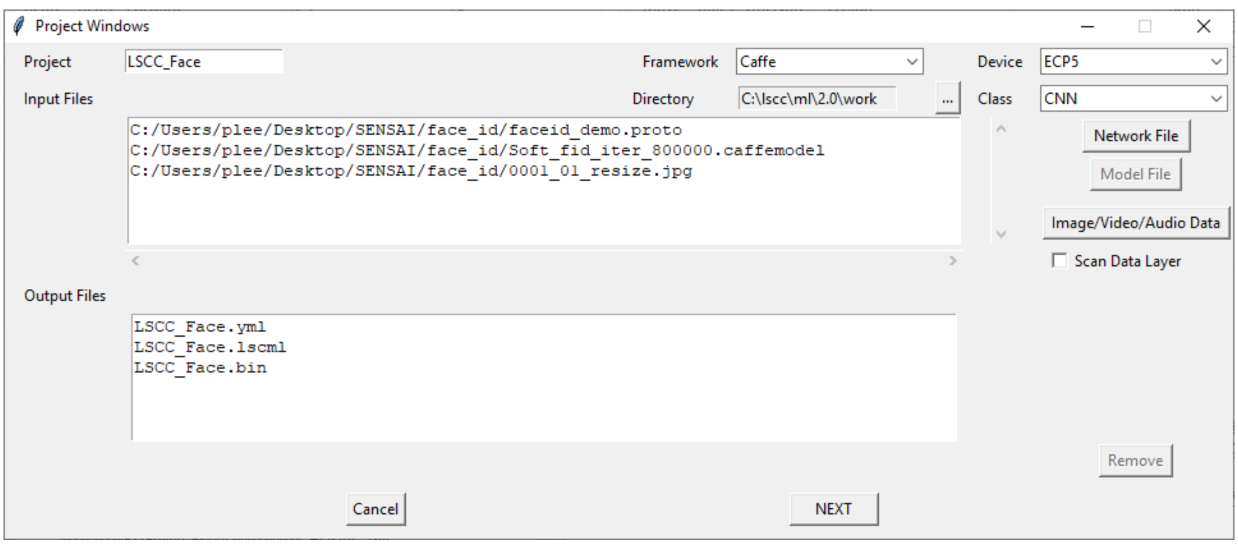


Figure 5.5 SensAI – Image Data File Selection

6. Click **Next**.
7. Configure your project settings.

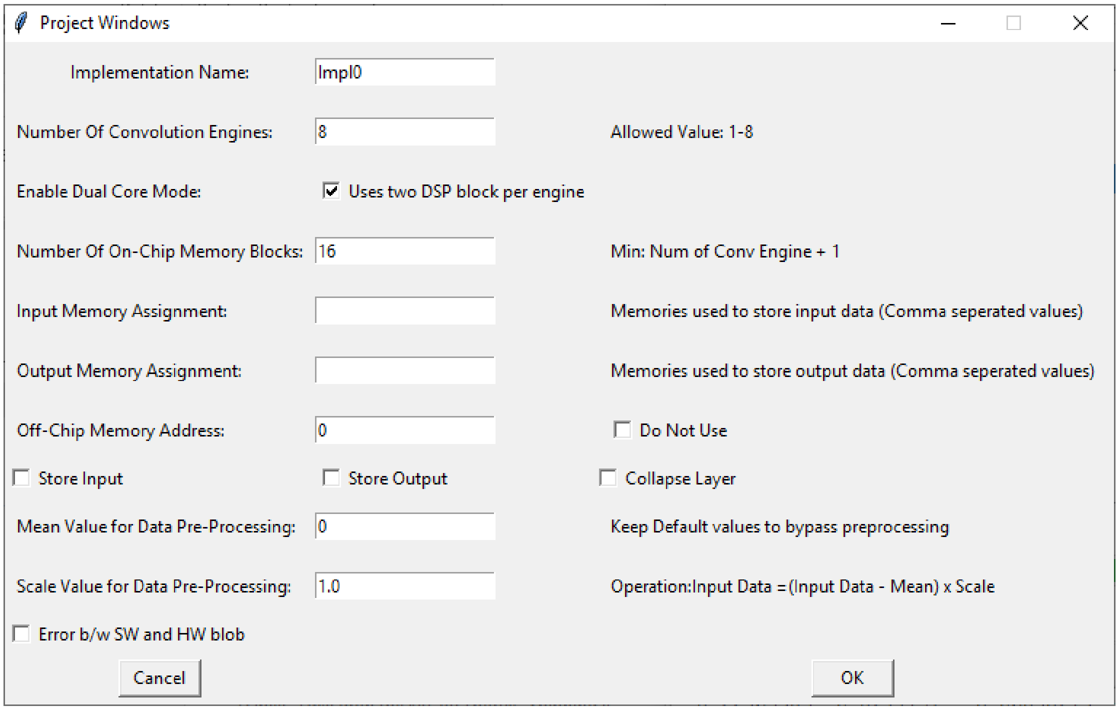


Figure 5.6 SensAI – Project Settings

8. Click **OK** to create project.
9. Double-click **Analyze**.

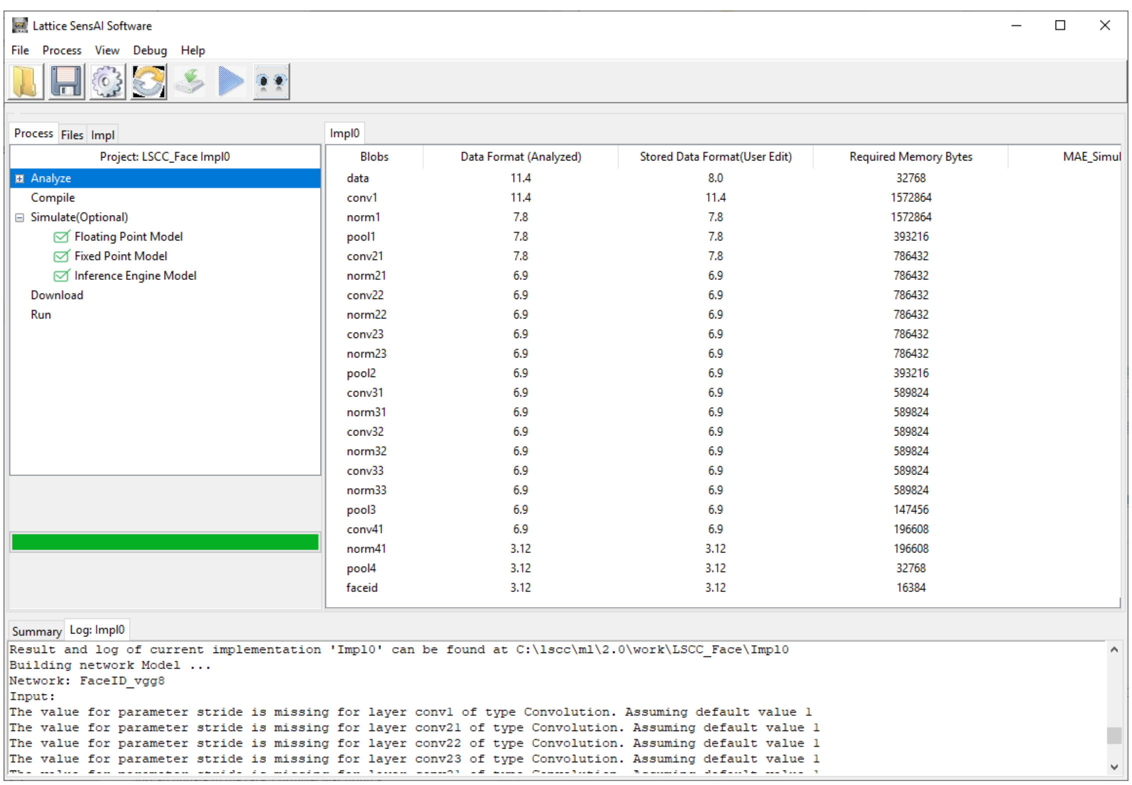


Figure 5.7 SensAI – Analyze Project

The SensAI tool generates the Q format for each layer based on range analysis, input image, and other parameters.

10. Double-click **Compile**.

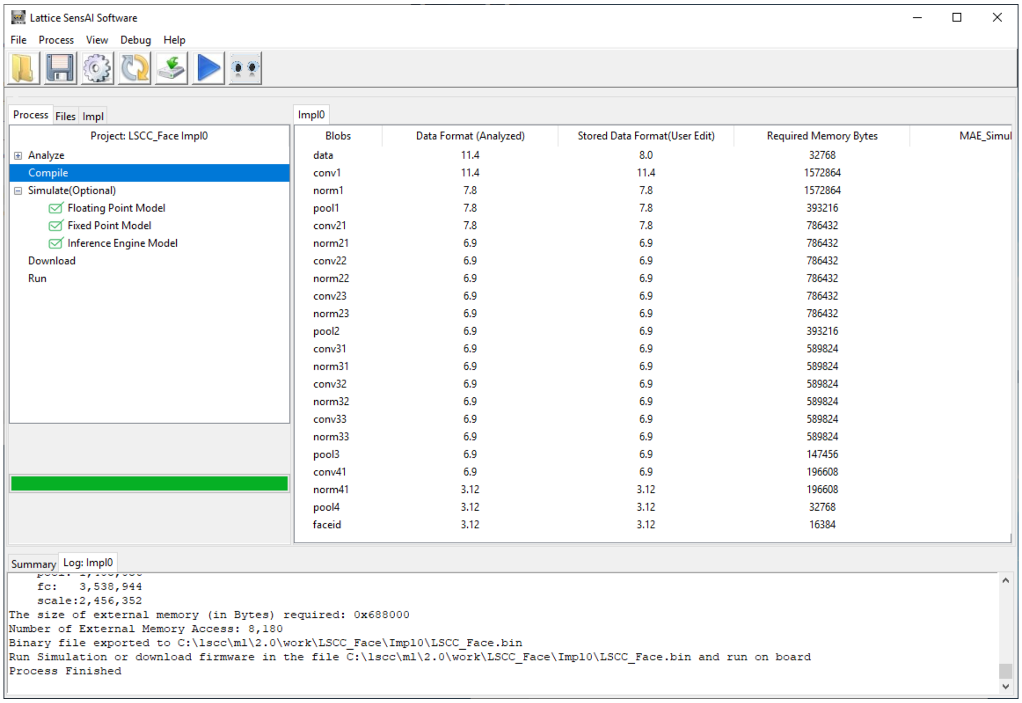


Figure 5.8 SensAI – Compile Project

The Firmware Bin file location is displayed in the compilation log. You can use the generated firmware bin on the hardware for testing. You can also perform simulation on sensAI by double clicking on **Simulation**.

After successful simulation, SensAI dumps the last layer output with floating point Caffe simulation values and actual h/w simulation output values in the output window.

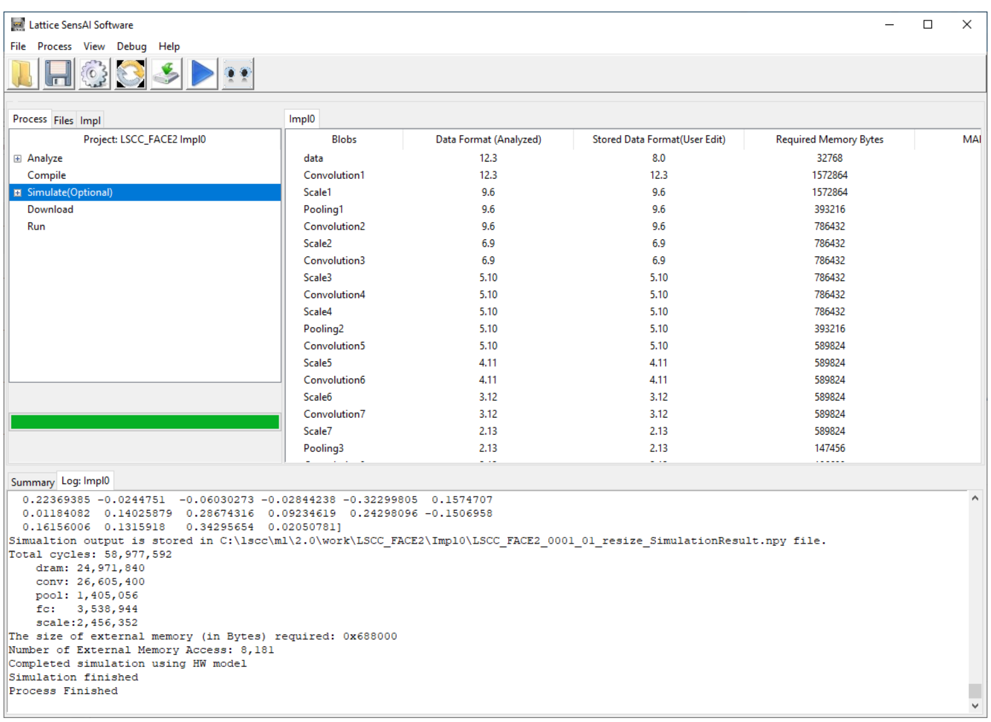


Figure 5.9 SensAI – Simulation and Output

5.2. Creating Binary File with TensorFlow

To create a project in SensAI tool with TensorFlow framework:

Figure 5.10 SensAI – Home Screen

1. Click File > New.
2. Enter the following settings:
    - Project name
    - Framework – TensorFlow
    - Class – CNN
    - Device – ECP5

    You can also choose the directory where you want to save the project files.

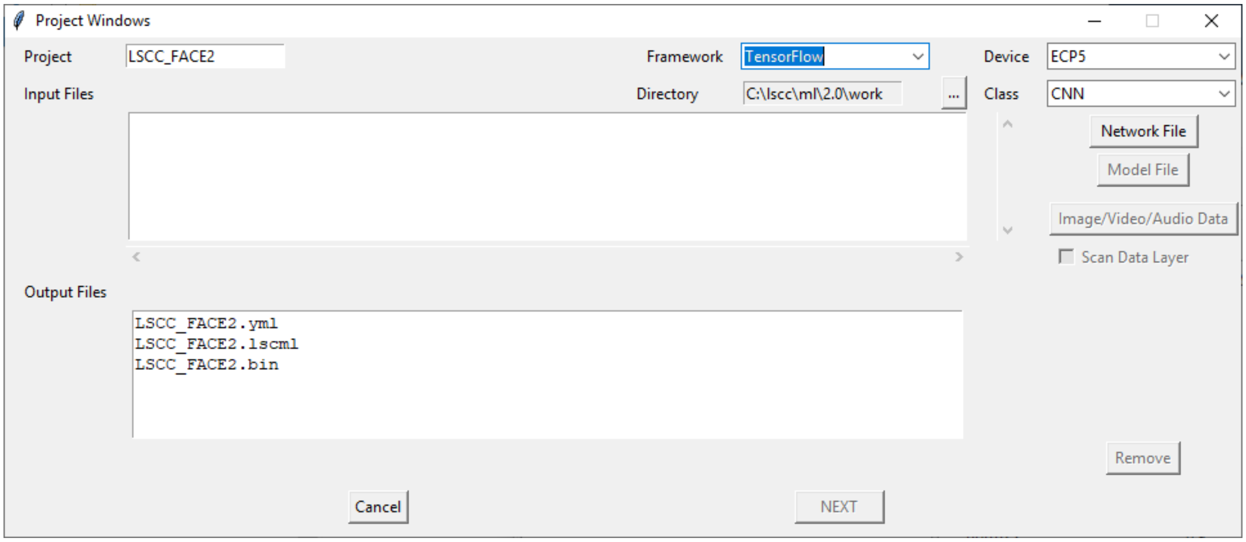


Figure 5.11 sensAI –Project Creation, Framework, and Device Selection

3. Click **Network File** and select the network (PB) file.

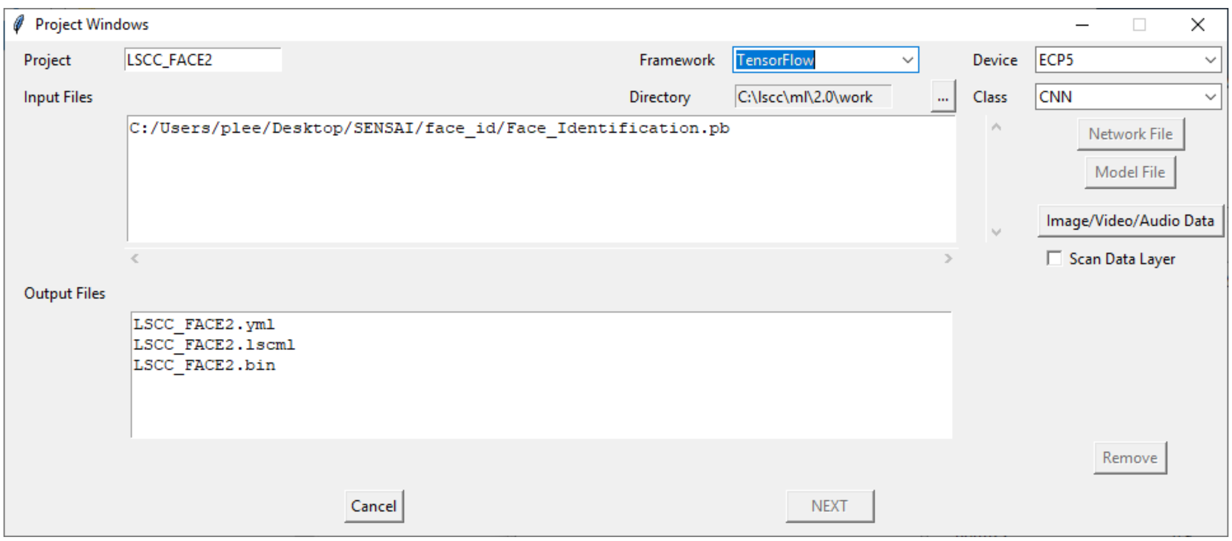


Figure 5.12 SensAI – Network File Selection

4. Click **Image/Video/Audio Data** and select the image input file.

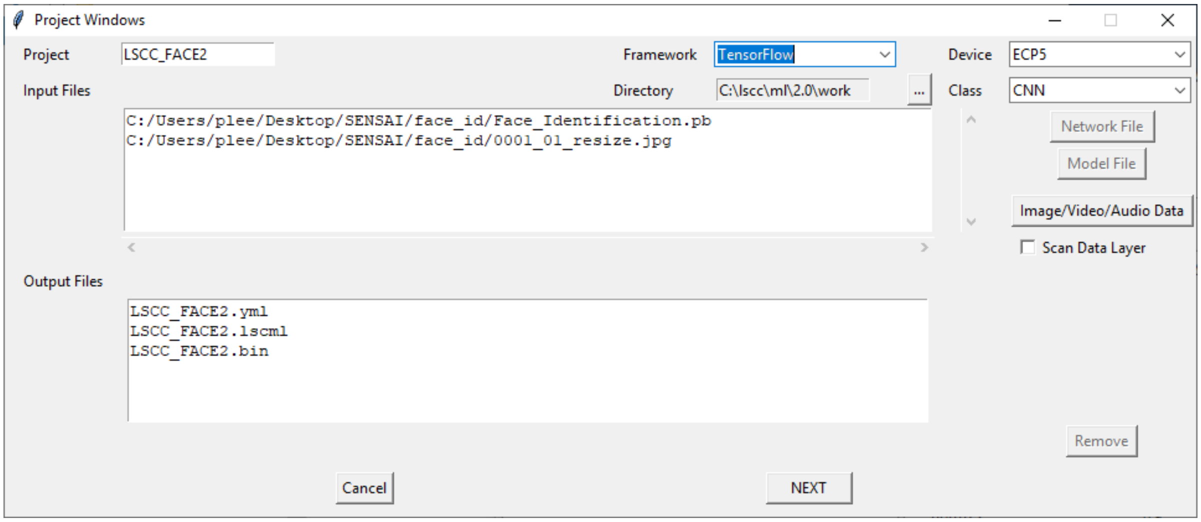


Figure 5.13 SensAI – Image Data File Selection

5. Click **Next**.
6. Configure your project settings.

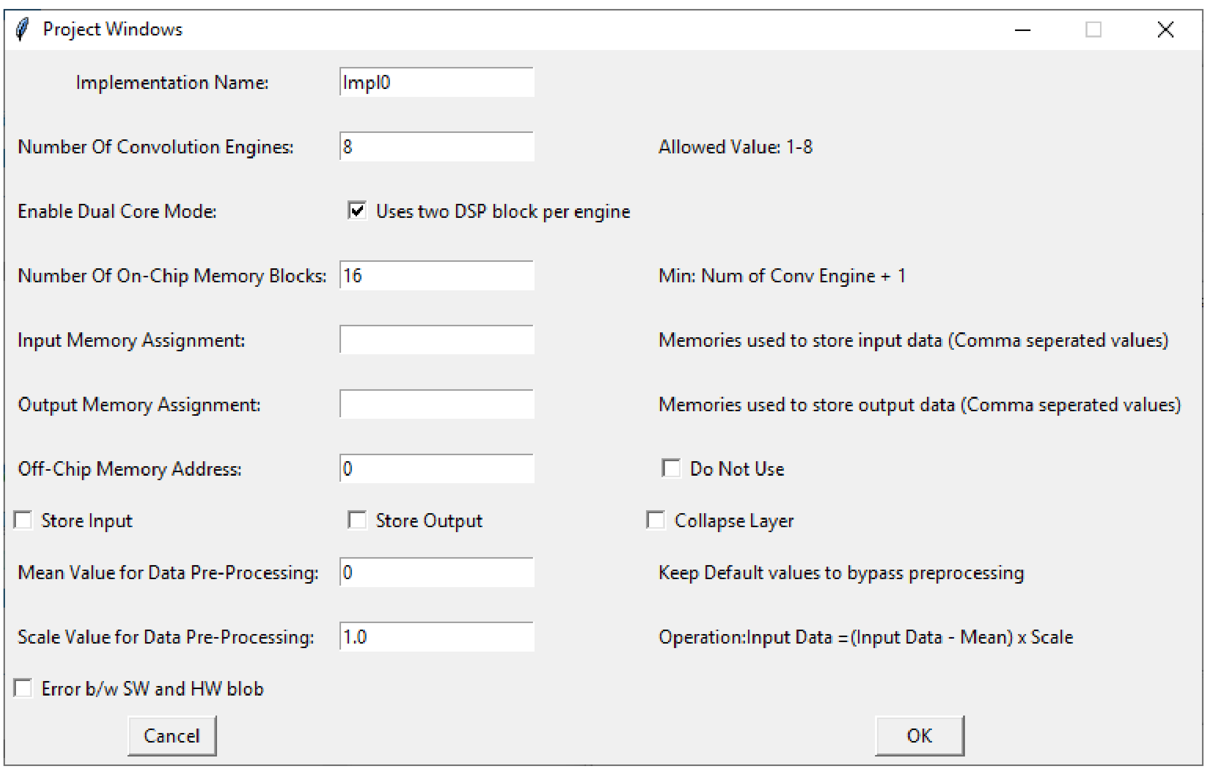


Figure 5.14 SensAI – Project Settings

7. Click **OK** to create the project.

8. Double click **Analyze**.

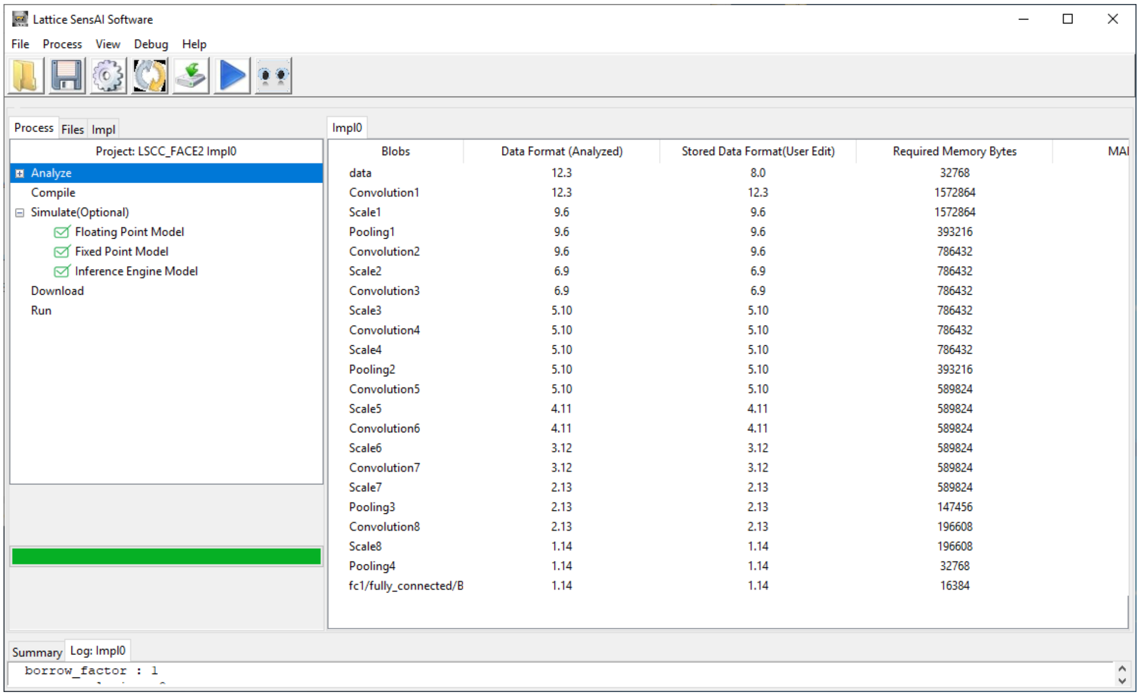


Figure 5.15 SensAI – Analyze Project

The sensAI tool generates the Q format for each layer based on range analysis, input image, and other parameters.

9. Double-click **Compile**.

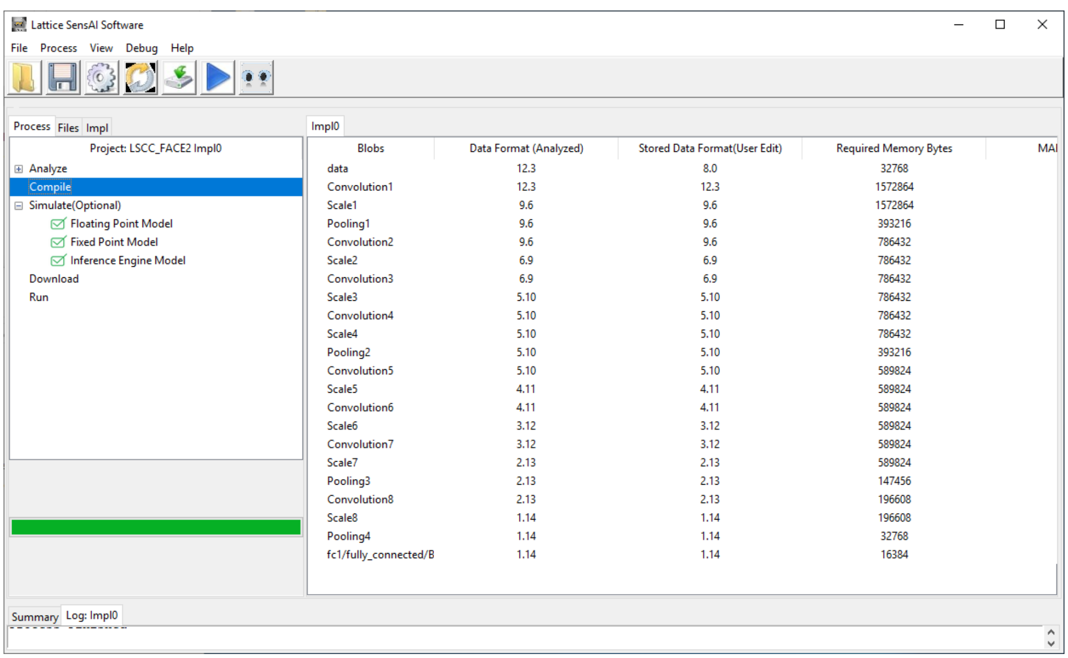


Figure 5.16 SensAI – Compile Project

The Firmware Bin file location is displayed in the compilation log. You can use the generated firmware bin on the hardware for testing. You can also perform simulation on SensAI by double clicking on **Simulation**.

After successful simulation, SensAI dumps the last layer output with floating point TensorFlow simulation values and actual h/w simulation output values in the output window.

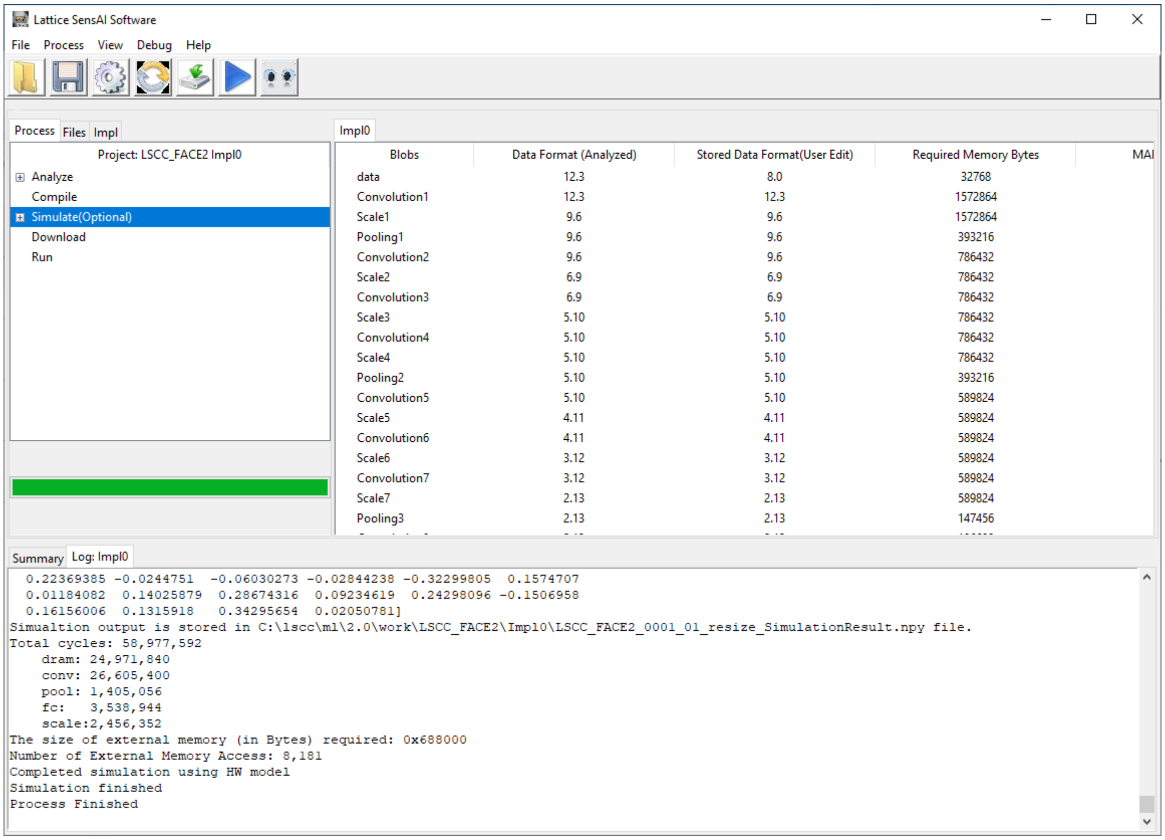


Figure 5.17 SensAI – Simulation and Output

6. RTL Design Overview

6.1. Functional Block Diagram

Figure 6.1 shows the block diagram of the Face Identification reference design.

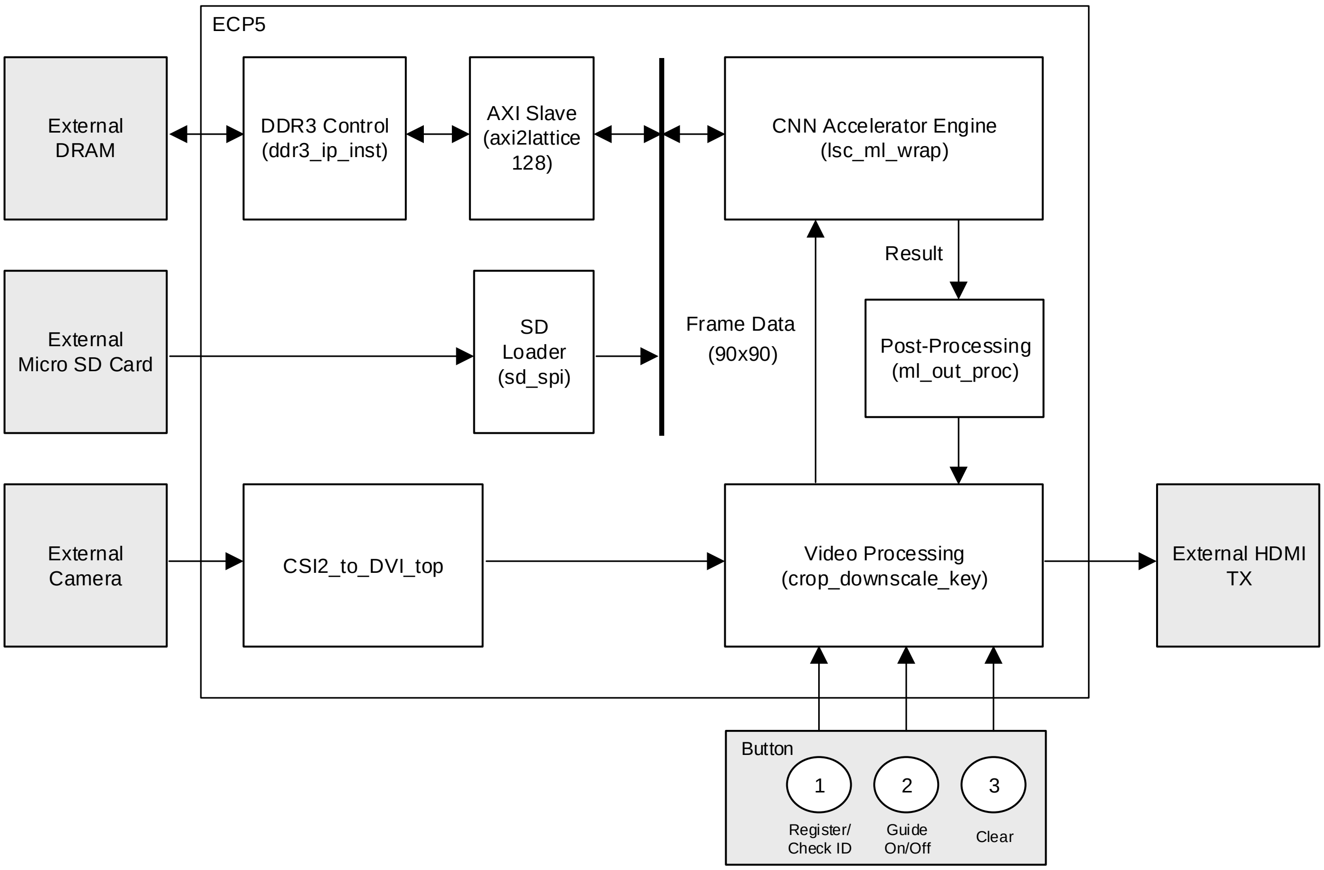


Figure 6.1 Face Identification Reference Design Block Diagram

6.2. Top Level Blocks

The reference design uses ECP5-85 FPGA containing the following major blocks:

- CNN accelerator engine
- SD card to SPI interface
- AXI Slave interface
- DDR3 memory interface
- CSI2 to DVI interface
- Video processing module
- Post processing module

6.3. Functional Blocks Overview

6.3.1. CNN Accelerator Engine

The Lattice Semiconductor CNN Accelerator IP Core is a calculation engine for Deep Neural Network with fixed point weight or binary weight. It calculates full layers of Neural Network including convolution layer, pooling layer, batch normalization layer, and full connect layer by executing sequence code with weight value which is generated by the Lattice Neural Network Compiler tool. The engine is optimized for convolutional neural network, hence it can be used for vision-based applications such as classification or object detection and tracking. The IP Core does not require an extra processor. The IP Core can perform all required calculations by itself.

6.3.2. SD Loader

SD card interface in this design is used to get the instruction data into the DRAM for execution by the CNN accelerator IP.

6.3.3. AXI Slave and DDR3 Memory Interface

AXI interface allows instruction code to be written in DRAM before execution of CNN Accelerator IP Core. Input data may also be written in DRAM. CNN Accelerator IP Core reads instruction code from DRAM, and performs computing using internal sub execution engines. Intermediate data may also be transferred from/to DRAM per instruction code.

6.3.4. CSI2 to DVI Interface

This module implements a bridge function that converts the camera input MIPI CSI data to DVI output using CrossLink pASSP and SiI1136 HDMI transmitter.

6.3.5. Video Processing Module

The video processing is handled in 'crop_downscale_key' module which provides all the necessary functions needed to manage: processing of input data, receiving output data, and generating a composite image for output to the HDMI interface.

Key Logic includes:

- Operational Mode Management
- Downscaling
- OSD Text Display onto HDMI

6.3.5.1. Operational Mode Management

The information of the push button connected to ECP5 is passed into this module. This module defines the following operational modes:

- REG MODE (when button 1 is pressed)
- CHECK MODE (no button is pressed, which is the default status)
- GUIDED MODE (when button 2 is pressed)
- CLEAR MODE (when button 3 is pressed)

Information of the current active mode is passed to Post-Processing module which processes the CNN output data accordingly. REG Mode, CHECK Mode, and CLEAR Mode are used in Post Processing module 'ml_out_proc'.

GUIDED Mode is managed in *crop_downscale_key* module as follows:

- This mode gives you guidance on where to place your face for proper detection. If you are too far when registered, the output cannot differentiate you from the background.
- This mode is set to ON when button 2 is pressed. This sets a signal when the guidance marks information needs to be multiplexed into the HDMI display output.
- Fixed guidance marks are set for top, bottom, left, and right by constant pixel and line count values.
- This mode is set to OFF when button 2 is pressed again. Guidance marks are removed from the HDMI display.

6.3.5.2. Downscaling

Output from 'CSI2_to_DVI_top' module is a stream of R, G, B data that reflects the camera image. This input image is then downscaled to 90 × 90 pixels, stored in a frame buffer and passed to output. Image data is written from the frame buffer into the CNN acceleration engine prior to the start of the processing. The data values are considered to be valid only when horizontal and vertical masks are inactive. Mask parameters are set such that it masks out boundary area of full HD resolution (1920 × 1080) to give output resolution of 1080 × 1080.

Downscaling generates a single accumulated pixel value for each 12 × 12 grid of pixels which leads to generate 90 × 90 values (1080/12 × 1080/12) from 1080 × 1080 values. Each accumulated value is written into frame buffer.

Read data from buffer is formatted for the compatibility with the trained network, according to the CNN input data layer configuration.

6.3.5.3. OSD Text Display onto HDMI (lsc_osd_text)

This module takes detected Face features, number of registered Face entries, and calculated distances from registered Face features as input. It calculates ASCII character for each input value and provides bitmap to be displayed on screen.

For invalid Face identification, it provides ASCII character of *?*. For non-registered Face Identification Distance, it provides ASCII character of _. It sets a signal for the HDMI output when text needs to be displayed.

HDMI Output interface contains R, G, B data values multiplexed as follows:

- If Signal Text is On, pass all R, G, B value as 12'hFFF for White color display.
- If Signal Green On, pass only Green pixel value (GUIDED Mode ON).
- If Signal Mask is On, pass darker pixel values.
- Else, pass input R, G, B pixel data value as is.

6.3.6. Post Processing Module

The primary functionality of this block ('ml_out_proc module') is to capture the CNN valid output, Detect or Register Face Identification and pass valid Face Index and Distance to the 'crop_downscale_key' module.

Table 6.1. Post-Processing Core Parameters

Constant	Default (Decimal)	Description
NUM_FEAT	256	Number of Features (values) provided by CNN for each processed frame.
FRAC_POS	13	Fraction Part Width as per Q-Format representation of last output layer.
DIST_THRESH	768	Upper Threshold for Mean Squared Difference (distance) value calculated for Face Features. Value is assigned based on trained model threshold and interpreted based on Q-Format of last CNN layer. Default value is based on Q-format 2.13. In general, DIST_THRESH = 384*(2^n) where n = (14 – FRAC_POS). For example, if you want to change FRAC_POS = 12, DIST_THRESH should be kept to (384*(2^2)) = 1536, where value 384 should be adjusted based on your training and actual environment.

CNN provides 256 values (16-bit each) representing Face Features for each frame processed. These values are processed according to the mode selected (REG/CHECK/CLEAR).

6.3.6.1. REG MODE

- This mode is used to register/store the values of face measurements provided by CNN into memory.
- The stored feature values are used as a reference while CHECK Mode is on.
- This module can store up to 8 Face Identification measurements. Each Face Identification measurement uses separate 8K RAM.

6.3.6.2. CHECK MODE

- This mode is used to detect the valid Face Identification matching with current Face from stored Face Identifications in REG MODE.
- It compares stored Face Identification (0-7) feature values (0-NUM_FEAT) with current CNN output feature values and performs mean square difference to estimate the average distance from actual values for each feature.
- Final accumulated means square difference value for each Face Identification is compared with each other and the index with least difference value is selected.
- If the difference value is less than DIST_THRESH, the selected face index is valid and has minimum distance from the current face.
- The matched Face Index and all Distance values are passed to *crop_downscale_key* module for display. If no Face Identification matches (all calculated distance values > DIST_THRESH OR no Face Identification is registered by REG MODE), then the output face index is passed with invalid value (0xF).

6.3.6.3. CLEAR MODE

- When this mode is activated, all registered Face Index and Distance information are cleared. You can now register new faces using the REG mode.

7. Generating the Bitstream File

This section describes the steps to compile RTL Bitstream using the Lattice Diamond tool.

To generate the RTL Bitstream file:

1. Open the Lattice Diamond Software.

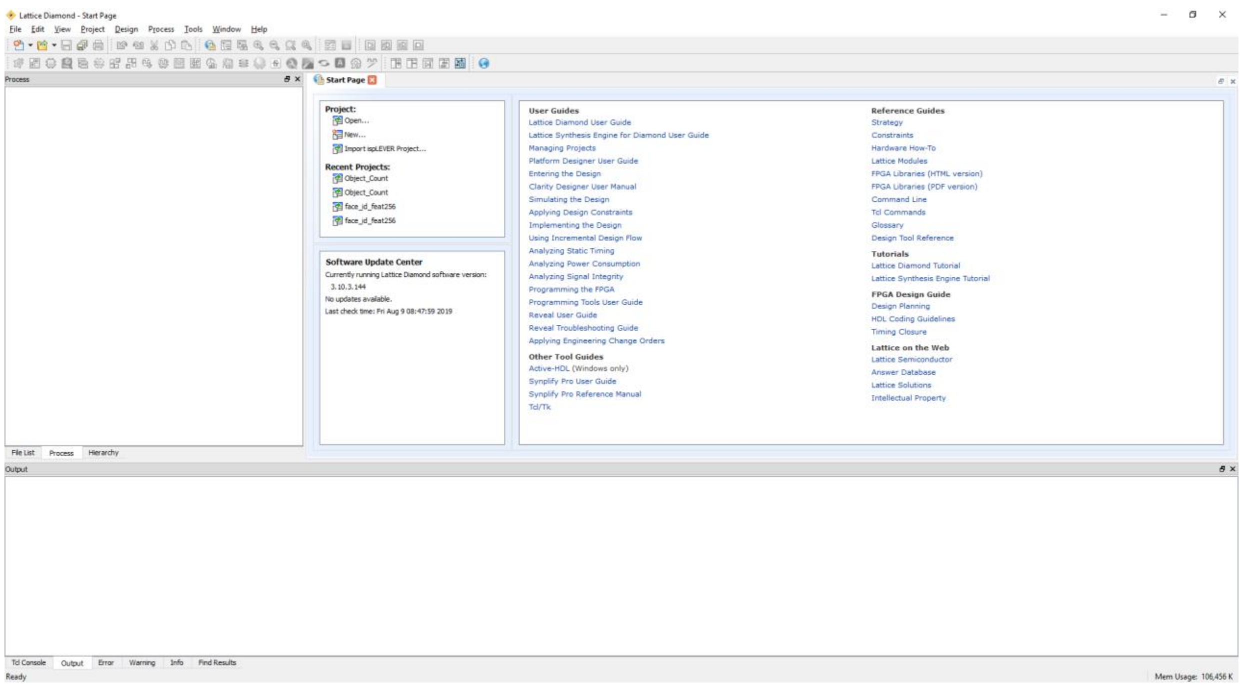


Figure 7.1 Diamond – Default Screen

2. Click **File > Open > Project**. Select the Diamond project file for ECP5 Face Identification Demo RTL.

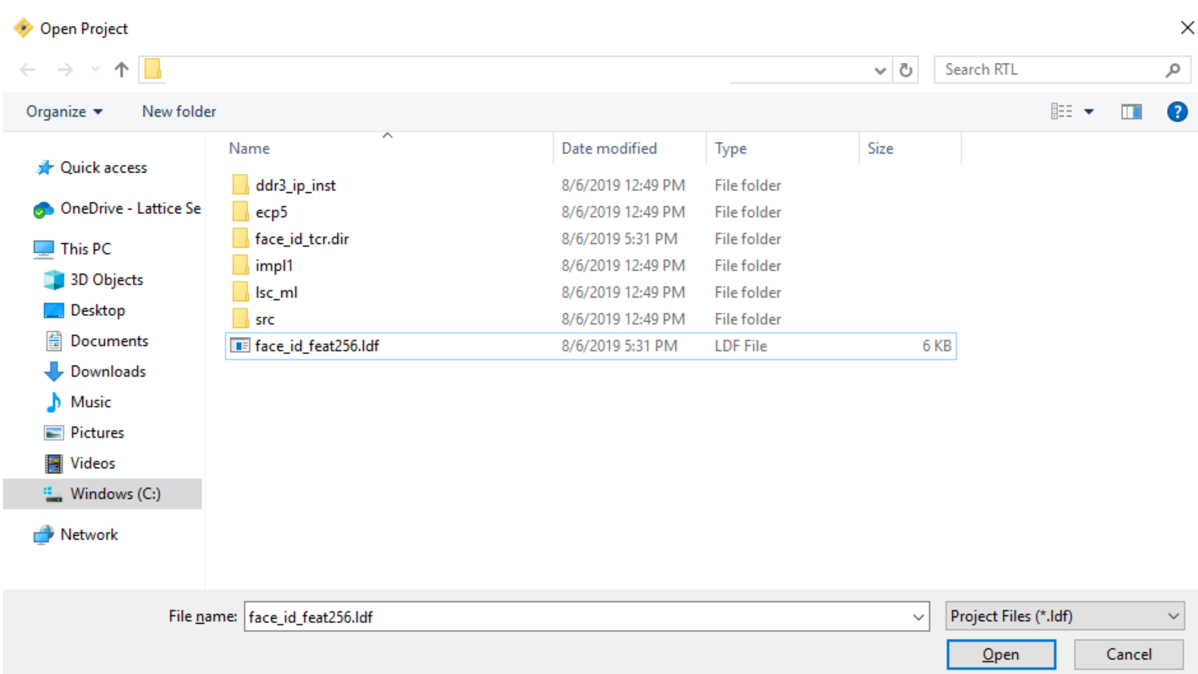


Figure 7.2 Diamond – ECP5 Face Identification Diamond Project File Selection

3. Double-click **Bitstream File** in the left pane to trigger the Bitstream generation.

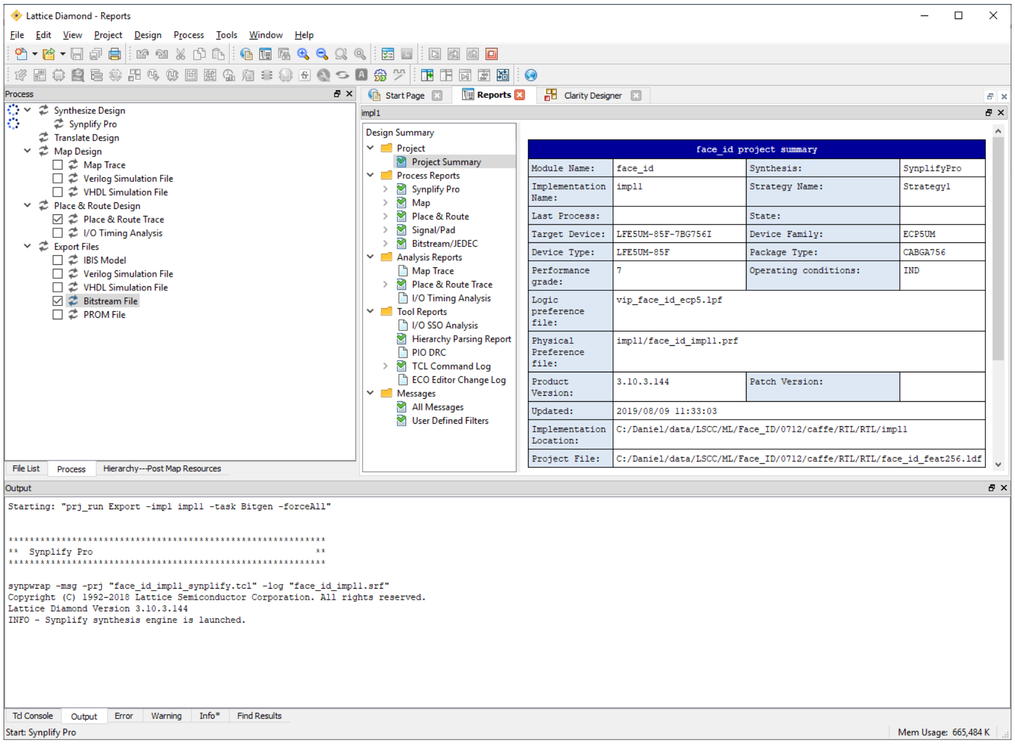


Figure 7.3 Diamond – Trigger Bitstream Generation

4. After successful Bitstream generation, the Diamond tool displays the *Saving bit stream in …* message in the Reports window. Bitstream is generated at *Implementation Location* as shown in Figure 7.4.

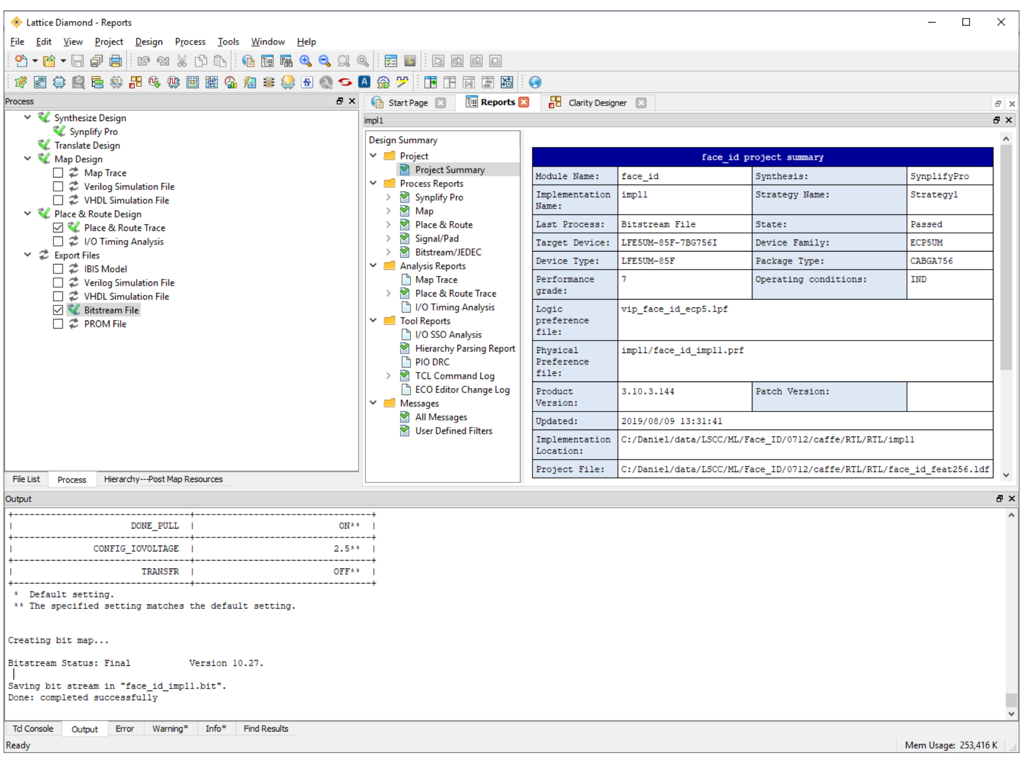


Figure 7.4 Diamond – Bit File Generation Report Window

8. Programming the Binary and Bitstream Files

Both the CrossLink™ VIP Input Bridge Board and the ECP5 VIP Processor Board must be configured and programmed. The demo design firmware must also be programmed onto the MicroSD card, which is plugged into the MicroSD Card Adaptor Board.

8.1. Programming the CrossLink SPI Flash

8.1.1. Erasing the CrossLink SRAM Prior to Reprogramming

If the CrossLink is already programmed, either directly or loaded from SPI Flash, erase the CrossLink SRAM before reprogramming the CrossLink SPI Flash. Keep the board powered on to prevent reloading on reboot.

To erase CrossLink:

1. Launch Diamond Programmer with **Create a new blank project**.
2. Select **LIFMD** for **Device Family** and **LIF-MD6000** for **Device**.

Figure 8.1. Device Selection

3. Right-click and select **Device Properties.**
4. Select **SSPI SRAM Programming** for Access mode and **Erase Only** for Operation.

Figure 8.2. Device Operation

5. Click **OK** to close the Device Properties window.
6. Click the **Program** button  in Diamond Programmer to start the Erase sequence.

8.1.2. Programming the SPI on the CrossLink VIP Input Bridge Board

To program the SPI on the CrossLink VIP Inout Bridge Board:

1. Ensure the CrossLink device is erased by performing Steps 1-6.
2. Right-click and select **Device Properties**.
3. Select **SPI Flash Programming** for **Access mode** and make the following selections:
    a. For **Programming file**, browse and select the **CrossLink bitfile** (*.bit).
    b. For **SPI Flash Options**, refer to Table 8.1.

Table 8.1. SPI Flash Options Selection Guide

Item	Rev B	Rev C
Family	SPI Serial Flash	SPI Serial Flash
Vendor	Micron	Macronix
Device	SPI-N25Q128A	MX25L12835F
Package	8-pin SO8	8-Land WSON

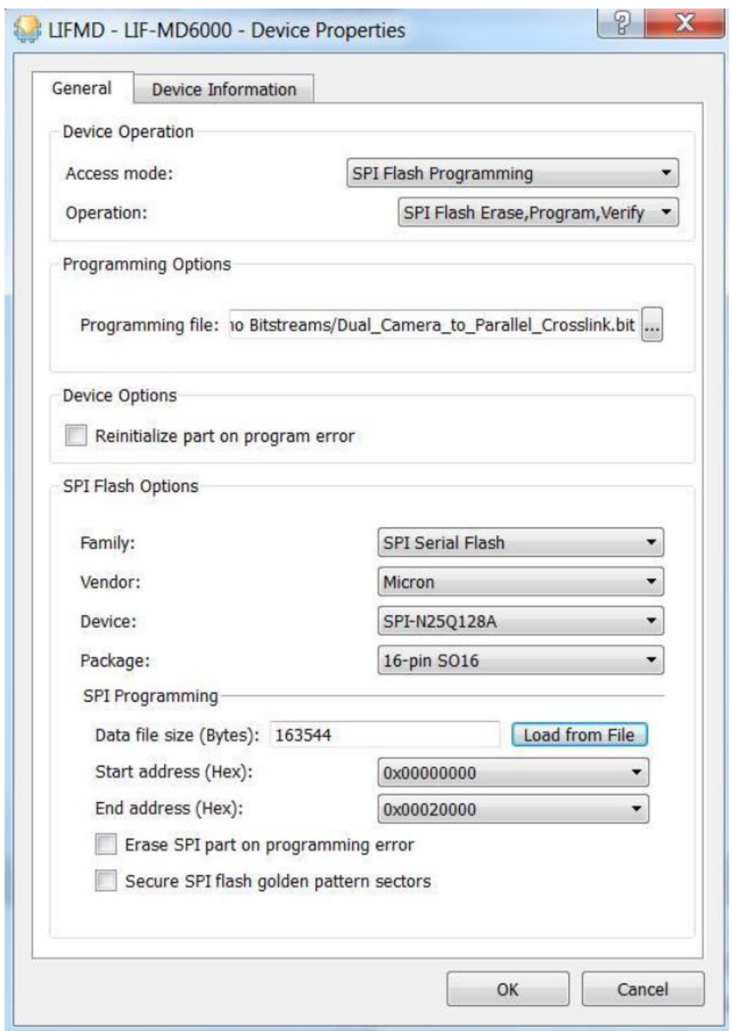


Figure 8.3. Device Properties

4. Click **OK** to close the **Device Properties** window.
5. Click the **Program** button  in Diamond Programmer to start the programming sequence.
6. After successful programming, the **Output** console displays the results as shown in Figure 8.4.

Figure 8.4. Output Console

8.2. Programming the ECP5 VIP Processor Board

8.2.1. Erasing the ECP5 Prior to Reprogramming

If the ECP5 VIP Processor Board and the CrossLink VIP Processor Board are already configured and programmed, erase first the ECP5 SRAM memory, then program the ECP5 SPI Flash in the next section. The demo design firmware must also be programmed onto the MicroSD card which is plugged into the MicroSD Card Adaptor Board.

Keep the board powered when re-programming the SPI Flash in the next section.

To erase the ECP5 device:

1. Launch Diamond Programmer with **Create a new blank project.**
2. Select **ECP5UM** for **Device Family** and **LFE5UM-85F** for **Device**.

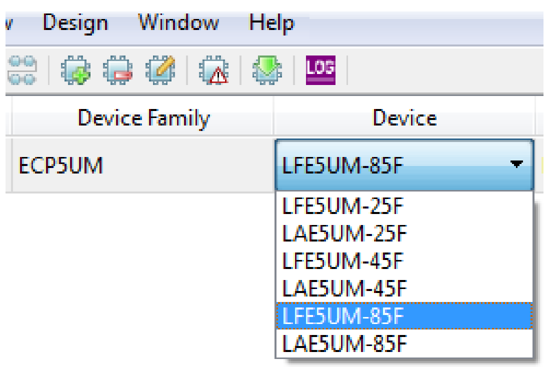


Figure 8.5. Selecting Device

3. Right-click and select **Device Properties**.
4. Select **JTAG 1532 Mode** for **Access Mode** and **Erase Only** for **Operation**.

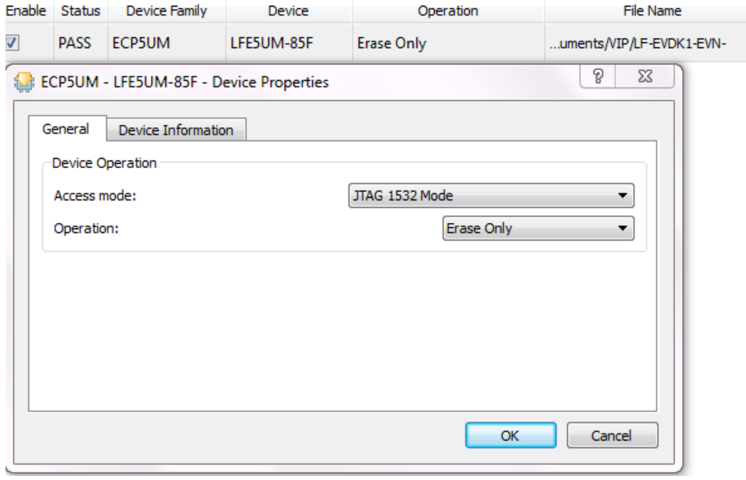


Figure 8.6. Device Operation

5. Click **OK** to close the Device Properties window.
6. Click the **Program** button  in Diamond Programmer to start the Erase sequence.

8.2.2. Programming the SPI on the ECP5 VIP Processor Board

To program the SPI:

1. Ensure the ECP5 device is erased by performing Steps 1-6.
2. Right-click and select **Device Properties**.
3. Select **SPI Flash Background Programming** for **Access mode** and make the following selections:
   a. For **Programming file**, browse and select the Human Count Demo bitfile (*.bit).
   b. For **SPI Flash Options**, refer to Table 8.2.

Table 8.2. SPI Flash Options Selection Guide

Item	Rev B	Rev C
Family	SPI Serial Flash	SPI Serial Flash
Vendor	Micron	Macronix
Device	SPI-N25Q128A	MX25L12835F
Package	8-pin SO8	8-Land WSON

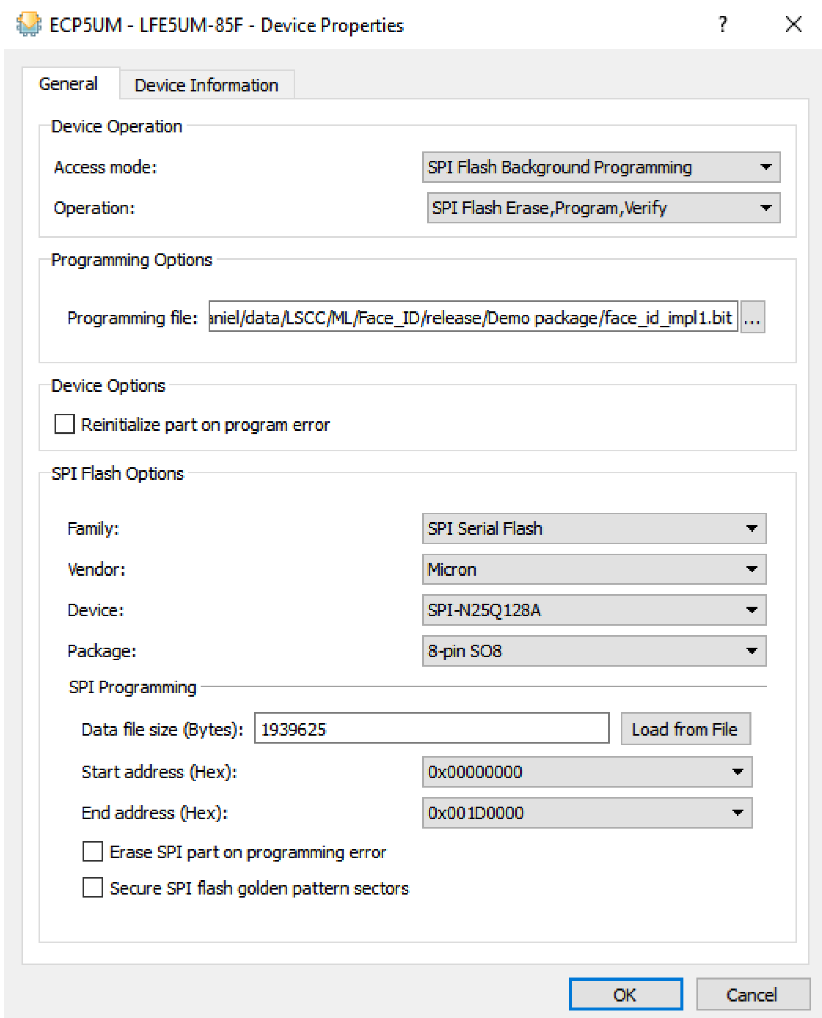


Figure 8.7 Device Properties

4. Click **OK** to close the **Device Properties** window.
5. Click the **Program** button in Diamond Programmer to start the programming sequence.
6. After successful programming, the **Output** console displays the results as shown in Figure 8.8.

Figure 8.8 Output Console

8.2.3. Programming the MicroSD Card Firmware

To write the image to the MicroSD card:

1. Download and install the Win32diskimager Image Writer software from the following link: https://sourceforge.net/projects/win32diskimager/.
2. Use Win32diskimager to write the appropriate Flash image file to the SD memory card. Depending on your PC, you may need a separate adapter (not described in this document) to physically connect to the card.
3. In Win32 Disk Imager, select the **Image File** and **Card Reader** as shown in Figure 8.9

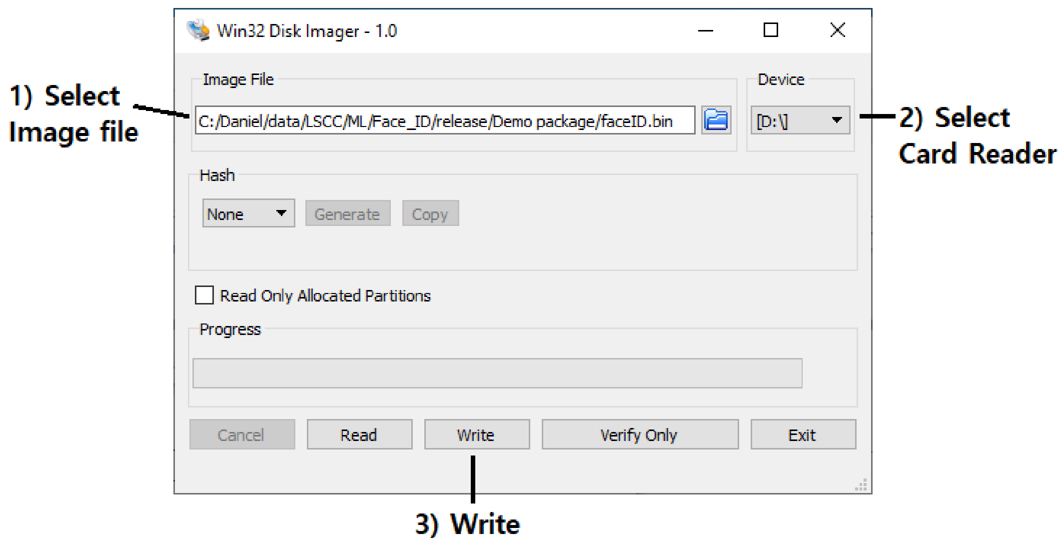


Figure 8.9 Win32 Disk Imager

4. Click **Write**.

9. Running the Demo

To run the demo:

1. Insert the configured MicroSD card into the MicroSD Card Adapter, and connect it to the Embedded Vision Development Kit.
2. Cycle the power on the Embedded Vision Development Kit to allow ECP5 and CrossLink to be reconfigured from Flash.
3. Connect the 1 × 4 membrane keypad to GND-D12 of J28 on the CrossLink VIP Bridge Board ass shown in Figure 9.1.

Figure 9.1 1 × 4 Membrane Keypad Attached to CrossLink VIP Bridge Board

4. Connect the Embedded Vision Development Kit to the HDMI monitor. The camera image should be displayed on the monitor as shown in Figure 9.2. Turn on *Guide marks* by pressing 2 in keypad as shown in Figure 9.2.

Figure 9.2 Face Identification Demo Initial Screen

5. Register your face by pressing 1 in the keypad.

Figure 9.3 Face Registration

A maximum of eight faces can be registered. The detected face index is displayed in *Face ID* as shown in Figure 9.4.

Figure 9.4 Face Detection

6. Press 3 on the keypad to clear all registered faces.

Appendix A. TensorFlow Installation Example without Using Anaconda

Note: Lattice cannot help on resolving all installation issues caused by system dependencies.

To install TensorFlow without using Anaconda:

1. Check the nvidia-smi installation and device model.

```
daniel@daniel:~$ nvidia-smi

Command 'nvidia-smi' not found, but can be installed with:

sudo apt install nvidia-340
sudo apt install nvidia-utils-390

daniel@daniel:~$ ubuntu-drivers devices
== /sys/devices/pci0000:00/0000:00:01.0/0000:01:00.0 ==
modalias : pci:v000010DEd00001C03sv0000174Bsd00002438bc03sc00i00
vendor   : NVIDIA Corporation
model    : GP106 [GeForce GTX 1060 6GB]
driver   : nvidia-driver-390 - distro non-free recommended
driver   : xserver-xorg-video-nouveau - distro free builtin
```

Figure A.1. NVIDIA-SMI

2. Run the command sudo apt install nvidia-utils-390.

   Note: NVIDIA driver and utils are dependent on the NVIDIA HW GPU version.

```
daniel@daniel:~$ sudo apt install nvidia-utils-390
[sudo] password for daniel:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libnvidia-compute-390
Suggested packages:
  nvidia-driver-390
The following NEW packages will be installed:
  libnvidia-compute-390 nvidia-utils-390
0 upgraded, 2 newly installed, 0 to remove and 144 not upgraded.
Need to get 20.9 MB of archives.
After this operation, 86.8 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us.archive.ubuntu.com/ubuntu bionic-updates/restricted amd64 libnvi
dia-compute-390 amd64 390.116-0ubuntu0.18.04.1 [20.6 MB]
Get:2 http://us.archive.ubuntu.com/ubuntu bionic-updates/restricted amd64 nvidia
-utils-390 amd64 390.116-0ubuntu0.18.04.1 [327 kB]
Fetched 20.9 MB in 30s (691 kB/s)
Selecting previously unselected package libnvidia-compute-390:amd64.
(Reading database ... 160035 files and directories currently installed.)
Preparing to unpack .../libnvidia-compute-390_390.116-0ubuntu0.18.04.1_amd64.deb
 ...
Unpacking libnvidia-compute-390:amd64 (390.116-0ubuntu0.18.04.1) ...
Selecting previously unselected package nvidia-utils-390.
Preparing to unpack .../nvidia-utils-390_390.116-0ubuntu0.18.04.1_amd64.deb ...
Unpacking nvidia-utils-390 (390.116-0ubuntu0.18.04.1) ...
Setting up libnvidia-compute-390:amd64 (390.116-0ubuntu0.18.04.1) ...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Setting up nvidia-utils-390 (390.116-0ubuntu0.18.04.1) ...
daniel@daniel:~$
```

Figure A.2 NVIDIA-Utils Installation

3. Check if nvidia-smi command works. Otherwise, remove the NVIDIA package. Run *sudo apt-get purge nvidia**.

```
daniel@daniel:~$ nvidia-smi
NVIDIA-SMI has failed because it couldn't communicate with the NVIDIA driver. Ma
ke sure that the latest NVIDIA driver is installed and running.

daniel@daniel:~$ sudo apt-get purge nvidia*
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'nvidia-325-updates' for glob 'nvidia*'
Note, selecting 'nvidia-346-updates' for glob 'nvidia*'
Note, selecting 'nvidia-driver-binary' for glob 'nvidia*'
Note, selecting 'nvidia-331-dev' for glob 'nvidia*'
Note, selecting 'nvidia-384-dev' for glob 'nvidia*'
Note, selecting 'nvidia-libopencl1-346-updates' for glob 'nvidia*'
```

Figure A.3 NVIDIA-SMI Command Checking

4. Install the latest driver and run the commands below:

```
sudo add-apt-repository ppa:graphics-drivers/ppa
```

```
Package 'nvidia-opencl-icd-331-updates' is not installed, so not removed
Package 'nvidia-opencl-icd-340-updates' is not installed, so not removed
Package 'nvidia-opencl-icd-384' is not installed, so not removed
The following package was automatically installed and is no longer required:
  libnvidia-compute-390
Use 'sudo apt autoremove' to remove it.
The following packages will be REMOVED:
  nvidia-utils-390*
0 upgraded, 0 newly installed, 1 to remove and 144 not upgraded.
After this operation, 1,015 kB disk space will be freed.
Do you want to continue? [Y/n] Y
(Reading database ... 160065 files and directories currently installed.)
Removing nvidia-utils-390 (390.116-0ubuntu0.18.04.1) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
daniel@daniel:~$ sudo add-apt-repository ppa:graphics-drivers/ppa
```

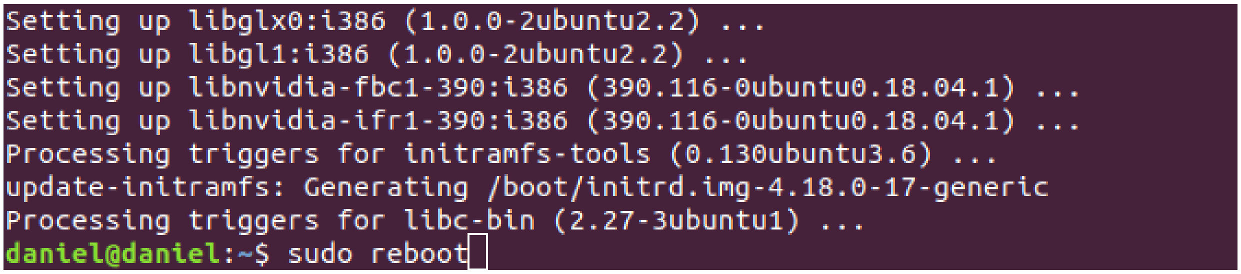
Figure A.4 NVIDIA Driver Installation

```
sudo apt-get upate
sudo apt-get install nvidia-390 nvidia-settings
```

```
Get:25 http://us.archive.ubuntu.com/ubuntu bionic-backports/universe amd64 DEP-1
1 Metadata [7,356 B]
Fetched 4,545 kB in 49s (93.0 kB/s)
Reading package lists... Done
daniel@daniel:~$ sudo apt-get update
Hit:1 http://us.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://ppa.launchpad.net/graphics-drivers/ppa/ubuntu bionic InRelease
Hit:3 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:4 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:5 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease
Reading package lists... Done
daniel@daniel:~$ sudo apt-get install nvidia-390 nvidia-settings
```

Figure A.5 NVIDIA Driver Installation

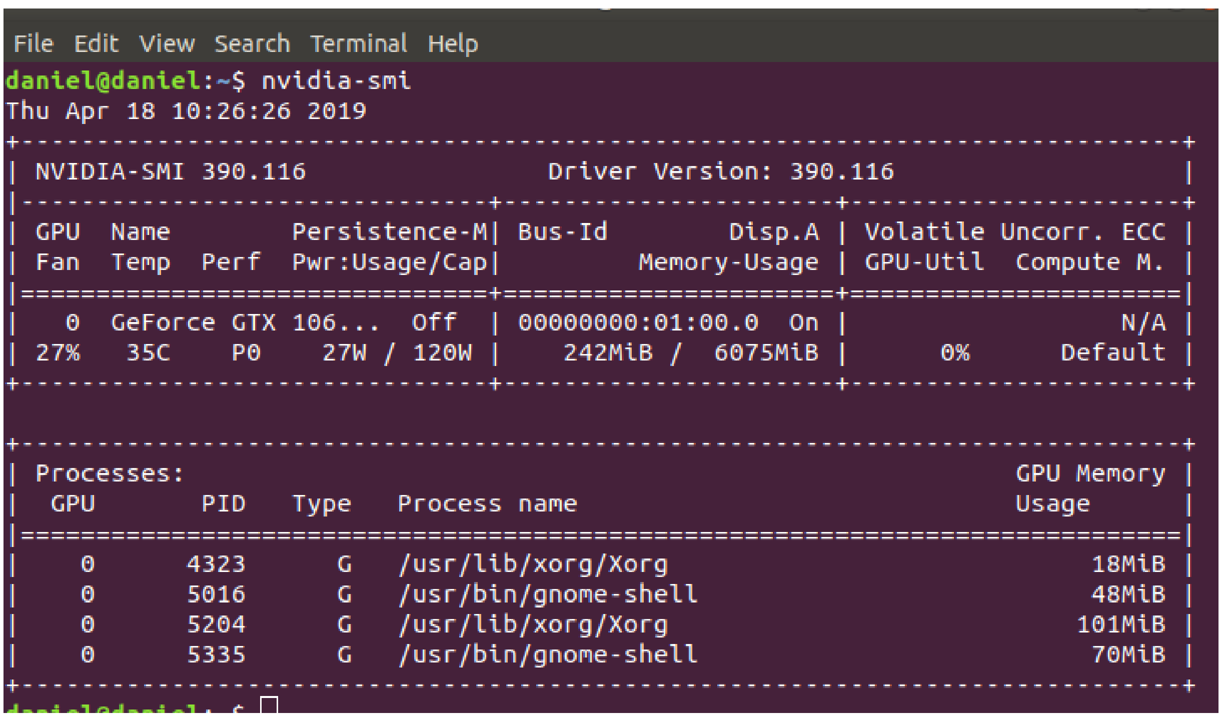
5. Reboot the system.



```
Setting up libglx0:i386 (1.0.0-2ubuntu2.2) ...
Setting up libgl1:i386 (1.0.0-2ubuntu2.2) ...
Setting up libnvidia-fbc1-390:i386 (390.116-0ubuntu0.18.04.1) ...
Setting up libnvidia-ifr1-390:i386 (390.116-0ubuntu0.18.04.1) ...
Processing triggers for initramfs-tools (0.130ubuntu3.6) ...
update-initramfs: Generating /boot/initrd.img-4.18.0-17-generic
Processing triggers for libc-bin (2.27-3ubuntu1) ...
daniel@daniel:~$ sudo reboot
```

Figure A.6 Reboot After NVIDIA Driver Installation

6. Check nvidia-smi.

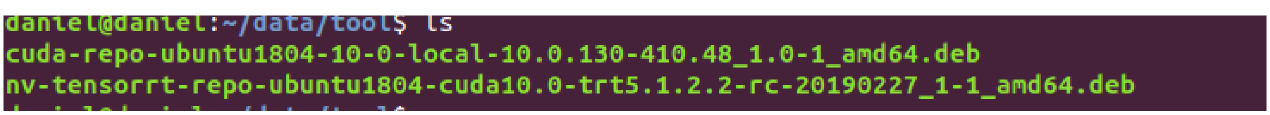


```
File  Edit  View  Search  Terminal  Help
daniel@daniel:~$ nvidia-smi
Thu Apr 18 10:26:26 2019
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 390.116                Driver Version: 390.116                   |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce GTX 106...  Off  | 00000000:01:00.0  On |                  N/A |
| 27%   35C    P0    27W / 120W |    242MiB /  6075MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0      4323      G   /usr/lib/xorg/Xorg                            18MiB |
|    0      5016      G   /usr/bin/gnome-shell                          48MiB |
|    0      5204      G   /usr/lib/xorg/Xorg                           101MiB |
|    0      5335      G   /usr/bin/gnome-shell                          70MiB |
+-----------------------------------------------------------------------------+
```

Figure A.7 Check NVIDIA-SMI

7. Download Cuda-repo and TensorRT package from NVIDIA development site.
Note: NVIDIA Cuda and TensorRT vesion needs to be matched to the Ubuntu version. For example, cuda-repo-ubuntu1804-10-0-local-10.0.130-410.48_1.0-1_amd64.deb nv-tensorrt-repo-ubuntu1804-cuda10.0-trt5.1.2.2-rc-20190227_1-1_amd64.deb.



```
daniel@daniel:~/data/tool$ ls
cuda-repo-ubuntu1804-10-0-local-10.0.130-410.48_1.0-1_amd64.deb
nv-tensorrt-repo-ubuntu1804-cuda10.0-trt5.1.2.2-rc-20190227_1-1_amd64.deb
```

Figure A.8 CUDA and TensorRT

8. Install CUDA and TensorRT.
   Note: Refer to the installation guide of NVIDIA CUDA and TensorRT for the detailed instructions.

```
daniel@daniel:~/data/tool$ ls
cuda-repo-ubuntu1804-10-0-local-10.0.130-410.48_1.0-1_amd64.deb
nv-tensorrt-repo-ubuntu1804-cuda10.0-trt5.1.2.2-rc-20190227_1-1_amd64.deb
daniel@daniel:~/data/tool$
daniel@daniel:~/data/tool$
daniel@daniel:~/data/tool$
daniel@daniel:~/data/tool$ sudo dpkg -i cuda-repo-ubuntu1804-10-0-local-10.0.130
-410.48_1.0-1_amd64.deb
Selecting previously unselected package cuda-repo-ubuntu1804-10-0-local-10.0.130
-410.48.
(Reading database ... 166930 files and directories currently installed.)
Preparing to unpack cuda-repo-ubuntu1804-10-0-local-10.0.130-410.48_1.0-1_amd64.
deb ...
Unpacking cuda-repo-ubuntu1804-10-0-local-10.0.130-410.48 (1.0-1) ...
Setting up cuda-repo-ubuntu1804-10-0-local-10.0.130-410.48 (1.0-1) ...

The public CUDA GPG key does not appear to be installed.
To install the key, run this command:
sudo apt-key add /var/cuda-repo-10-0-local-10.0.130-410.48/7fa2af80.pub

daniel@daniel:~/data/tool$ sudo apt-key add /var/cuda-repo-10-0-local-10.0.130-4
10.48/7fa2af80.pub
OK
daniel@daniel:~/data/tool$
```

Figure A.9 CUDA Installation

```
daniel@daniel:~/data/tool$ sudo apt-key add /var/cuda-repo-10-0-local-10.0.130-4
10.48/7fa2af80.pub
OK
daniel@daniel:~/data/tool$ sudo apt-get update
Get:1 file:/var/cuda-repo-10-0-local-10.0.130-410.48  InRelease
Ign:1 file:/var/cuda-repo-10-0-local-10.0.130-410.48  InRelease
Get:2 file:/var/cuda-repo-10-0-local-10.0.130-410.48  Release [574 B]
Get:2 file:/var/cuda-repo-10-0-local-10.0.130-410.48  Release [574 B]
Get:3 file:/var/cuda-repo-10-0-local-10.0.130-410.48  Release.gpg [833 B]
Get:3 file:/var/cuda-repo-10-0-local-10.0.130-410.48  Release.gpg [833 B]
Get:4 file:/var/cuda-repo-10-0-local-10.0.130-410.48  Packages [24.4 kB]
Hit:5 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:6 http://us.archive.ubuntu.com/ubuntu bionic InRelease
Hit:7 http://ppa.launchpad.net/graphics-drivers/ppa/ubuntu bionic InRelease
Hit:8 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:9 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease
Reading package lists... Done
daniel@daniel:~/data/tool$ sudo apt-get install cuda
```

Figure A.10 CUDA Installation

```
daniel@daniel:~/data/tool$ ls
cuda-repo-ubuntu1804-10-0-local-10.0.130-410.48_1.0-1_amd64.deb
nv-tensorrt-repo-ubuntu1804-cuda10.0-trt5.1.2.2-rc-20190227_1-1_amd64.deb
daniel@daniel:~/data/tool$ sudo dpkg -i nv-tensorrt-repo-ubuntu1804-cuda10.0-trt
5.1.2.2-rc-20190227_1-1_amd64.deb
[sudo] password for daniel:
Selecting previously unselected package nv-tensorrt-repo-ubuntu1804-cuda10.0-trt
5.1.2.2-rc-20190227.
(Reading database ... 180487 files and directories currently installed.)
Preparing to unpack nv-tensorrt-repo-ubuntu1804-cuda10.0-trt5.1.2.2-rc-20190227_
1-1_amd64.deb ...
Unpacking nv-tensorrt-repo-ubuntu1804-cuda10.0-trt5.1.2.2-rc-20190227 (1-1) ...
Setting up nv-tensorrt-repo-ubuntu1804-cuda10.0-trt5.1.2.2-rc-20190227 (1-1) ...
daniel@daniel:~/data/tool$
daniel@daniel:~/data/tool$ sudo apt-get update
```

Figure A.11 CUDA Installation

9. Check the TensorRT installation.

```
Reading package lists... Done
daniel@daniel:~/data/tool$
daniel@daniel:~/data/tool$ sudo apt-get install tensorrt libnvinfer-samples libc
udnn7
```

```
daniel@daniel:~/data/tool$ dpkg -l | grep TensorRT
ii  libnvinfer-dev                                        5.1.2-1+cuda10.0
                            amd64        TensorRT development libraries and hea
ders
ii  libnvinfer-samples                                    5.1.2-1+cuda10.0
                            all          TensorRT samples and documentation
ii  libnvinfer5                                           5.1.2-1+cuda10.0
                            amd64        TensorRT runtime libraries
ii  tensorrt                                              5.1.2.2-1+cuda10
.0                          amd64        Meta package of TensorRT
daniel@daniel:~/data/tool$
```

Figure A.12 TensorRT Installation

10. Install the library for TensorRT.

```
ii  libnvinfer5                                           5.1.2-1+cuda10.0
                            amd64        TensorRT runtime libraries
ii  tensorrt                                              5.1.2.2-1+cuda10
.0                          amd64        Meta package of TensorRT
daniel@daniel:~/data/tool$ sudo apt-get install python3-libnvinfer-dev
```

```
daniel@daniel:~/data/tool$ sudo apt-get install uff-converter-tf
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libnvidia-common-390 libwayland-client0:i386 libwayland-server0:i386
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  graphsurgeon-tf
```

Figure A.13 Library Installation for TensorRT

11. Check TensorRT and relevant package.

```
Setting up uff-converter-tf (5.1.2-1+cuda10.0) ...
daniel@daniel:~/data/tool$ dpkg -l | grep TensorRT
ii  graphsurgeon-tf                                          5.1.2-1+cuda10.0
                             amd64        GraphSurgeon for TensorRT package
ii  libnvinfer-dev                                           5.1.2-1+cuda10.0
                             amd64        TensorRT development libraries and hea
ders
ii  libnvinfer-samples                                       5.1.2-1+cuda10.0
                             all          TensorRT samples and documentation
ii  libnvinfer5                                              5.1.2-1+cuda10.0
                             amd64        TensorRT runtime libraries
ii  python3-libnvinfer                                       5.1.2-1+cuda10.0
                             amd64        Python 3 bindings for TensorRT
ii  python3-libnvinfer-dev                                   5.1.2-1+cuda10.0
                             amd64        Python 3 development package for Tenso
rRT
ii  tensorrt                                                 5.1.2.2-1+cuda10
.0                           amd64        Meta package of TensorRT
ii  uff-converter-tf                                         5.1.2-1+cuda10.0
                             amd64        UFF converter for TensorRT package
daniel@daniel:~/data/tool$
```

Figure A.14 Check TensorRT Installation.

12. Check the python version.

```
daniel@daniel:~$
daniel@daniel:~$ python

Command 'python' not found, but can be installed with:

sudo apt install python3
sudo apt install python
sudo apt install python-minimal

You also have python3 installed, you can run 'python3' instead.

daniel@daniel:~$ python3
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()
```

Figure A.15 Python Version

13. Check the pip installation and run the pip command.

```
daniel@daniel:~$
daniel@daniel:~$ pip3 --version

Command 'pip3' not found, but can be installed with:

sudo apt install python3-pip

daniel@daniel:~$ sudo apt install python3-pip
```

Figure A.16 PIP Installation

14. Check the pip version.

```
Setting up python3-dev (3.6.7-1~18.04) ...
daniel@daniel:~$ pip3 --version
pip 9.0.1 from /usr/lib/python3/dist-packages (python 3.6)
daniel@daniel:~$
```

Figure A.17 Check PIP Version

15. Check the virtualenv installation.

```
daniel@daniel:~$ virtualenv --version

Command 'virtualenv' not found, but can be installed with:

sudo apt install virtualenv
```

Figure A.18 Virtualenv Installation Checking

16. Install virtualenv.

```
daniel@daniel:~$ sudo apt install virtualenv
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  python3-virtualenv
The following NEW packages will be installed:
  python3-virtualenv virtualenv
0 upgraded, 2 newly installed, 0 to remove and 133 not upgraded.
Need to get 47.8 kB of archives.
After this operation, 171 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Figure A.19 Install Virtualenv

17. Check virtualenv installation and update

```
Setting up virtualenv (15.1.0+ds-1.1) ...
daniel@daniel:~$ virtualenv --version
15.1.0
daniel@daniel:~$
```

```
daniel@daniel:~$ sudo apt update
Hit:1 http://us.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:5 http://ppa.launchpad.net/graphics-drivers/ppa/ubuntu bionic InRelease
Reading package lists... 88%
```

Figure A.20 Virtualenv and Update Checking

18. Create a vitualenv environment.

```
daniel@daniel:~$
daniel@daniel:~$ ls
data     Documents  examples.desktop  Pictures  Templates
Desktop  Downloads  Music             Public    Videos
daniel@daniel:~$ virtualenv --system-site-packages -p python3 ./venv
```

Figure A.21 Virtualenv Environment Creation

19. Check the '*venv*' folder and activate virtual mode.

```
daniel@daniel:~$ ls
data     Documents  examples.desktop  Pictures  Templates  Videos
Desktop  Downloads  Music             Public    venv
daniel@daniel:~$
```

Figure A.22 Check Venv Creation

```
daniel@daniel:~$ source ./venv/bin/activate
(venv) daniel@daniel:~$
```

Figure A.23 Activate Virtualenv

20. Install TensorFlow-GPU.

```
(venv) daniel@daniel:~$ pip install tensorflow-gpu
Collecting tensorflow-gpu
  Downloading https://files.pythonhosted.org/packages/7b/b1/0ad4ae02e17ddd62109c
d54c291e311c4b5fd09b4d0678d3d6ce4159b0f0/tensorflow_gpu-1.13.1-cp36-cp36m-manyli
nux1_x86_64.whl (345.2MB)
    3% |                               | 13.8MB 117kB/s eta 0:47:05
```

Figure A.24 TensorFlow-GPU Installation

21. Check TensorFlow Installation.

```
(venv) daniel@daniel:~$ python
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> exit()
(venv) daniel@daniel:~$ pip show tensorflow
(venv) daniel@daniel:~$ pip list | grep tensorflow
tensorflow-estimator    1.13.0
tensorflow-gpu          1.13.1
```

Figure A.25 TensorFlow Installation Checking

22. Install OpenCV.

```
(venv) daniel@daniel:~$ pip install opencv-python
Collecting opencv-python
  Downloading https://files.pythonhosted.org/packages/7b/d2/a2dbf83d4553ca6b3701
d91d75e42fe50aea97acdc00652dca515749fb5d/opencv_python-4.1.0.25-cp36-cp36m-manyl
```

Figure A.26 OpenCV-Python Installation

23. Install easydict and joblib.

```
(venv) daniel@daniel:~$ pip install easydict
Collecting easydict
  Downloading https://files.pythonhosted.org/packages/4c/c5/5757886c4f538c1b3f95
f6745499a24bffa389a805dee92d093e2d9ba7db/easydict-1.9.tar.gz
Building wheels for collected packages: easydict
  Building wheel for easydict (setup.py) ... done
  Stored in directory: /home/daniel/.cache/pip/wheels/9a/88/ec/085d92753646b0eda
1b7df49c7afe51a6ecc496556d3012e2e
Successfully built easydict
Installing collected packages: easydict
Successfully installed easydict-1.9
(venv) daniel@daniel:~$ pip install joblib
```

Figure A.27 Easydict and Joblib Installation

24. Install Keras python.

```
Successfully installed joblib-0.13.2
(venv) daniel@daniel:~$ pip install Keras
```

Figure A.28 Keras Installation

Appendix B. Caffe Compilation Example

For more information on how to install Caffe, go to https://caffe.berkeleyvision.org/.

Note: Caffe has some dependencies of environment and library packages, Lattice cannot help on resolving all installation issues caused by system dependencies.

To compile Caffe:

1. Go to the Caffe source files and directory.

```
daniel@daniel:~/data/Human_ID/caffe-face/caffe-face$ ls
caffe.cloc        docker         LICENSE                  mnist_example  tools
cmake             docs           LICENSE-Caffe            models         train
CMakeLists.txt    examples       Makefile                 python
CONTRIBUTING.md   face_example   Makefile.config          README.md
CONTRIBUTORS.md   include        Makefile.config.example  scripts
data              INSTALL.md     matlab                   src
daniel@daniel:~/data/Human_ID/caffe-face/caffe-face$
```

Figure B.1 Caffe OpenSource Structure

2. Run the script *make all.*

```
daniel@daniel:~/data/Human_ID/caffe-face/caffe-face$ make all
PROTOC src/caffe/proto/caffe.proto
CXX .build_release/src/caffe/proto/caffe.pb.cc
CXX src/caffe/util/db_leveldb.cpp
CXX src/caffe/util/insert_splits.cpp
CXX src/caffe/util/benchmark.cpp
CXX src/caffe/util/cudnn.cpp
CXX src/caffe/util/io.cpp
CXX src/caffe/util/upgrade_proto.cpp
CXX src/caffe/util/db.cpp
```

Figure B.2 Run Make File

3. Run the script *make test.*

```
daniel@daniel:~/data/Human_ID/caffe-face/caffe-face$
daniel@daniel:~/data/Human_ID/caffe-face/caffe-face$ make test
CXX src/caffe/test/test_split_layer.cpp
CXX src/caffe/test/test_gradient_based_solver.cpp
```

Figure B.3 Run Make-Test

4. Run the script *make runtest.*

```
daniel@daniel:~/data/Human_ID/caffe-face/caffe-face$
daniel@daniel:~/data/Human_ID/caffe-face/caffe-face$ make runtest
```

```
[----------] 5 tests from DeconvolutionLayerTest/2, where TypeParam = caffe::GPUDevice<float>
[ RUN      ] DeconvolutionLayerTest/2.TestGradient
[       OK ] DeconvolutionLayerTest/2.TestGradient (6509 ms)
[ RUN      ] DeconvolutionLayerTest/2.TestNDAgainst2D
[       OK ] DeconvolutionLayerTest/2.TestNDAgainst2D (24 ms)
[ RUN      ] DeconvolutionLayerTest/2.TestGradient3D
[       OK ] DeconvolutionLayerTest/2.TestGradient3D (1047 ms)
[ RUN      ] DeconvolutionLayerTest/2.TestSetup
[       OK ] DeconvolutionLayerTest/2.TestSetup (0 ms)
[ RUN      ] DeconvolutionLayerTest/2.TestSimpleDeconvolution
[       OK ] DeconvolutionLayerTest/2.TestSimpleDeconvolution (1 ms)
[----------] 5 tests from DeconvolutionLayerTest/2 (7581 ms total)

[----------] Global test environment tear-down
[==========] 1963 tests from 261 test cases ran. (255204 ms total)
[  PASSED  ] 1963 tests.
daniel@daniel:~/data/Human_ID/caffe-face/caffe-face$
```

Figure B.4 Run Make-Runtest

Technical Support Assistance

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

Revision History

Revision 1.0, August 2019

Section	Change Summary
All	Initial release

www.latticesemi.com