CrossLink-NX Auto ML based Object Classification Reference Design

Reference Design

FPGA-RD-02258-1.0

June 2022

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, with all faults and associated risk the responsibility entirely of the 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
CNN	Convolutional Neural Network
FPGA	Field-Programmable Gate Array
ML	Machine Learning
CBSR	Convolution + Batch Normalization + Scale + Relu
NNC	Neural Network Compiler

1. Introduction

This document describes how AutoKeras is utilized to get a best model architecture out of set of various architectures explored by AutoKeras and train the model which can be compatible with Lattice NNC software for handgesture application.

1.1. AutoKeras for Lattice NNC

Autokeras can be utilized to select a classification model structure which is compatible with Lattice NNC. To run a CNN model on certain Lattice devices, the model is required to have 8-bit kernel/weight quantization and/or 8-bit activation quantization. This is achieved by using custom layers in Autokeras. A custom Convolution and a custom Dense layer is used for this purpose which also achieves a CBSR (Convolution + Batch Normalization + Scale + Relu) and alternate MaxPool structures.

2. Software and Hardware requirement

Since this document talks about selecting and training neural network for Handgesture detection application using AutoKeras, for the actual use of the trained model, please refer to UltraPlus Hand Gesture Detection document

If there are any differences from the above original documentation it will be mentioned as a part of this current document.

Lattice Neural Network Compiler (NNC) Version 5.0 is used to create the binary file (UltraPlus Hand Gesture Detection document uses Version 3.1).

3. Project Structure

This project is mainly used to achieve following four tasks.

1. Creating environment for AutoKeras
2. Preparing Dataset
3. Training using AutoKeras
4. Evaluating the best model exported by AutoKeras

See the directory structure below to achieve the four above mentioned tasks. How to use these folders and subfolders will be explained in the later chapters of this document.

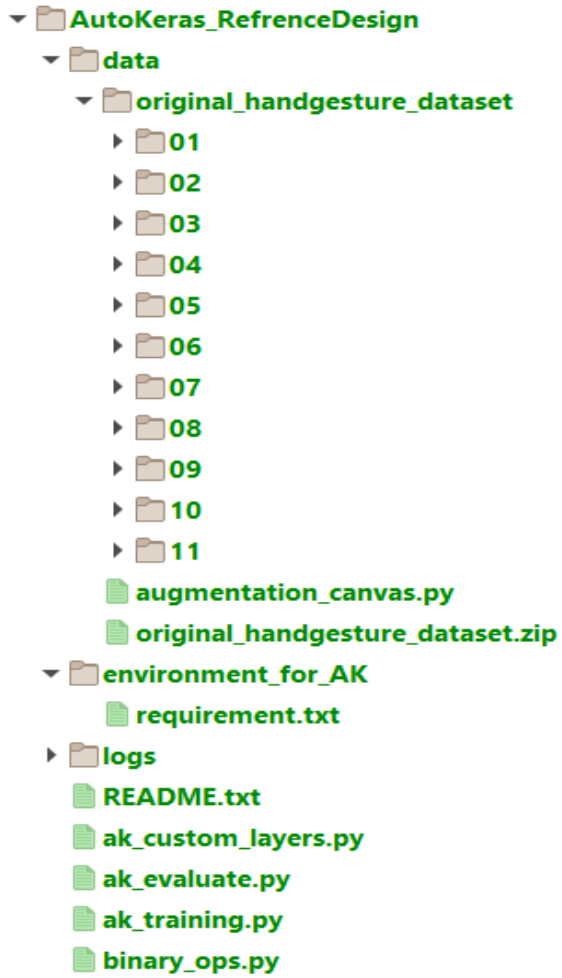


Figure 3.1. Directory structure

4. Creating environment for AutoKeras

4.1. Machine Requirements

The user can either create an environment by following the steps below or can directly use sensAI studio for AutoKeras training . To use the later method, please refer to the Lattice sensAI studio documentation. The experiments were performed on 64-bit Ubuntu 18.04 with NVIDIA GPU drivers and/or libraries, with CUDA version 10.1. The Python version used was 3.6.9.

4.2. Environment for AutoKeras

The virtual environment can be created using Python venv by using the command

```
python3 –m venv path/to/new/virtual/env
```

To activate this environment run,

```
source venv path/to/new/virtual/env/bin/activate
```

The required packages can be installed using pip method. The required packages are mentioned in the folder *environment_for_AK/requirement.txt* file and user can install them after activating the virtual environment created above by using the following command

```
python3 –m pip install –r requirement.txt
```

After running the last command above, Tensorflow version 2.3.0 and AutoKeras version 1.0.15 and rest required packages will be installed.

5. Preparing Dataset

5.1. Creating Dataset

An example dataset is included with this reference design for user to test AutoKeras with. But if user wants to create her/his own handgesture dataset, please refer to section 3.1 Creating of dataset of UltraPlus Hand Gesture Detection document

5.2. Dataset Augmentation

Data Augmentation is not done in this project as a part of AutoKeras. Augmentation is a standalone script which is similar one as the augmentation in section 4.2 Data Augmentation of UltraPlus Hand Gesture Detection document . For ease of use, that script is included here. This script will rotate, zoom and shift the input images. An example of image augmentation can be seen Figure 5.1 below.

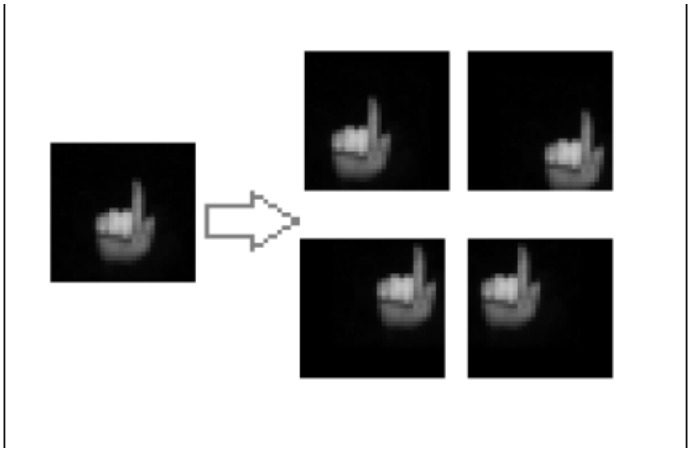


Figure 5.1. Data Augmentation example

The toy dataset for handgesture is provided for user in the folder *data/original_handgesture_dataset.zip*. Once this file is extracted you will have a folder *original_handgesture_dataset* with eleven subfolders inside it as seen in Figure 5.1. Folder 01,02,03,04,05,06,07,08,09,10 are different hand gestures and class 11 or folder 11 is a background class. This classification training expects different class of images in different folder. This dataset now will be used to create the augmented dataset. To start augmentation run,

```
python3 ./data/augmentation_canvas.py --
input_dir=./data/original_handgesture_dataset - output_dir=./data/augmented_dataset
```

By running the command given above, the augmentation_canvas script augments the data and stores inside *data/augmented_dataset*.

For AutoKeras training, the data is expected in tf.data.dataset format, hence unlike the original handgesture model training, the data is not converted to tfrecord format. We use image_dataset_from_directory method of AutoKeras to load our data in batches. Please note that this method takes the name of the folders in alphanumeric form. i.e; for example, say your class names (name of those subfolders of different classe images) are 02,10,11,8,09 then it's alphanumeric order is going to be 02,09,10,11,8. So to have the ten classes of hand gestures as first ten outputs of FC/Dense layer in order, the folder names are given in alphanumeric order as seen in Figure 5.1.

6. Training using AutoKeras

To start the training with default parameters run

```
python3 ak_training.py --data_dir=./data/augmented_dataset --log_dir=./logs --total_train_images=755357
```

The above command will start the Neural Architectural Search with AutoKeras and create multiple architectures (The parameter max_trials will decide how many variations to be explored). Each of these architectures will be trained for some small number of epochs and the architecture with least validation loss will be picked and trained for more number of epochs. All these architectures will be stored in an automatic created folder *auto_model* .

While training, the dataset from *data_dir* is split into 80-20 train and validation dataset (so the parameter *total_train_images* is ceil of 80% of the total images in *augmented_dataset* folder) and the input is scaled by 1./128 to make the input range from 0.0 to 2.0.

The final model after training is stored as best_model.h5 inside *logs* folder.

Following are some of the parameters in ak_training script:

- data_dir : dataset directory which has eleven class subdirectories
- log_dir : Directory to export best model of autokeras
- total_train_images : Total train images used for training. Required to calculate Learning Rate at different epochs.
- image_resolution : Input image resolution (Height or width)
- color_mode : “grayscale” for grayscale images or “rgb” for color images
- quantrelu : Activation Quantization
- kernel_quant : Kernel quantization
- epochs : Total number of epochs to train the best model.
- max_trials : Maximum number of Neural Architectures for AutoKeras to explore in order to select the best model from.
- over_write : When False, if training gets interrupted in between and restarted, then it resumes it’s architecture search from the place it was stopped (but please make sure not to delete auto_model folder which). However, it will not resume the training of the final/best model selected by AutoKeras.

Once the training is done, the model will be exported to log_dir with the name best_model.h5. However, this model has last layer as softmax activation. Hence we trim this model to last but one, dense layer and save it as intm.h5 in the log directory. Use this intm.h5 model to create binary file with Lattice NNC. The other model with name like intrm_model_loss_0.****_acc_****.h5 is the same as intm.h5 but the name is stored with final validation loss and validation accuracy value (The 20% validation data which was generated during Training). (Note : if validation accuracy in the model name is shown as 0.93766 that means the validation accuracy is 93.766%)

7. Evaluate the final model

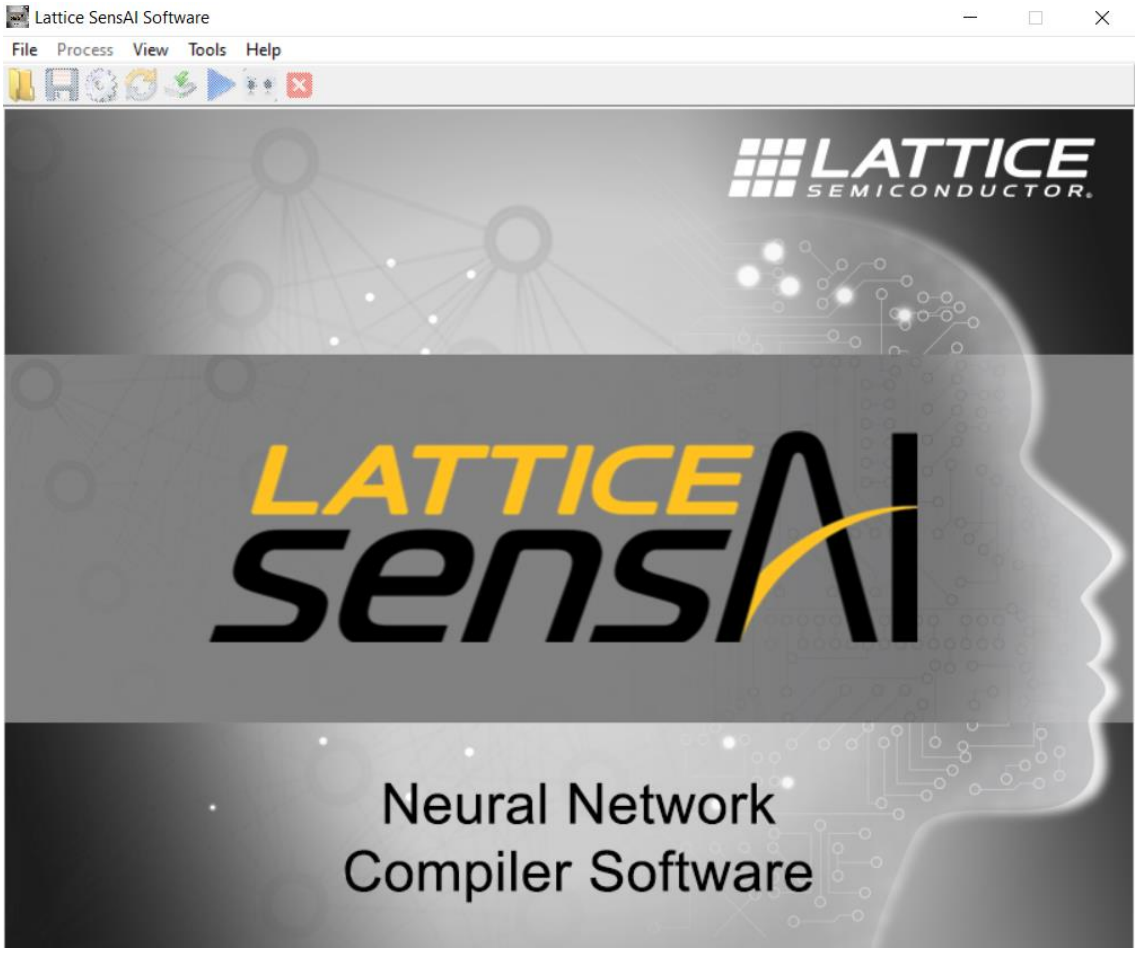
To evaluate the final model on Validation data, run

```
python3 ak_evaluate.py
```

This command will evaluate loss and accuracy of *augmented_dataset* . This script is also not a part of AutoKeras. It only uses modules from tensorflow.keras to load and evaluate the model exported by AutoKeras. To use your own validation dataset path, use the *–data_dir* argument from the *ak_evaluate.py* script.

8. Creating Binary file with Lattice NNC

This chapter describes how to generate binary file using the Lattice NNC version 5.0 program.

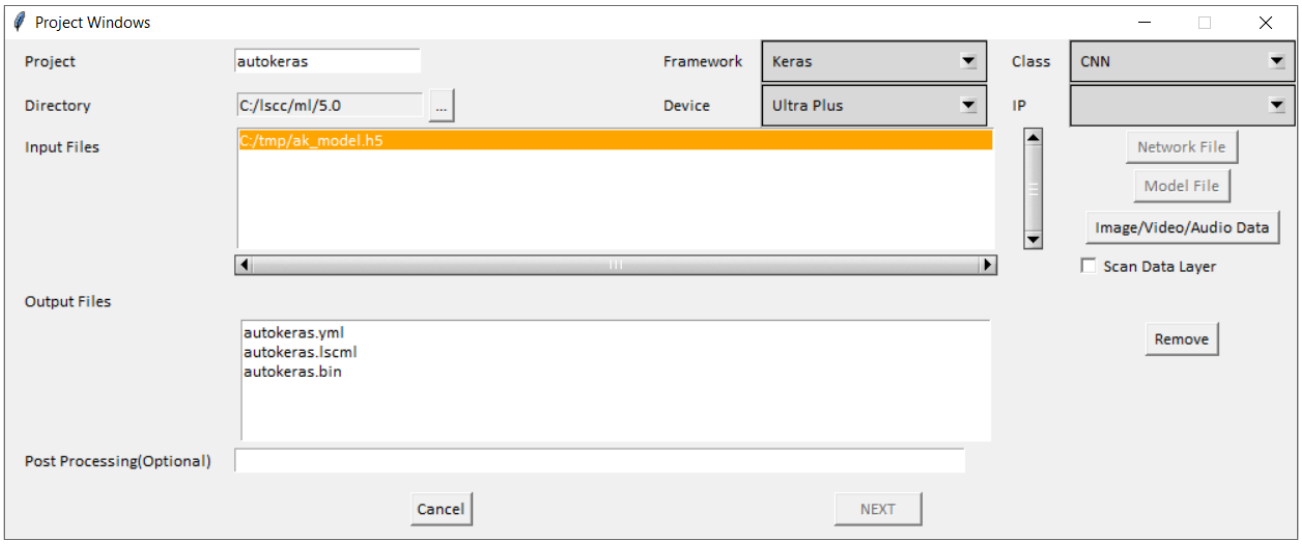


To create the project in SensAI tool:

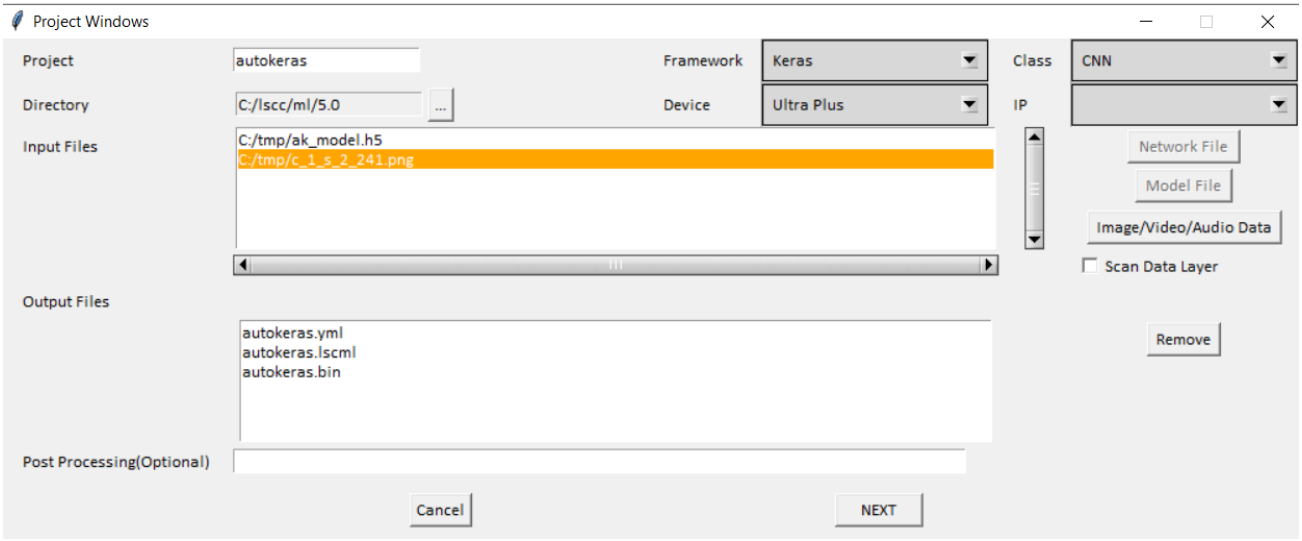
1. Click File > New.
2. Enter the following settings:

- Project name
- **Framework** – **Keras**
- **Class** – **CNN**
- **Device** – **UltraPlus**

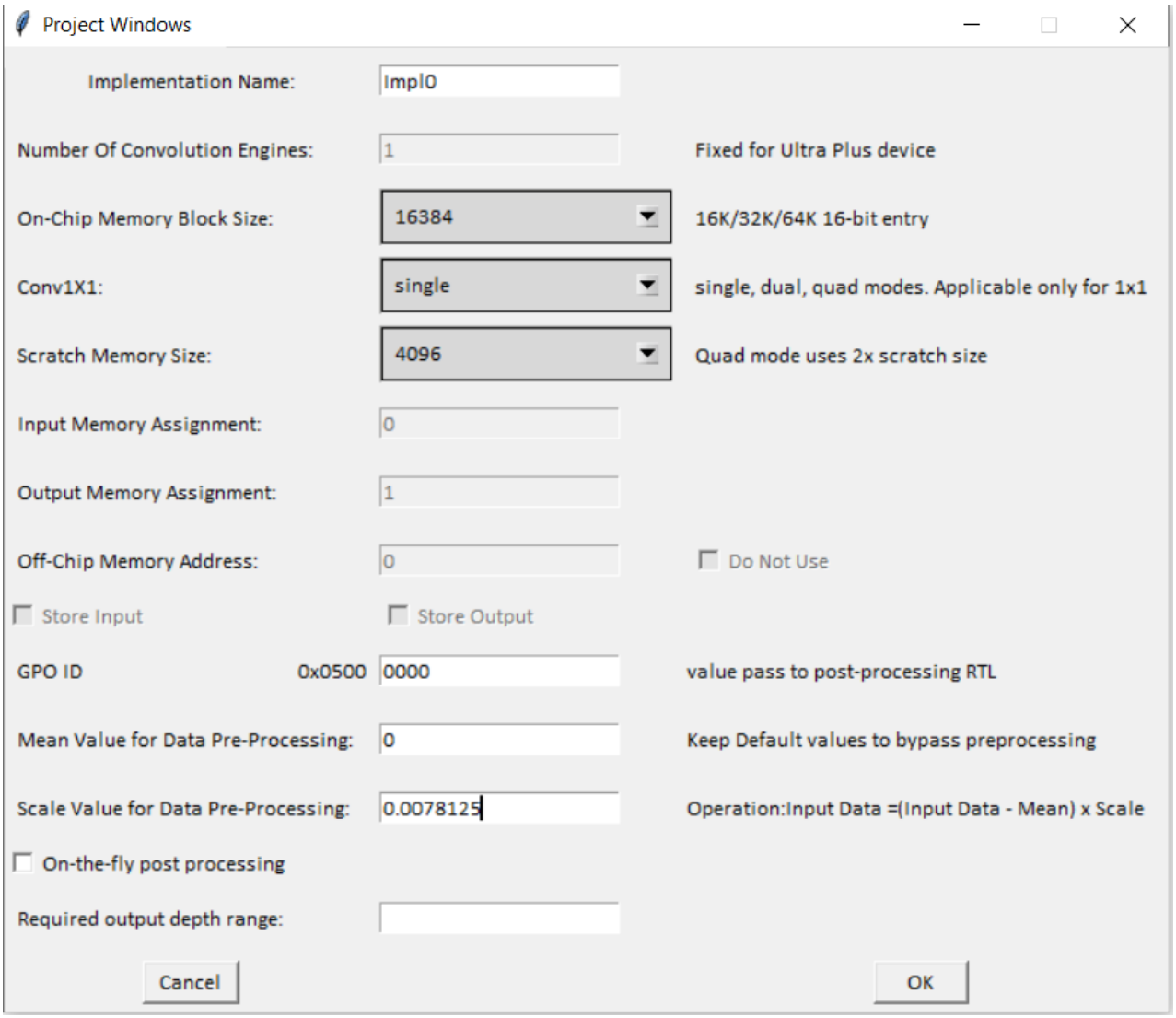
3. Click **Network File** and select the network (.h5) file.



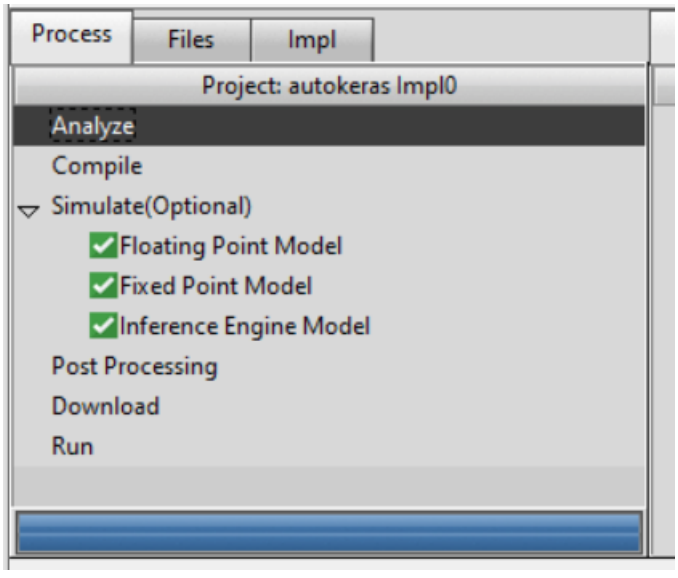
4. Click **Image/Video/Audio Data** button and select your image input file



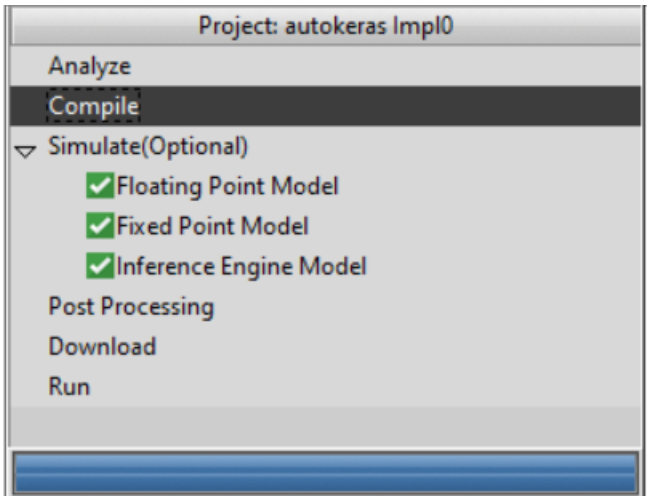
5. Click **Next**.
6. Select the following attributes :
    a. Mean Value for Data Pre-Processing : **0**
    b. Scale Value for Data Pre-Processing : **0.0078125**



7. Click **OK** to create the project.
8. Double click on **Analyze**



9. Double click on **Compile** to generate firmware and filter binary file



Firmware bin file location will be displayed in the compilation log. Use generated firmware bin on hardware for testing.

9. Hardware Implementation

Please refer to section 7 of UltraPlus Hand Gesture Detection document

10. Creating FPGA bitstream file

Please refer to section 8 of UltraPlus Hand Gesture Detection document

11. Programming and running the demo

Please refer to sections 9 and 10 of UltraPlus Hand Gesture Detection document

12. Limitations and Restrictions

These limitations are general limitations while using AutoKeras for Lattice NNC and not specific to this reference design.

1. The model training was done considering multiclass “CLASSIFICATION” task only.
2. The model architectures were experimented with input size of 32x32x1.
3. The optimizer which Autokeras chooses sometimes has a very small initial learning rate and sometimes it is used along with learning rate decay and this affects training accuracy/loss. Hence a constant optimizer was used (SGD with initial LR=0.1 and with a learning rate scheduler callback option)
4. For now the only hyperparameter which is varying is, the number of channels (depth) in each layer. If the “number of layers” is kept as a hyperparameter then it tries to go for a very large depth near FC layer and this creates FC output value exploding. So the number of layers are for now fixated.
5. “Max model size” parameter is tested with few experiments (with given seed and resolution) to create a model (.bin file size) smaller than the limit for certain devices like UltrPlus.

For Reproducibility, when the seed is provided, it searches through same hyperparameter combinations every time we run the script. However, the loss value which the AutoKeras get might differ slightly and as a result may not give the same architecture as earlier. But the accuracy remains approximately within +/- 3% range.

References

- Customized models using AutoKeras

Technical Support Assistance

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

Revision History

Revision 1.0, June 2022

Section	Change Summary
All	Initial release.

www.latticesemi.com