NETINT AI Toolkit

Quick Start Guide V4.5

NETINT Technologies Inc.

Table of Contents

1. Legal Notices

Information in this document is provided in connection with NETINT products. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted by this document. Except as provided in NETINT's terms and conditions of sale for such products, NETINT assumes no liability whatsoever and NETINT disclaims any express or implied warranty, relating to sale and/or use of NETINT products including liability or warranties relating to fitness for a particular purpose, merchantability, or infringement of any patent, copyright, or other intellectual property right.

A "Mission Critical Application" is any application in which failure of the NETINT Product could result, directly or indirectly, in personal injury or death. Should you purchase or use NETINT's products for any such mission critical application, you shall indemnify and hold NETINT and its subsidiaries, subcontractors and affiliates, and the directors, officers, and employees of each, harmless against all claims costs, damages, and expenses and reasonable attorney's fees arising out of, directly or indirectly, any claim of product liability, personal injury, or death arising in any way out of such mission critical application, whether or not NETINT or its subcontractor was negligent in the design, manufacture, or warning of the NETINT product or any of its parts.

NETINT may make changes to specifications, technical documentation, and product descriptions at any time, without notice. The information here is subject to change without notice. Do not finalize a design with this information. The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications.

NETINT™, Codensity™, Quadra™ and the NETINT Logo are trademarks of NETINT Technologies Inc. All other trademarks or registered trademarks are the property of their respective owners.

2. Introduction

NETINT provides high density and efficient video transcoding solutions using the powerful video processing engines inside our Codensity G5 Application-Specific Integrated Circuit (ASIC). We can provide multiple stream transcoding functions and services, directly to video content providers. We can also provide Transcoding as a Service (TaaS), which can be integrated into video streaming systems and services. Our functions and services can be used for highly efficient Video-on-Demand file transcoding, as well as real-time, live video streaming applications.

The NETINT AI Toolkit offers an Artificial Intelligence development environment for the user to translate, quantize, inference, export deep learning models, and to deploy on NETINT Neural Network Processing Units (NETINT NPUs). For more detailed information, please refer to the complete NETINT AI Toolkit User Guide.

3. Setup the NETINT AI Development Environment

This section describes the NETINT AI environment setup and toolkit installation.

Prerequisites

- The host system meets the requirements as described in NETINT AI Toolkit User Guide.
- Ubuntu 20.04 LTS 64-bit with Python 3.8 (recommended).

Procedure

1. Updating Python to the recommended version (python3.8.10).
   ```
   sudo apt install python3.8.10
   ```
2. Installing and updating pip to the specified version.
   ```
   sudo apt install python3-pip

   python3 -m pip install --user --upgrade pip==21.0.1
   ```
3. Unpacking <*NETINT_toolkit*> package to the home directory.
   ```
   tar -zxvf <NETINT_toolkit>.tar.gz -C ~
   ```
4. Install dependencies.
   ```
   pip3 install -r ~/NETINT_toolkit/requirements.txt
   ```
5. Installing toolkit CPU wheel package.
   ```
   pip3 install ~/NETINT_toolkit/installation_packages/acuity_bin_cpu-*.whl
   ```
6. Installing patch for IDE.
   ```
   sudo apt install libcanberra-gtk-module libcanberra-gtk3-module
   ```
7. Setup environment variables.
   ```
   ~/NETINT_toolkit/tools/setup_bashrc.py

   source ~/.bashrc
   ```
8. Verifying the development environment.
   ```
   ~/NETINT_toolkit/tools/envcheck.py
   ```

4. Pipeline Walkthrough

The NETINT AI Toolkit provides the complete solution for importing, quantizing, validating, and exporting deep learning models for NETINT NPUs. This section provides a general workflow for using the NETINT AI Toolkit as a tool to convert the classification model (Lenet using TensorFlow). More details of the workflow and some complete examples can be found in the NETINT AI Toolkit User Guide.

The Lenet example is located at:

```
~/NETINT_toolkit/examples/lenet_tf/
```

4.1. Import the Model

Importing the Lenet model based on TensorFlow framework (*lenet_tf.pb*) to unified format

```
cd ~/NETINT_toolkit/examples/lenet_tf/
./convert.sh -c
```

Three files, *lenet_tf.data*, *lenet_tf.json*, and *lenet_tf_inputmeta.yml*, will be generated.

4.2. Validate the Imported Model

Execute the following command to validate the imported model (float32):

```
./convert.sh -v
```

It will run the test data listed in *dataset.txt* and save the results into *raw_data_f* folder. The results contain one input tensor file (data after preprocessing), and one/more output tensor files.

4.3. Quantize the Imported Model

Execute the following command to quantize the imported model:

```
./convert.sh -q
```

It will produce a quantized model based on quantize-parameters and a calibration dataset (e.g., dataset.txt). The quantized results will be saved as *lenet_tf.quantize*.

4.4. Validate the Quantized Model

Execute the following command to validate the quantized model:

```
./convert.sh -vq
```

The test data listed in dataset.txt will be inferenced, and the generated output tensor files will be saved into *raw_data_i*. The results often contain one input tensor file (data after preprocessing), and one/more output tensor files.

4.5. Export to OpenVX Model

Execute the following command to export the OpenVX model:

```
./convert.sh -o
```

It will generate all the source code for obtaining OpenVX model. Such files are saved in the *./ovx* folder.

4.6. Compile the OpenVX Model

Execute the following command to compile the OpenVX model:

```
cd ./ovx
make -f makefile.linux
```

It will generate a binary executable file.

4.7. Evaluate the OpenVX Model

Execute *run_c_model.py* with the following commands:

```
cd ..
./run_c_model.py -a
```

It will run all input data listed in dataset.txt and generate tensor output files. The results normally contain one input tensor file (data after preprocessing), and one/more output tensor files. All files are saved into *raw_data_c* folder.

Note: you may need to modify *run_c_model.py* according to the model requirements.

4.8. Model Post-processing

Execute the following command to produce final results based on raw_data_x:

```
./postprocess_tensor.py -F
./postprocess_tensor.py -I
./postprocess_tensor.py -C
```

The final results are saved in the folder *./results* for comparison.

Note: you may need to modify *postprocess_tensor.py* according to the model requirements.

4.9. Export an OpenVX Network Binary Graph

Execute the following command to export the OpenVX Network Binary Graph (NBG):

```
./convert.sh -e
```

It will generate *.nb (NBG file) as well as source code. All of them are saved in the `./ovx_network_nbg_unify/` folder.

4.10. Inference with the Network Binary Graph on the Chip

Before executing the following command, you need to build Xcoder Codec Library and compile the aiperf tool (Refer to NETINT AI Toolkit User Guide for more detailed information).

Execute the following command to evaluate the NBG as follows:

```
sudo ./aiperf -network_binary ./lenet_tf.nb -input_dir ./dataset.txt -
       output_dir ./output -file_type batch
```

It will produce the results under *./output* based on lenet_tf.nb with inputs from *dataset.txt*.

5. Reference

For detailed information about NETINT AI Toolkit, please refer to the complete user guide.

A set of examples of using NETINT AI Toolkit can be found at:

```
~/NETINT_toolkit/examples
```

The source code of aiperf can be found at:

```
~/NETINT_toolkit/aiperf
```