Arm China Zhouyi Compass

Version: 3.3

Getting Started Guide

Confidential

arm CHINA

Arm China Zhouyi Compass

Getting Started Guide

Release Information

Document History

Issue	Date	Confidentiality	Change
0200-00	18 June 2021	Confidential	First release for 2.0
0201-00	30 September 2021	Confidential	First release for 2.1
0202-00	31 December 2021	Confidential	First release for 2.2
0203-00	31 March 2022	Confidential	First release for 2.3
0204-00	30 June 2022	Confidential	First release for 2.4
0205-00	30 September 2022	Confidential	First release for 2.5
0300-00	31 December 2022	Confidential	First release for 3.0
0301-00	31 March 2023	Confidential	First release for 3.1
0302-00	30 June 2023	Confidential	First release for 3.2
0303-00	30 September 2023	Confidential	First release for 3.3

Confidential Proprietary Notice

create or refer to any partnership relationship with any other company. Arm China may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with Arm China, then the click through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

Confidentiality Status

This document is Confidential. This document may only be used and distributed in accordance with the terms of the agreement entered into by Arm China and the party that Arm China delivered this document to.

Product Status

The information in this document is Final, that is for a developed product.

Web Address

https://www.armchina.com

Contents
Arm China Zhouyi Compass Getting Started Guide

Preface

This preface introduces the *Arm China Zhouyi Compass Getting Started Guide.*

It contains the following sections:

About this book

This book is intended to provide a simple guide for getting started quickly with the Arm China Zhouyi Compass software, to run a neural network model on the Zhouyi *Neural Processing Unit* (NPU) (on real hardware or on the simulator).

This book introduces the simple usage of the Zhouyi Compass NN compiler, simulator and driver, along with an example of running a neural network step by step, with which you can have a quick evaluation.

The book is meant to complement rather than replace the *Arm China Zhouyi Compass Software Technical Overview*. It does not include the detailed information about the NPU toolchain and NPU runtime.

Intended audience

This guide is for developers and engineers who want to evaluate the Zhouyi NPU processor.

Glossary

The Arm® Glossary is a list of terms used in Arm China documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm China meaning differs from the generally accepted meaning.

See the *Arm® Glossary* for more information.

Additional reading

This book contains information that is specific to this product. See the following documents for other relevant information.

Arm China publications

The following confidential document is only available to licensees:

- *Arm China Zhouyi Compass Software Technical Overview*.
- *Arm China Zhouyi Compass NN-Compiler User Guide*.
- *Arm China Zhouyi Compass Driver Runtime User Guide*.

Feedback

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content

If you have comments on content, send an e-mail to *errata@armchina.com*. Give:

- The title *Arm China Zhouyi Compass Getting Started Guide*.
- The number 61010012_0303_00_en.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

Arm China also welcomes general suggestions for additions and improvements.

Note

Arm China tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

Chapter 1
Getting started with Zhouyi Compass

This chapter provides a quick start guide for using Zhouyi Compass software to run a neural network model on the Zhouyi NPU.

It contains the following sections:

1.1 Introduction

This section describes how to use the Zhouyi Compass to run a neural network on the NPU (simulator). It provides a ResNet50 example to demonstrate how to process the ResNet50 model step by step and run it on the NPU.

This example is more about running a standard network, which means that no customized operations are considered. Therefore, only the NN compiler, simulator and Linux driver are used in this case, while the toolchain and customized operators are not involved. If you are interested in the toolchain and customized operation support, see the *Arm China Zhouyi Compass NN Compiler User Guide.*

In addition, the delivered software package includes an out-of-the-box use case (path: `Out-Of-Box/out-of-box-nn-compiler/customized_op_example/`) to show you the complete flow.

You can follow the instructions in this chapter to get started with your job quickly. The complete case (including code and scripts) is in the software package (path: `Out-Of-Box/out-of-box-nn-compiler /user-case-example/`).

There are six main steps in this example:

1. Install the Compass SDK.
2. Generate the model file.
3. Prepare the quantization dataset.
4. Build the executable graph.
5. Run the executable graph with the driver.
6. Check the output result with visual display.

You can also execute the following two scripts to get started quickly. These two scripts contain all the preceding steps.

```
$ cd  Out-Of-Box/out-of-box-nn-compiler
$ source env_setup.sh
$ cd  user-case-example/tf
$ ./out_of_box_example.sh
```

Note

Many default configurations are used in these scripts. You need to modify them when using them later.

1.2 Installing the Compass SDK

If you have already installed the Compass NPU NN compiler and Compass NPU simulator, you can skip this section.

You can confirm the installation in the current environment with the following commands:

```
$sh aipubuild –v

$sh aipu_simulator_x1 –v
```

For information about installing the NPU NN compiler, see *Chapter 4 Using the NN compiler* in the *Arm China Zhouyi Compass NN Compiler User Guide.*

For information about installing the NPU simulator, see *6.1 About the NPU simulator* in the *Arm China Zhouyi Compass Software Technical Overview.*

In the `Out-Of-Box/out-of-box-nn-compiler` directory, a demo dependency list is provided to install pip requirements:

```
$sh pip3 install –r lib_dependency.txt
```

In the `Out-Of-Box/out-of-box-nn-compiler` directory, quick install scripts are provided to install the Compass SDK:

```
$bash source env_setup.sh
```

Or

```
$tcsh source env_setup.csh
```

You need to modify the `lib_dependency.txt` and `env_setup.sh` as needed after performing the following actions before installation:

- Use a compatible Python version to install WHL.

  You can use `sudo` or add the `--user/--target <dir>/--prefix <dir>` option when you do not have the root permission. Note that when you are using the `--target` or `--prefix` option, PYTHONPATH needs to be set to the installation directory.
- Install some compatible third-party packages to enable the GPU, such as TensorFlow or PyTorch.

1.3 Generating the model file

For the pre-trained model file, you can download it from the official Model Zoo website.

For TensorFlow resnet_50, you can get the file from the official GitHub location:
https://github.com/tensorflow/models/tree/master/research/slim

or directly download it at:

http://download.tensorflow.org/models/resnet_v1_50_2016_08_28.tar.gz.

To generate the model file:

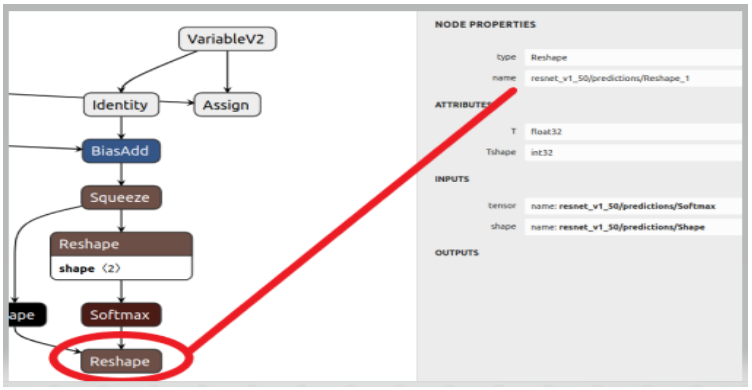
1. Unpack the downloaded file.
   You will get the resnet_50 checkpoint file `-- resnet_v1_50.ckpt`. This file contains the inference graph and pre-trained weights data.
2. Convert the checkpoint file to a freezed PB file. You can also download the pre-trained PB file from a third-party GitHub source.
3. Use the `git clone` command to clone TensorFlow models code with the URL **https://github.com/tensorflow/models**.
4. Export the inference graph of resnet_50. TensorFlow 1.x (1.13 and 1.15 recommended) and `tf_slim` package are needed.

```
$ python3 export_inference_graph.py \
        --alsologtostderr \
        --model_name=resnet_v1_50 \
        --image_size=224 \
        --labels_offset=1 \    # resnet_50 specific, default is 0
        --output_file=/tmp/resnet_v1_50_inf.pb
```

5. Freeze the exported graph with pre-trained weights.
   In this step, you will use a tool in TensorFlow code, so you need to use the `git clone` command to clone TensorFlow code with the URL **https://github.com/tensorflow/tensorflow**.
   `freeze_graph.py` is located at `tensorflow/tensorflow/python/tools/`, and TensorFlow 2.x is needed.

```
$ python3 freeze_graph.py \
            --input_graph=/tmp/resnet_v1_50_inf.pb \
            --input_checkpoint=/tmp/resnet_v1.ckpt \
            --input_binary=true --output_graph=/tmp/resnet_v1_50_frozen.pb \
            --output_node_names= resnet_v1_50/predictions/Reshape_1
```

   For the output node name of the graph, you can get it from the Graph Virtualization Tool (TensorBoard/Netron).



`resnet_v1_50_frozen.pb` is the model file that the NPU needs.

1.4 Preparing the quantization dataset

No matter whether you want to use the calibration function to get the accuracy of quantized models or use your customized data for model quantization, you need to prepare the dataset.

The dataset consists of two parts:

- The data file—Contains the pre-processed image or speech data (which is the input data that will be directly feed to the input node). Pre-processing varies from model to model, usually including resizing, and normalization. You need to ensure that the functions for the dataset pre-processing are exactly the same as the functions used during the model training.
- The label file—Contains the ground-truth, such as the category ID and location coordinates. Each ground truth should be stored in the order corresponding to the image/speech data in the data file.

These two files should be saved in the NumPy files eventually.

In the example, to generate the pre-processed dataset, the following pre-processing functions (as the ones during the model training) are applied to the dataset data:

- clip
- aspect_preserving_resize
- central_crop
- resize

In the `Out-Of-Box/out-of-box-nn-compiler` directory, you can read the script `./user-case-example/tf/preprocess_resnet_50_dataset/preprocess_for_resnet50_dataset.py` for details. After running the script, you can get the pre-processed data and label.

1.5 Building the executable graph and running it on the simulator

With the generated model file and pre-processed dataset, you can edit the NN compiler configuration file and then generate the NPU executable file for the example model.

The following shows the build configuration file for the example model resnet_50:

```
[Common]
mode=run

[Parser]
model_name = resnet_50
detection_postprocess =
model_domain = image_classification
output = resnet_v1_50/predictions/Reshape
input_model = ./resnet_50_model/frozen.pb
input = Placeholder
input_shape = [1,224,224,3]

[Optimizer]
dataset = NumpyDataset
calibration_data = ./preprocess_resnet_50_dataset/dataset.npy
metric=TopKMetric
data = ./preprocess_resnet_50_dataset/dataset.npy
label = ./preprocess_resnet_50_dataset/label.npy

[GBuilder]
simulator=./aipu_simulator
target=Z2_1104
inputs=./resnet_50/input.bin
outputs=./resnet_50/output_resnet_50.bin
profile= True
```

Note the following parameters:

- `input`: Define the input node of the model, excluding the pre-processing nodes. You need to check with the model structure.
- `output`: Define the output node of the model. You need to check with the model structure.
- `input_shape`: The shape of the input node.
- `dataset`: The plugin name of the quantization dataset to load the calibration file, using built-in dataset plugin `NumpyDataset`.
- `Metric` (optional): It calculates the result using the data/label file and built-in metric `TopKMetric`.
- `calibration_data`/`data`/`label`: This example uses the same dataset to quantize the model and metricize the accuracy.

The NN compiler (optimizer) automatically uses CUDA to accelerate calculation when it is available. To disable this feature, set the environment variable CUDA_VISIBLE_DEVICES to ‘-1’ or ‘’.

For the settings of other parameters, see Chapter 4 of the *Arm China Zhouyi Compass NN Compiler User Guide*.

With the generated NPU executable file, you can run it on the simulator with the NPU driver.

1.6 Running the executable graph with the driver

This section describes how to run a neural network model based on the NPU driver and Linux OS. To make it easier to understand the NPU driver usage, this section only focuses on running a neural network model on the simulator.

First, you will find the Linux driver in `AI610-SDK-1012-xxxx-xxx/Linux-driver/` of the SDK release package.

The following table lists the files in the `linux` directory.

Table 1-1 Files in the linux directory

File name	Description
`driver`	`Umd`: NPU User Library source code. `kmd`: NPU Linux driver source code. This file is not needed in this case because the graph does not run on hardware. For more information, see *Arm China Zhouyi Compass Driver and Runtime User Guide*.
`devicetree`	The reference of the device tree. Refer to this file when you cross-compile KMD for a specific hardware platform.
`readme`	Readme file.

A specific example which uses UMD is integrated into the `/Out-Of-Box/out-of-box-linux` directory.

The following table lists the files in the `out-of-box` directory.

Table 1-2 Files in the out-of-box directory

File name	Description
`src`	User application source code. For more information, see *Arm China Zhouyi Compass Driver and Runtime User Guide*.
`benchmarks`	The neural network benchmark named `reset_50` which contains three binary files (you can build them first by running the case in `Out-Of-Box/out-of-box-nn-compiler/user-case-example/tf/`): • `aipu.bin`: Customized neural network model. • `input0.bin`: Input data needed by the model. • `output.bin`: Inference result must be compared with this reference file.
`build_umd.sh`	The script that compiles UMD and generates a runtime dynamic library.
`run.sh`	The script that invokes the simulator to infer a neural network benchmark.
`readme.txt`	Readme file.
`makefile`	Makefile.
`out-of-box-test.sh`	The entry to trigger all test procedures.

This example demonstrates how to run a neural network benchmark based on UMD and the simulator. Its calling flow is as follows:

1. Run `out-of-box-test.sh`. This script will trigger the following steps.
2. Call `build_umd.sh`. It will change the working directory to `AI610-SDK-1012-xxxx-xxx/Linux-driver/umd/` and call the internal `build.sh` to compile the whole UMD library source code. Finally, it will generate the runtime library named `libaipudrv.so`.

3. Call `makefile`. It will compile the user application together with `libaipudrv.so` in the current `src` directory and generate the test binary named `aipu_simulation_test` stored in the build directory.
4. Call `run.sh`. It will execute `aipu_simulator_test` and indirectly invoke the simulator to infer one neural network benchmark represented by `aipu.bin`, and then take `input.bin` as the input data. Finally, the simulator will generate the result data. This result data must be compared with `output.bin` to further check its inference correctness.

   If the case is running functionally, it will show the following prompt at the log tail.

   ```
   [TEST INFO] Test Result Check PASS!

   [TEST RUN INFO] memory section dump files saved under: ./output/resnet_50-pass-202X-XX-XX-XX-XX-XX
   ```

If you have performed all preceding steps, you can select another way to quickly run this demo application by using this command:

```
$sh run.sh -c resnet_50
```

This command acts exactly the same as step 4.

1.7 Checking the output result with visual display

In different hardware environments, the output binary may have minor errors with the `ref` binary. To help you check the result, this section takes the image classification result check with visual display for example.

You can find the category labels in `Out-Of-Box/out-of-box-nn-compiler/user-case-example/tf/imagenet_classes.py`. This file can be customized if another dataset is used.

After generating the output binary successfully, run the following script to check the result:

```
$python3 compare_class.py
```

This script produces a visual image of the output binary file, and the terminal will print out the respective category names, which can be compared to check whether the network has detected the category correctly.

Use a Shetland sheepdog image with the resnet_50 model for example, if the output is processed correctly, it will show the following log and a Shetland sheepdog picture in the terminal:

```
[TEST PASS]  Class Check PASSED! Class number is 230

You detect Shetland sheepdog, Shetland sheep dog, Shetland successfully.

Show input picture...
```

If the processed output mismatches with the label, it will show the failed log in the terminal:

```
[TEST FAIL]  Class Check FAILED! Class number for output is xx, Class number
for ref is xx.

You detect xx, but you should detect xx from ref label.
```