USER GUIDE

VIA Neuron Runtime Helper

1.00-06062024

Revision History

Version	Date	Remarks
1.00	06/06/2024	Initial release

Table of Contents

1. Overview

VIA Neuron Runtime Helper is a Python package to speed up building of new projects with machine learning capabilities, or integrating AI into existing projects. By providing APIs tightly integrated with MediaTek's AI Framework "NeuroPilot", VIA Neuron Runtime Helper simplifies the complex task of setting up hardware accelerated AI inferencing. This user guide will walk through the basic steps of compiling AI models for hardware acceleration, and executing AI inferencing using the VIA Neuron Runtime Helper's Python APIs.

2. AI Inferencing and Hardware Acceleration on the VIA AI Transforma Model 1

AI model inferencing on Edge devices is typically built upon executing mathematical operations such as arithmetic functions, grouping and sorting values, etc. These mathematical operations can be accelerated using specific hardware separated from the general-purpose processor. To allow hardware accelerated AI inferencing, tasks of function execution can be delegated through an AI framework, instead of building a pipeline.

The VIA AI Transforma Model 1 board supports TensorFlow Lite ("TFLite") as the underlaying framework for AI hardware acceleration. TFLite models can be executed directly on the VIA AI Transforma Model 1 board's C.P.U and G.P.U without special conversion. For instructions on running TFLite models (.tflite files) on the VIA AI Transforma Model 1 board, visit webpage https://www.tensorflow.org/lite/guide/inference#linux_platform.

To leverage dedicated hardware acceleration with the onboard N.P.U, a TFLite model needs to be converted to a statically compiled network (DLA file) before being executed by the NPU. "ncc-tflite" is a compiler tool provided by MediaTek to generate such a network.

Before converting a TFLite model to a DLA file, determine whether the source TFLite model has a 32-bit floating point data type, which generates results that are more accurate but are slower to execute, or, has been optimized and quantized to an 8-bit integer data type, which generates results that execute faster but sacrifices accuracy.

Note:
For more information on Model Optimization and Quantization, visit Google's TFLite webpage https://www.tensorflow.org/lite/performance/model_optimization.

Processors on the VIA AI Transforma Model 1 board can support the execution of TFLite models of various data types. The following table specifies the supported TFLite model data types:

Processor	TFLite Model Data Types							
	8-bit Asymmetric Unsigned Integer	8-bit Asymmetric Integer	8-bit Symmetric Integer	16-bit Symmetric Integer	16-bit Floating Point	32-bit Floating Point	8-bit Boolean	32-bit Integer
NPU (MDLA 3.0)	✓	✓	✓	✓	✓	✗	✗	✗
G.P.U	✗	✗	✗	✗	✓	✓	✗	✗
C.P.U	✓	✓	✗	✗	✓	✓	✓	✓

Note:
For more information on Google's TFLite 8-bit quantization specification, visit Google's TFLite webpage https://www.tensorflow.org/lite/performance/quantization_spec.

2.1 Converting a 32-bit Floating Point TFLite Model to a DLA File

Open a command-line interface to the VIA AI Transforma Model 1 board, and run the following command to convert the 32-bit floating point TFLite model (example: "yolov8s_f32_640.tflite") to a DLA file (example: "yolov8s_f32_640.dla") :

```
> cd /usr/share/ai_transforma/NeuronRuntimeHelper/
> ncc-tflite --arch=mdla3.0 --opt=3 --relax-fp32 yolov8s_f32_640.tflite -o yolov8s_f32_640.dla
```

2.2 Converting a Quantized 8-bit Asymmetric Unsigned Integer TFLite Model to a DLA File

Before converting a quantized TFLite model to DLA file, one of the following online or offline method can be used to generate a JSON configuration file describing the model's quantization parameters. Follow the instructions described below for the desired method, and for conversion of the quantized TFLite model to DLA file:

2.2.1 Online Method

Generation of the JSON Configuration File

Step 1

Open the Netron webpage https://netron.app/.

Note:
Netron is an open-sourced online neural network visualizer.

Step 2

Upload and view your TFLite model (example: "yolov8s_u8_640.tflite"), and identify the layer(s) in the TFLite model that have been quantized.

Step 3

Create a JSON configuration file (example: "yolov8s_u8_640.json") at location "/usr/share/ai_transforma/NeuronRuntimeHelper/"

Step 4

Add the following information into the JSON configuration file, replacing the values with the actual quantization parameters shown in Netron:

- Scale (a floating-point number representing the quantization scale)
- Zero point (an integer representing the quantization zero-point)

```
{
        "scale" : "0.004030249",
        "zero-point" : "-4.0"
}
```

Conversion of the Quantized TFLite Model to a DLA File

Open a command-line interface to the VIA AI Transforma Model 1 board, and run the following command to convert the quantized TFLite model (example: "yolov8s_u8_640.tflite") to a DLA file (example: "yolov8s_u8_640.dla"):

```
> cd /usr/share/ai_transforma/NeuronRuntimeHelper/
> ncc-tflite --arch=mdla3.0 --opt=3 --opt-aggressive --dla-metadata dequantize:yolov8s_
u8_640.json yolov8s_u8_640.tflite -o yolov8s_u8_640.dla
```

2.2.2 Offline Method

Generation of the JSON Configuration File

Step 1

Create a Python file "export_tflite_quant.py" at location "/usr/share/ai_transforma/NeuronRuntimeHelper/"

Step 2

Copy-paste the following content into the Python file, replacing the highlighted parts with the names of your JSON configuration file and quantized TFLite model:

```
import tensorflow as tf
import json

def extract_output_details(model_path):
    interpreter = tf.lite.Interpreter(model_path=model_path)
    interpreter.allocate_tensors()

    output_details = interpreter.get_output_details()
    output_info = {}

    for output_detail in output_details:
        output_info = {
            "scale": str(output_detail["quantization_parameters"]["scales"][0]),
            "zero_point": str(output_detail["quantization_parameters"]["zero_points"][0])
        }

    return output_info

def save_to_json(output_info, filename="yolov8s_u8_640.json"):
    """Saves the output tensor details to a JSON file."""
    with open(filename, "w") as json_file:
        json.dump(output_info, json_file, indent=4)

# Example
model_path = "yolov8_u8_640.tflite"
output_details = extract_output_details(model_path)
save_to_json(output_details)
```

Step 3

Save the Python file, and run the following command to generate the JSON configuration file:

```
> cd /usr/share/ai_transforma/NeuronRuntimeHelper/
> python3 export_tflite_quant.py
```

Conversion of the Quantized TFLite Model to a DLA File

Run the following command to convert the quantized TFLite model (example: "yolov8s_u8_640.tflite") to a DLA file (example: "yolov8s_u8_640.dla"):

```
> cd /usr/share/ai_transforma/NeuronRuntimeHelper/
> ncc-tflite --arch=mdla3.0 --opt=3 --opt-aggressive --dla-metadata dequantize:yolov8s_
u8_640.json yolov8s_u8_640.tflite -o yolov8s_u8_640.dla
```

Notes:
1) For detailed instructions and parameters to use the Neuron Compiler (ncc-tflite), refer to MediaTek's official documentation on webpage https://mediatek.gitlab.io/aiot/doc/aiot-dev-guide/master/sw/yocto/ml-guide/neuron-sdk/neuron_tools.html#ml-neuron-compiler.

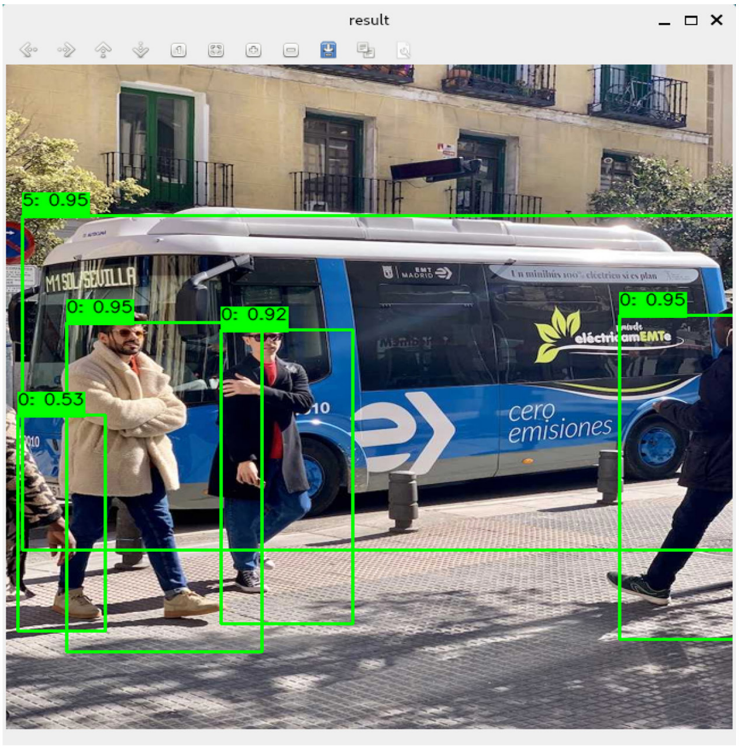
2) To learn more about the YOLOv8 model, visit webpage https://github.com/ultralytics/ultralytics.

2.3 Running the Neuron Runtime Helper Sample Code

Open a command-line interface to the VIA AI Transforma Model 1 board, and run the Neuron Runtime Helper sample code using the following command:

```
> cd /usr/share/ai_transforma/NeuronRuntimeHelper/
> python3 ai_transforma_example.py
```

The output will be displayed on screen as shown below:



3. Neuron Runtime Helper API Reference

This section provides descriptions of the Neuron Runtime Helper APIs:

API	Description
Initialize	Initializes MediaTek's NeuronRuntimeV2 instance
Execute	Runs inference on the loaded model
GetInputDimensions	Gets the dimensions of all input tensors
GetOutputDimensions	Gets the dimensions of all output tensors

API	Description
SetInputBuffer	Sets model input data
GetOutputBuffer	Gets model output data

Initialize

Function	Initialize() -> bool	
Description	Initializes MediaTek's NeuronRuntimeV2 instance	
Return	(bool)	True: If initialized successfully
		False: If failed to initialize

Execute

Function	Execute() -> bool	
Description	Runs inference on the loaded model	
Return	(bool)	True: If successfully run
		False: If failed to run

GetInputDimensions

Function	GetInputDimensions () -> list[list[int]]
Description	Gets the dimensions of all input tensors

GetOutputDimensions

Function	GetOutputDimensions () -> list[list[int]]
Description	Gets the dimensions of all output tensors

SetInputBuffer

Function	SetInputBuffer (input_buffer: numpy.ndarray, index: int)
Description	Sets model input data
Parameters	• inputBuffer: A NumPy array (uint8, float32) • index: model input tensor index

GetOutputBuffer

Function	GetOutputBuffer (index: int) -> numpy.ndarray
Description	Gets model output data
Parameters	index: model output tensor index
Return	A floating-point numpy array

Note:
For more information on AI inferencing, refer to Google's TensorFlow Lite inference document on webpage https://www.tensorflow.org/lite/guide/inference.

Taiwan Headquarters

1F, 531 Zhong-zheng Road,
Xindian Dist., New Taipei City 231
Taiwan

Tel: 886-2-2218-5452
Fax: 886-2-2218-9860
Email: embedded@via.com.tw

USA

940 Mission Court
Fremont, CA 94539,
USA

Tel: 1-510-687-4688
Fax: 1-510-687-4654
Email: embedded@viatech.com

Japan

3-15-7 Ebisu MT Bldg. 6F,
Higashi, Shibuya-ku
Tokyo 150-0011
Japan

Tel: 81-3-5466-1637
Fax: 81-3-5466-1638
Email: embedded@viatech.co.jp

China

Tsinghua Science Park Bldg. 7
No. 1 Zongguancun East Road,
Haidian Dist., Beijing, 100084
China

Tel: 86-10-59852288
Fax: 86-10-59852299
Email: embedded@viatech.com.cn

Europe

Email: embedded@via-tech.eu