Canonical

Running AI Application on RZ/V2L with Ubuntu images

Prepared by: Hon Ming Hui

Reviewed by:

Prepared on: Mar 3, 2026

Version: 1.1

1. Prerequisites

To set up the RZ/V2L with the Ubuntu image, please refer to the Quick Start Guide available at https://ubuntu.com/download/renesas-iot.

2. Basics of AI applications on RZ/V2L

2.1. Major components

The AI application comprises three principal components:

1. The compiled model tailored for the RZ/V2L platform.
2. The runtime library responsible for interaction with the DRP-AI driver.
3. The user application.

2.2. Compiled model

Renesas provides the Robust Unified Heterogeneous Model Integration (RUHMI) framework for the optimization and deployment of AI models. The project repository is available on GitHub: GitHub - renesas-rz/rzv_drp-ai_tvm.

Model compilation is typically executed on a powerful host machine rather than the target device (RZ/V2L). Renesas offers the necessary setup, utilizing Ubuntu 22.04 as the base operating system. Further details on the setup can be found at https://github.com/renesas-rz/rzv_drp-ai_tvm/tree/main/setup (important note: an x86 architecture machine is required).

2.3. TVM Run time library

The TVM runtime, which communicates with the DRP AI driver, is provided as a pre-built library, available at https://github.com/renesas-rz/rzv_drp-ai_tvm/tree/main/obj/build_runtime.

As the TVM Runtime library evolves in tandem with the Model compiler, it is advisable for the final application to consistently utilize a specific, unified version of the TVM components.

2.4. User application

This application integrates all components to form the final user application. Within the Ubuntu environment, the recommended method for creating such an application is through snap.

There is a tutorial application that come with the DRP AI repository https://github.com/renesas-rz/rzv_drp-ai_tvm/tree/main/apps. The corresponding example as snap is available at https://github.com/canonical/rzv_drp-ai_tvm_snap. The primary configuration file for a snap is the snapcraft.yaml. The following provides an examination of key code snippets from this example.

1. The `tvm-runtime` component is responsible for retrieving the runtime libraries. It will also stage all the requisite header files necessary for application compilation.

```
None
parts:
  tvm-runtime:
    plugin: nil
    source: https://github.com/renesas-rz/rzv_drp-ai_tvm.git
    source-type: git

# Refer to the full yaml file for the detail steps
```

2. The subsequent section addresses the compilation of the application utilizing CMake. The Snapcraft CMake plugin will execute this task, provided the appropriate CMAKE parameters are configured as detailed below.

```
None
 tutorial-app:
    after: [tvm-runtime]
    plugin: cmake
    source: ${CRAFT_PROJECT_DIR}/../parts/tvm-runtime/src/apps
    source-type: local
    cmake-parameters:
      - -DCMAKE_INSTALL_PREFIX=/usr/
      - -DCMAKE_C_COMPILER=aarch64-linux-gnu-gcc
      - -DCMAKE_CXX_COMPILER=aarch64-linux-gnu-g++
    build-environment:
      - PRODUCT: V2L
```

```
      - TVM_ROOT: ${CRAFT_PROJECT_DIR}/../parts/tvm-runtime/src
```

3. The `stage-packages` section within the `tutorial-app` component is employed to incorporate all necessary dependencies for compiling the AI applications and subsequently installing them within the resulting snap. Utilizing existing packages from the Ubuntu archive within a snap environment is a relatively uncomplicated process.

```
None
    stage-packages:
      - libmmngr1:$CRAFT_ARCH_BUILD_FOR
      - libmmngrbuf1:$CRAFT_ARCH_BUILD_FOR
```

4. And then the final step is to expose the application to the user

```
None
apps:
  tutorial-app:
    command: usr/bin/tutorial_app_v2ml
    environment:
      LD_LIBRARY_PATH:
$LD_LIBRARY_PATH:$SNAP/usr/lib:$SNAP/usr/lib/$CRAFT_ARCH_TRIPLET_BUILD_FOR/la
pack:$SNAP/usr/lib/$CRAFT_ARCH_TRIPLET_BUILD_FOR/blas
    plugs: [camera, opengl, wayland]
```

5. This example didn't include the compiled model within the snap so that the user can develop and iterate with different models. But for deployment scenario, it would be better to also stage the compiled model within the snap

3. How-to

3.1. Setup your environment

Instructions for setting up snapcraft are available at the following link: https://documentation.ubuntu.com/snapcraft/stable/tutorials/craft-a-snap/.

3.2. Creating the snap

```Shell
git clone https://github.com/canonical/rzv_drp-ai_tvm_snap
cd rzv_drp-ai_tvm_snap
snapcraft pack
```

After that you will get a rzv-drp-ai-tvm-examples_1.0_arm64.snap file and you can deploy it onto the device

3.3. Install and run the snap

```Shell
sudo snap install --devmode rzv-drp-ai-tvm-examples_1.0_arm64.snap
# Extract your models and parameters to a working directory
sudo rzv-drp-ai-tvm-examples.tutorial-app
```

3.4. Publishing your snap

To facilitate the publication or deployment of your application, you may upload your snap to the snap store, enabling users to locate and install it with a single command. The snapcraft documentation provides further information on this process.