Qualcomm TensorFlow Lite SDK Tools Quick Start Guide

80-50450-52 Rev. AB

October 10, 2023

Revision history

Revision	Date	Description
AA	September 2023	Initial release
AB	October 2023	■ In Generate platform SDK, updated the commands to build the user space images and platform SDK ■ Added Generate TFLite SDK with Linux workstation ■ Added Work with QNN external TFLite Delegate ■ In Benchmark, updated the scripts for External Delegate

Contents

Figures

Tables

1 Introduction to Qualcomm TFLite SDK tools

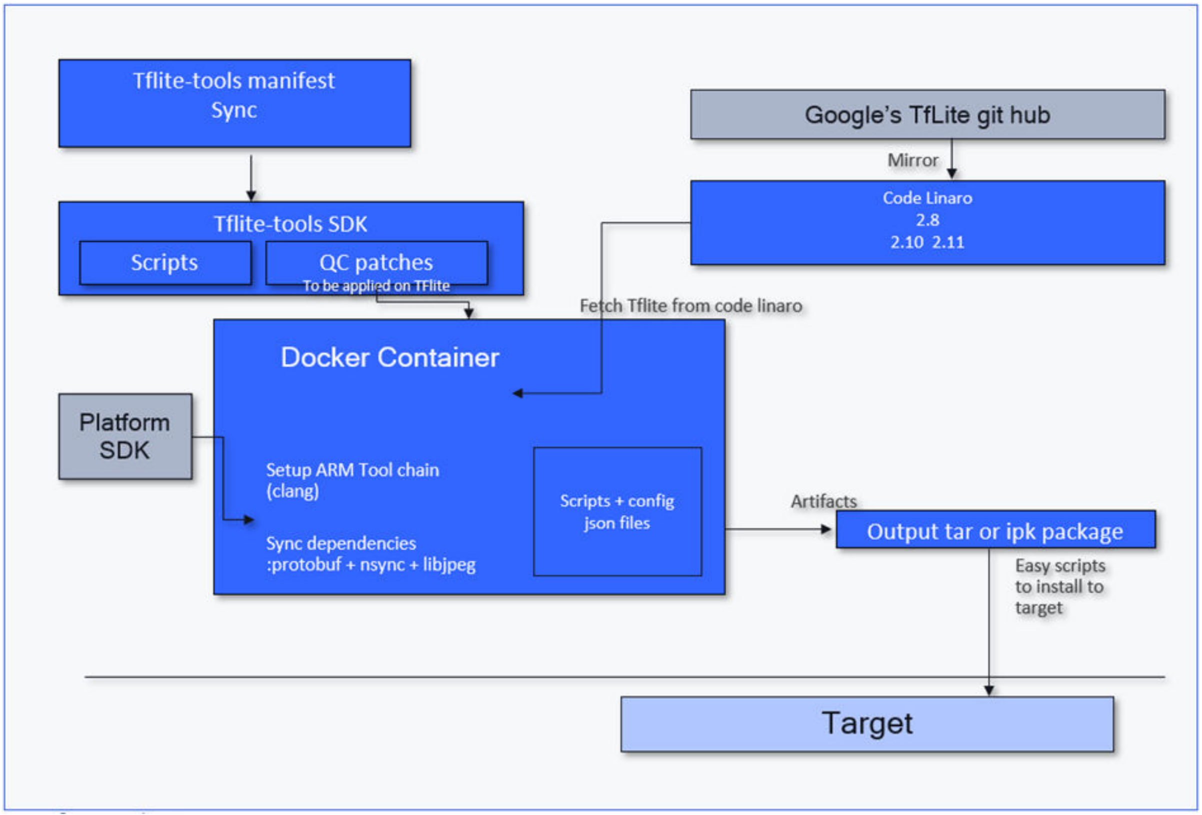
The Qualcomm TensorFlow Lite software development kit (Qualcomm TFLite SDK) tools provide the TensorFlow Lite framework for on-device artificial intelligence (AI) inferencing, which facilitates application developers to develop or run suitable AI applications.

This document provides step-by-step instructions to compile a standalone Qualcomm TFLite SDK and set up the development environment. This enables the developer workflow, which includes:

- setting up the build environment where the developer can compile the Qualcomm TFLite SDK
- developing standalone Qualcomm TFLite SDK applications

For support, seehttps://www.qualcomm.com/support.

The following figure provides a summary of the Qualcomm TFLite SDK workflow:



Tflite-tools manifest
Sync
Google's TfLite git hub
Mirror
Code Linaro
2.8
2.10 2.11
Tflite-tools SDK
Scripts
QC patches
To be applied on TFlite
Fetch Tflite from code linaro
Docker Container
Platform SDK
Setup ARM Tool chain (clang)
Sync dependencies :protobuf + nsync + libjpeg
Scripts + config json files
Artifacts
Output tar or ipk package
Easy scripts to install to target
Target

Figure 1-1 Qualcomm TFLite SDK workflow

The tool requires a platform SDK and a configuration file (JSON format) to generate the Qualcomm TFLite SDK artifacts.

To build an end-to-end application using multimedia, AI, and computer vision (CV) subsystems, see *Qualcomm Intelligent Multimedia SDK (QIM SDK) Quick Start Guide* (80-50450-51).

The table shows Qualcomm TFLite SDK version mapping with CodeLinaro release tag:

Table 1-1 Release information

Qualcomm TFLite SDK version	CodeLinaro release tag
V1.0	Qualcomm TFLITE.SDK.1.0.r1-00200-TFLITE.0
	▪ TFLITE.SDK.1.0.r1-00500-TFLITE.0.xml

Table 1-2 Supported Qualcomm TFLite SDK versions

Qualcomm TFLite SDK version	Supported software product	Supported TFLite version
V1.0	QCS8550.LE.1.0	▪ 2.6.0 ▪ 2.8.0 ▪ 2.10.1 ▪ 2.11.1 ▪ 2.12.1 ▪ 2.13.0

References

Table 1-3 Related documents

Title	Number
Qualcomm	
00067.1 Release Note for QCS8550.LE.1.0	RNO-230830225415
Qualcomm Intelligent Multimedia SDK (QIM SDK) Quick Start Guide	80-50450-51
Qualcomm Intelligent Multimedia SDK (QIM SDK) Reference	80-50450-50
Resources	
https://source.android.com/docs/setup/start/initializing	–

Table 1-4 Acronyms and definitions

Acronym or term	Definition
AI	Artificial intelligence
BIOS	Basic input/output system
CV	Computer vision
IPK	Itsy package file
QIM SDK	Qualcomm Intelligent multimedia software development kit
SDK	Software development kit
TFLite	TensorFlow Lite
XNN	X_{th} nearest neighbor

2 Set up build environment for Qualcomm TFLite SDK tools

The Qualcomm TFLite SDK tools are released in source form; therefore, establishing the build environment to compile it is a mandatory but one-time setup.

Prerequisites

- Ensure that you have `sudo`access to the Linux host machine.
- Ensure that the Linux host version is Ubuntu 18.04 or Ubuntu 20.04.
- Increase the maximum user watches and maximum user instances on the host system.
- Add the following command lines to`/etc/sysctl.conf`and reboot the host:

```
fs.inotify.max_user_instances=8192
fs.inotify.max_user_watches=542288
```

2.1 Install required host packages

The host packages are installed on the Linux host machine.

Run the commands to install the host packages:

```
$ sudo apt install -y jq
$ sudo apt install -y texinfo chrpath libxml-simple-perl openjdk-8-jdk-
headless
```

For Ubuntu 18.04 and higher:

```
$ sudo apt-get install git-core gnupg flex bison build-essential zip curl
zlib1g-dev gcc-multilib g++-multilib libc6-dev-i386 libncurses5
lib32ncurses5- dev x11proto-core-dev libx11-dev lib32z1-dev libgl1-mesa-dev
libxml2-utils xsltproc unzip fontconfig
```

For more information, see https://source.android.com/docs/setup/start/initializing.

2.2 Set up docker environment

A docker is a platform used to build, develop, test, and deliver software. To compile the SDK, the docker must be configured on the Linux host machine.

Ensure that CPU virtualization is enabled on the Linux host machine. If it is not enabled, do the following to enable it from the basic input/output system (BIOS) configuration settings:

1. Enable virtualization from BIOS:
   a. Press **F1** or **F2** when the system is booting up to step into BIOS.

      The BIOS window is displayed.

   b. Switch to the **Advanced** tab.
   c. In the **CPU Configuration** section, set **Virtualization Technology** to **Enabled**.
   a. Press **F12** to save and exit, and then restart the system.

   If these steps do not work, follow the specific instructions from the system provider to enable the virtualization.

2. Remove any old instances of the docker:

```
$ sudo apt remove docker-desktop
$ rm -r $HOME/.docker/desktop
$ sudo rm /usr/local/bin/com.docker.cli
$ sudo apt purge docker-desktop
```

3. Set up the docker remote repository:

```
$ sudo apt-get update
$ sudo apt-get install ca-certificates curl gnupg lsb-release
$ sudo mkdir -p /etc/apt/keyrings
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --
dearmor -o /etc/apt/keyrings/docker.gpg
$ echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/
keyrings/ docker.gpg] https://download.docker.com/linux/ubuntu $
(lsb_release -cs) stable" |    sudo tee /etc/apt/sources.list.d/
docker.list > /dev/null
```

4. Install docker engine:

```
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli
```

5. Add user to docker group:

```
$ sudo groupadd docker
$ sudo usermod -aG docker $USER
```

6. Reboot the system.

3 Generate platform SDK

The platform SDK is a mandatory requirement to compile the Qualcomm TFLite SDK tools. It provides all the required platform dependencies required by the Qualcomm TFLite SDK.

Do the following to generate the platform SDK:

1. Create a build for the preferred software product.

   The instructions to build the QCS8550.LE.1.0release are provided in the release notes. To access the release notes, see References.

   If the images were previously built, execute step 2, and then create a clean build.

2. Run the following command to build the user space images and platform SDK:

   For QCS8550.LE.1.0, add the machine feature `qti-tflite-delegate` in `MACHINE_FEATURES` in the `kalama.conf` file and source the build environment according to instructions from the release notes.

   After generating user space images from build, run the following command to generate the platform SDK.

   ```
   $ bitbake -fc populate_sdk qti-robotics-image
   ```

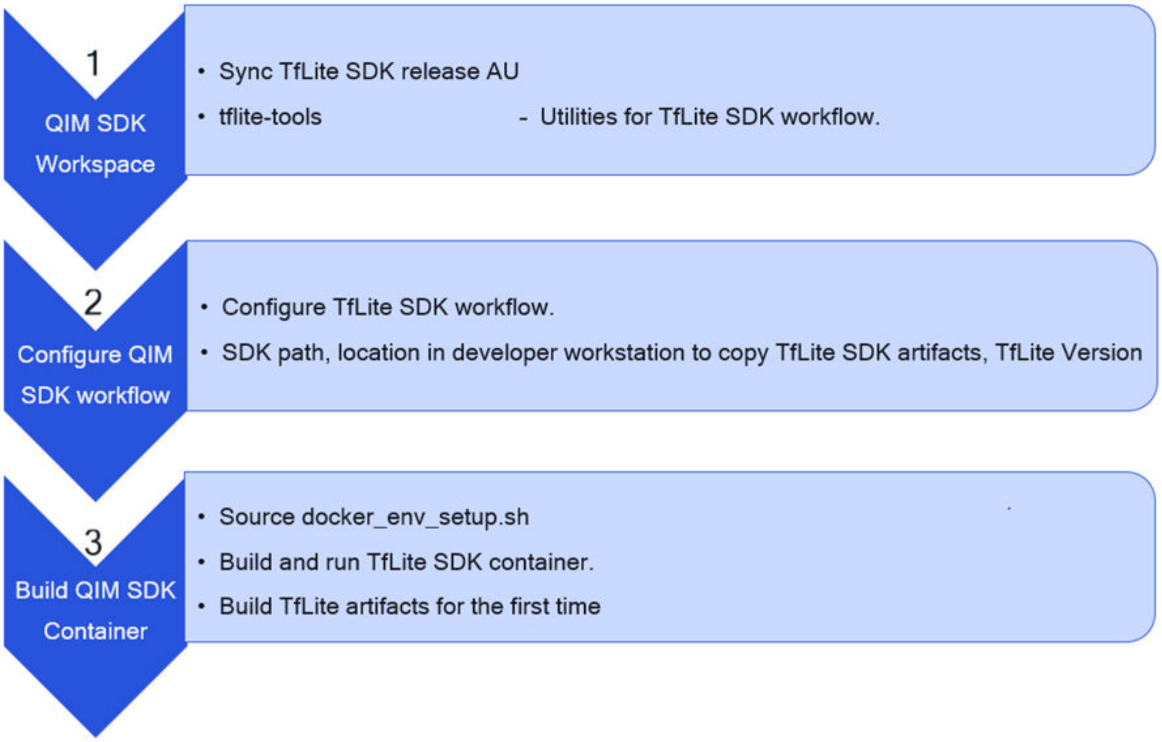
4 Build Qualcomm TFLite SDK tools – developer workflow

The Qualcomm TFLite SDK tools workflow requires the developer to provide the configuration file with valid input entries. The helper shell scripts from the tflite-tools project (present in the Qualcomm TFLite SDK source tree) provide helper utility functions to set up the shell environment, which can be used for the Qualcomm TFLite SDK workflow.

The developer builds the Qualcomm TFLite SDK projects within the container and generates the artifacts using the utilities provided by tflite-tools.

After a Qualcomm TFLite SDK container is built, the developer can attach to the container and use the helper utilities in the container shell environment for continuous development.

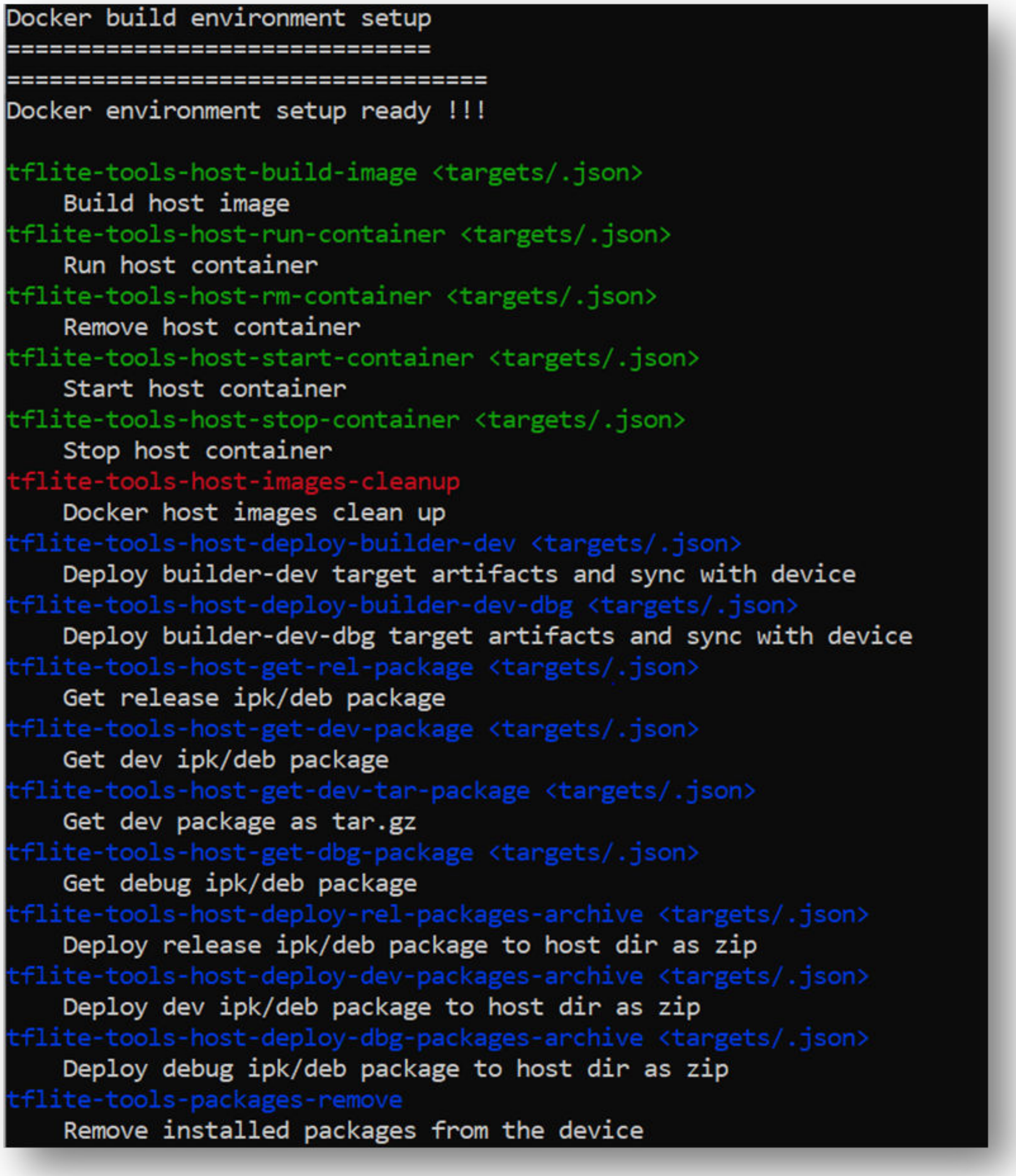
- There is a provision to install the Qualcomm TFLite SDK artifacts to a Qualcomm device connected to the Linux host via USB/adb.
- There is also a provision to copy the Qualcomm TFLite SDK artifacts from the container to a different host machine where the Qualcomm device is connected.



1
QIM SDK Workspace
• Sync TfLite SDK release AU
• tflite-tools - Utilities for TfLite SDK workflow.
2
Configure QIM SDK workflow
• Configure TfLite SDK workflow.
• SDK path, location in developer workstation to copy TfLite SDK artifacts, TfLite Version
3
Build QIM SDK Container
• Source docker_env_setup.sh
• Build and run TfLite SDK container.
• Build TfLite artifacts for the first time

Figure 4-1 Qualcomm TFLite SDK container workflow

The following figure lists the set of utilities available after setting up the container build environment using the helper scripts for building the Qualcomm TFLite SDK.



```
Docker build environment setup
==============================
==================================
Docker environment setup ready !!!

tflite-tools-host-build-image <targets/.json>
    Build host image
tflite-tools-host-run-container <targets/.json>
    Run host container
tflite-tools-host-rm-container <targets/.json>
    Remove host container
tflite-tools-host-start-container <targets/.json>
    Start host container
tflite-tools-host-stop-container <targets/.json>
    Stop host container
tflite-tools-host-images-cleanup
    Docker host images clean up
tflite-tools-host-deploy-builder-dev <targets/.json>
    Deploy builder-dev target artifacts and sync with device
tflite-tools-host-deploy-builder-dev-dbg <targets/.json>
    Deploy builder-dev-dbg target artifacts and sync with device
tflite-tools-host-get-rel-package <targets/.json>
    Get release ipk/deb package
tflite-tools-host-get-dev-package <targets/.json>
    Get dev ipk/deb package
tflite-tools-host-get-dev-tar-package <targets/.json>
    Get dev package as tar.gz
tflite-tools-host-get-dbg-package <targets/.json>
    Get debug ipk/deb package
tflite-tools-host-deploy-rel-packages-archive <targets/.json>
    Deploy release ipk/deb package to host dir as zip
tflite-tools-host-deploy-dev-packages-archive <targets/.json>
    Deploy dev ipk/deb package to host dir as zip
tflite-tools-host-deploy-dbg-packages-archive <targets/.json>
    Deploy debug ipk/deb package to host dir as zip
tflite-tools-packages-remove
    Remove installed packages from the device
```

Figure 4-2 Available utilities after sourcing the container build environment

The figure shows the sequence of execution of the utilities:

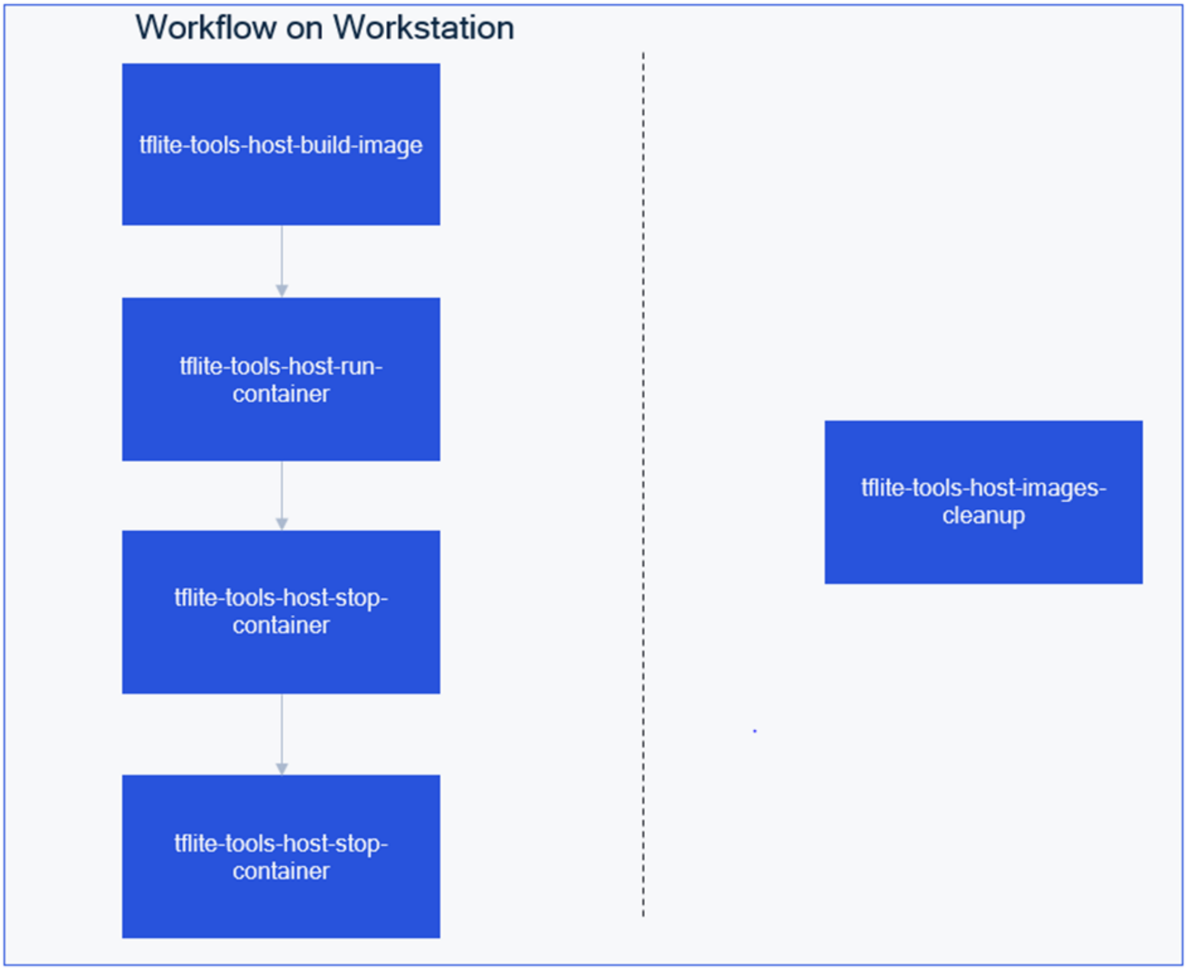

Workflow on Workstation
tflite-tools-host-build-image
tflite-tools-host-run-container
tflite-tools-host-stop-container
tflite-tools-host-stop-container
tflite-tools-host-images-cleanup

Figure 4-3 Sequence of utilities on host

4.1 Sync and build Qualcomm TFLite SDK

The Qualcomm TFLite SDK is compiled when the docker image is created.

To sync and build the Qualcomm TFLite SDK, do the following:

1. Create a directory on the host file system to sync the Qualcomm TFLite SDK workspace. For example:

   ```
   $mkdir <tflite-sdk-workspace>
   $cd <tflite-sdk-workspace>
   ```

2. Fetch the Qualcomm TFLite SDK source code from CodeLinaro:

   ```
   $ repo init -u https://git.codelinaro.org/clo/le/sdktflite/tflite/
   manifest.git --repo-branch=qc/stable --repo-url=git://git.quicinc.com/
   tools/repo.git -m TFLITE.SDK.1.0.r1-00200-TFLITE.0.xml -b release &&
   repo sync -qc --no-tags -j8
   ```

3. Create a directory on the host file system that can be mounted into docker.

   For example: `mkdir-p <tflite-sdk-workspace>/<host_dir>`

   This directory can be created anywhere on the Linux host machine, and it does not depend on where the Qualcomm TFLite SDK project is synced.

   After the workflow is completed within the container, the Qualcomm TFLite SDK artifacts can be found at the directory created in this step.

4. Edit the JSON configuration file present in `<tflite-sdk-workspace>/tflite-tools/targets/le-tflite-tools-builder.json` with the following entries:

```
{
    "Image": "tflite-tools-builder",
    "Device_OS": "le",
    "Additional_tag": "",
    "TFLite_Version": "2.11.1",
    "Delegates": {
        "Hexagon_delegate": "OFF",
        "Gpu_delegate": "ON",
        "Xnnpack_delegate": "ON"
    },
    "TFLite_rsync_destination": "<tflite-sdk-workspace>/<host_dir>",
    "SDK_path": "<path-to-workspace>/build-qti-distro-fullstack-perf/tmp-
glibc/deploy/sdk>",
    "SDK_shell_file": "<sdk-shell-filename not containing *-ext-*>",
    "Base_Dir_Location": "<base dir location - Optional>"
}
```

   For more information on the entries mentioned in the json configuration file, see the `Docker.md` readme file at `<tflite-sdk-workspace>/tflite-tools/`.

   NOTE For QCS8550, the Qualcomm® Hexagon™ DSP delegate is not supported.

5. Source the script to set up the environment:

```
$ cd <tflite-sdk-workspace>/tflite-tools
$ source ./scripts/host/docker_env_setup.sh
```

6. Build the Qualcomm TFLite SDK docker image:

```
$ tflite-tools-host-build-image ./targets/le-tflite-tools-builder.json
```

   If the build setup fails, see Troubleshoot docker setup.

   After successful completion, the following message is displayed:

```
"Status:Build image completed successfully!!"
```

   Running this step builds the Qualcomm TFLite SDK as well.

7. Run the Qualcomm TFLite SDK docker container. This starts the container with the tags provided in the JSON configuration file.

```
$tflite-tools-host-run-container ./targets/le-tflite-tools-builder.json
```

8. Attach to the container started from the previous step.

```
$ docker attach <tflite-tools-container>
```

The Qualcomm TFLite SDK is compiled, and the artifacts are ready to be deployed or further can be used to generate the QIM SDK TFLite plug-in.

4.2 Connect device to host and deploy artifacts

After compilation, there are two mechanisms to connect the device to a host and deploy the Qualcomm TFLite SDK artifacts.

- Device connected to a local Linux host:

  A developer connects the device to a workstation and installs the Qualcomm TFLite SDK artifacts from the container directly on the device (QCS8550).

- Device connected to a remote host:

  A developer connects the device to a remote workstation, and they can use the pack manager installer commands on Windows and Linux platforms to install the Qualcomm TFLite SDK artifacts to the device (QCS8550).

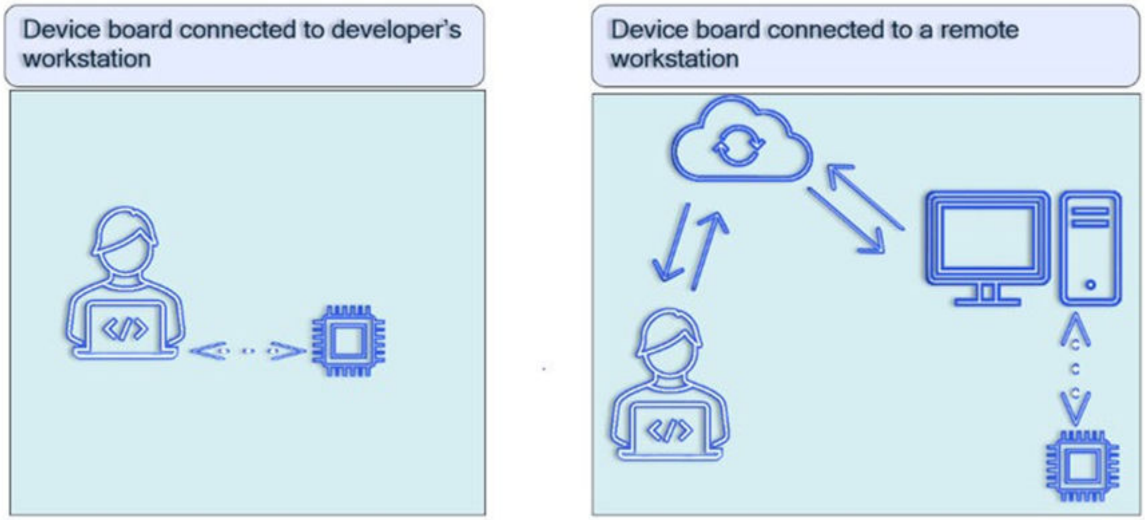
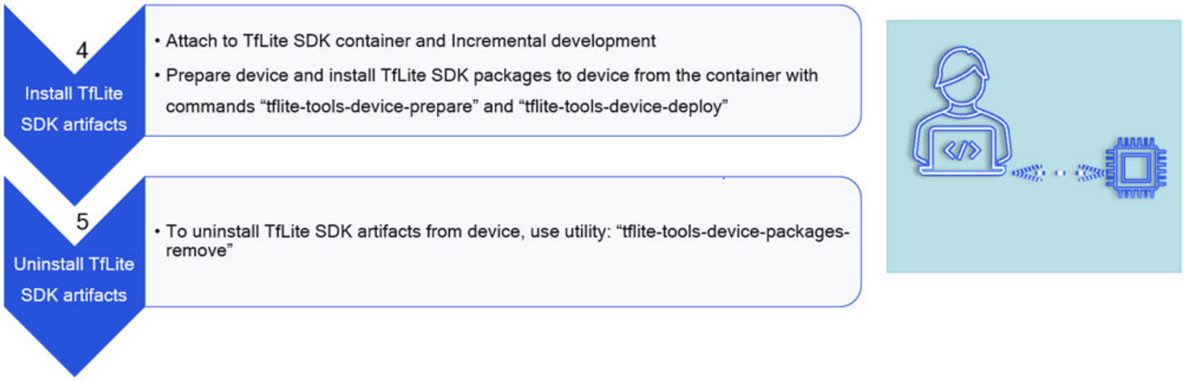

Device board connected to developer's workstation
</>
Device board connected to a remote workstation
</>

Figure 4-4 Connection of device board to developer and remote workstation

4.2.1 Connect device to workstation

The device is connected to the workstation and the development container can access the device over USB/adb.

The figure shows the stages in the sequence of the Qualcomm TFLite SDK workflow:



- Attach to TfLite SDK container and Incremental development
- Prepare device and install TfLite SDK packages to device from the container with commands “tflite-tools-device-prepare” and “tflite-tools-device-deploy”

- To uninstall TfLite SDK artifacts from device, use utility: “tflite-tools-device-packages-remove”

1. Run the following commands to install the artifacts to the device:

```
$ tflite-tools-device-prepare
$ tflite-tools-device-deploy
```

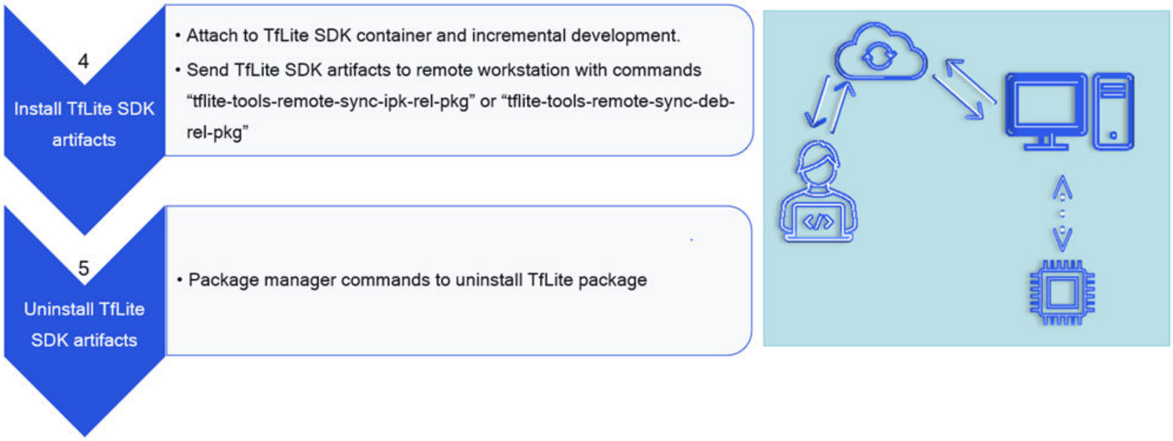
2. To uninstall the artifacts, run the following command:

```
$ tflite-tools-device-packages-remove
```

4.2.2 Connect device to remote machine

The device is connected to a remote machine, and the Qualcomm TFLite SDK container cannot access the device over USB/adb.

The figure shows the stages in the sequence of the Qualcomm TFLite SDK workflow:



- Attach to TfLite SDK container and incremental development.
- Send TfLite SDK artifacts to remote workstation with commands “tflite-tools-remote-sync-ipk-rel-pkg” or “tflite-tools-remote-sync-deb-rel-pkg”

- Package manager commands to uninstall TfLite package

Run the following commands in the tflite-tools container to copy the artifacts to a remote machine depending on the package manager on the device:

```
$ tflite-tools-remote-sync-ipk-rel-pkg
```

NOTE The remote machine information is provided in the JSON configuration file.

Install artifacts for Windows platform

The Qualcomm TFLite SDK artifacts can be installed on the device based on the operating system of the remote machine.

For the Windows platform, do the following:

On PowerShell, use the following script:

```
PS C:> adb root
PS C:> adb disable-verity PS C:> adb reboot
PS C:> adb wait-for-device PS C:> adb root
PS C:> adb remount
PS C:> adb shell mount -o remount,rw /
PS C:> adb shell "mkdir -p /tmp"
PS C:> adb push <tflite package> /tmp
```

If the package is an ipk (for QCS8550.LE.1.0), use the following commands:

```
PS C:> adb shell "opkg --force-depends --force-reinstall --force-overwrite
install /tmp/<tflite package>"
```

Install artifacts for Linux platform

Use the following commands:

```
$ adb root
$ adb disable-verity
$ adb reboot
$ adb wait-for-device
$ adb root
$ adb remount
$ adb shell mount -o remount,rw /
$ adb shell "mkdir -p /tmp"
$ adb push <tflite package> /tmp
```

If the package is an ipk (for QCS8550.LE.1.0):

```
$ adb shell "opkg --force-depends --force-reinstall --force-overwrite
install /tmp/<tflite package>"
```

4.3 Clean up docker image

After completing the developer workflow, the docker environment should be cleaned to free up the storage on the disk. Cleaning the docker removes the unused containers and images, thus freeing up the disk space.

Use the following commands to clean up the docker image:

1. Run the following command on the Linux workstation:

   ```
   $ cd <tflite-sdk-workspace>/tflite-tools
   ```

2. Stop the container:

   ```
   $ tflite-tools-host-stop-container ./targets/ le-tflite-tools-builder.json
   ```

3. Remove the container:

   ```
   $ tflite-tools-host-rm-container ./targets/ le-tflite-tools-builder.json
   ```

4. Remove the older docker images:

   ```
   $ tflite-tools-host-images-cleanup
   ```

4.4 Troubleshoot docker setup

If the `tflite-tools-host-build-image` command returns a `Nospace left on device` message, then move the docker directory to`/local/mnt`.

Do the following to troubleshoot the setup:

1. Back up the existing docker files:

   ```
   $ tar -zcC /var/lib docker > /mnt/pd0/var_lib_docker-backup-$(date +
   %s).tar.gz
   ```

2. Stop the docker:

   ```
   $ service docker stop
   ```

3. Verify that no docker process is running:

   ```
   $ ps faux | grep docker
   ```

4. Check the docker directory structure:

   ```
   $ sudo ls /var/lib/docker/
   ```

5. Move the docker directory to a new partition:

   ```
   $ mv /var/lib/docker /local/mnt/docker
   ```

6. Make a symlink to the docker directory in the new partition:

   ```
   $ ln -s /local/mnt/docker /var/lib/docker
   ```

7. Ensure that the docker directory structure remains unchanged:

   ```
   $ sudo ls /var/lib/docker/
   ```

8. Start docker:

```
$ service docker start
```

9. Restart all the containers after moving the docker directory.

4.5 Generate TFLite SDK with Linux workstation

The TFLite SDK workflow can be enabled without containers using the Linux workstation. This procedure is an alternative to using containers.

To sync and build the Qualcomm TFLite SDK, do the following:

1. Create a directory on the host file system to sync the Qualcomm TFLite SDK workspace. For example:

```
$mkdir <tflite-sdk-workspace>
$cd <tflite-sdk-workspace>
```

2. Fetch the Qualcomm TFLite SDK source code from CodeLinaro:

```
$ repo init -u https://git.codelinaro.org/clo/le/sdktflite/tflite/
manifest.git --repo-branch=qc/stable --repo-url=git://git.quicinc.com/
tools/repo.git -m TFLITE.SDK.1.0.r1-00200-TFLITE.0.xml -b release && repo
sync -qc --no-tags -j8 &&
repo sync -qc --no-tags -j8
```

3. 3. Edit the JSON configuration file present in `<tflite-sdk-workspace>/tflite-tools/targets/le-tflite-tools-builder.json` with the following entries:

```
{
    "Image": "tflite-tools-builder",
    "Device_OS": "le",
    "Additional_tag": "",
    "TFLite_Version": "2.11.1",
    "Delegates": {
        "Hexagon_delegate": "OFF",
        "Gpu_delegate": "ON",
        "Xnnpack_delegate": "ON"
    },
    "TFLite_rsync_destination": "<not applicable>",
    "SDK_path": "<path-to-workspace>/build-qti-distro-fullstack-perf/tmp-
glibc/deploy/sdk>",
    "SDK_shell_file": "<sdk-shell-filename not containing *-ext-*>",
    "Base_Dir_Location": "<Absolute path to TfLiteSDK sync directory>"
}
```

For more information on the entries mentioned in the json configuration file, see the `Docker.md` readme file at `<tflite-sdk-workspace>/tflite-tools/`.

NOTE For QCS8550, Hexagon DSP delegate is not supported.

4. Source the script to set up the environment:

```
$ cd <tflite-sdk-workspace>/tflite-tools
$ source ./scripts/host/host_env_setup.sh
```

5. Build the Qualcomm TFLite SDK.

   ```
   $ tflite-tools-setup targets/le-tflite-tools-builder.json
   ```

6. Run the following utility commands in the same Linux shell to collect the TFLite SDK artifacts from `TFLite_rsync_destination`.

   ```
   $ tflite-tools-host-get-rel-package targets/le-tflite-tools-builder.json
   $ tflite-tools-host-get-dev-package targets/le-tflite-tools-builder.json
   ```

7. Install artifacts based on the operating system.

   - For the Windows platform, on PowerShell, use the following script:

     ```
     PS C:> adb root
     PS C:> adb disable-verity PS C:> adb reboot
     PS C:> adb wait-for-device PS C:> adb root
     PS C:> adb remount
     PS C:> adb shell mount -o remount,rw /
     PS C:> adb shell “mkdir -p /tmp”
     PS C:> adb push <tflite package> /tmp
     ```

     If the package is an ipk (for QCS8550.LE.1.0), use the following commands:

     ```
     PS C:> adb shell “opkg --force-depends --force-reinstall --force-overwrite install /tmp/<tflite package>”
     ```

   - For the Linux platform, use the following script:

     ```
     $ adb root
     $ adb disable-verity
     $ adb reboot
     $ adb wait-for-device
     $ adb root
     $ adb remount
     $ adb shell mount -o remount,rw /
     $ adb shell “mkdir -p /tmp”
     $ adb push <tflite package> /tmp
     ```

     If the package is an ipk (for QCS8550.LE.1.0):

     ```
     $ adb shell “opkg --force-depends --force-reinstall --force-overwrite install /tmp/<tflite package>”
     ```

4.6 Generate Qualcomm TFLite SDK artifacts for QIM SDK build

To use the artifacts generated to enable the Qualcomm TFLite SDK GStreamer plug-in in QIM SDK, do the following:

1. Complete the procedure in Sync and build Qualcomm TFLite SDK, and then run the following command:

   ```
   $ tflite-tools-host-get-dev-tar-package ./targets/le-tflite-tools-builder.json
   ```

   A tar file is generated. It contains the Qualcomm TFLite SDK at the path provided at `“TFLite_rsync_destination”`

2. To enable the Qualcomm TFLite SDK GStreamer plug-in, use the tar file as an argument in the JSON configuration file for the QIM SDK build.

   For information on compiling QIM SDK, see *Qualcomm Intelligent Multimedia SDK (QIM SDK) Quick Start Guide* (80-50450-51).

5 Build Qualcomm TFLite SDK incrementally

If you are building the Qualcomm TFLite SDK for the first time, see Build Qualcomm TFLite SDK tools – developer workflow. The same build environment can be reused for incremental development.

The helper utilities (within the container) mentioned in the figure are available to developers to compile modified applications and plug-ins.

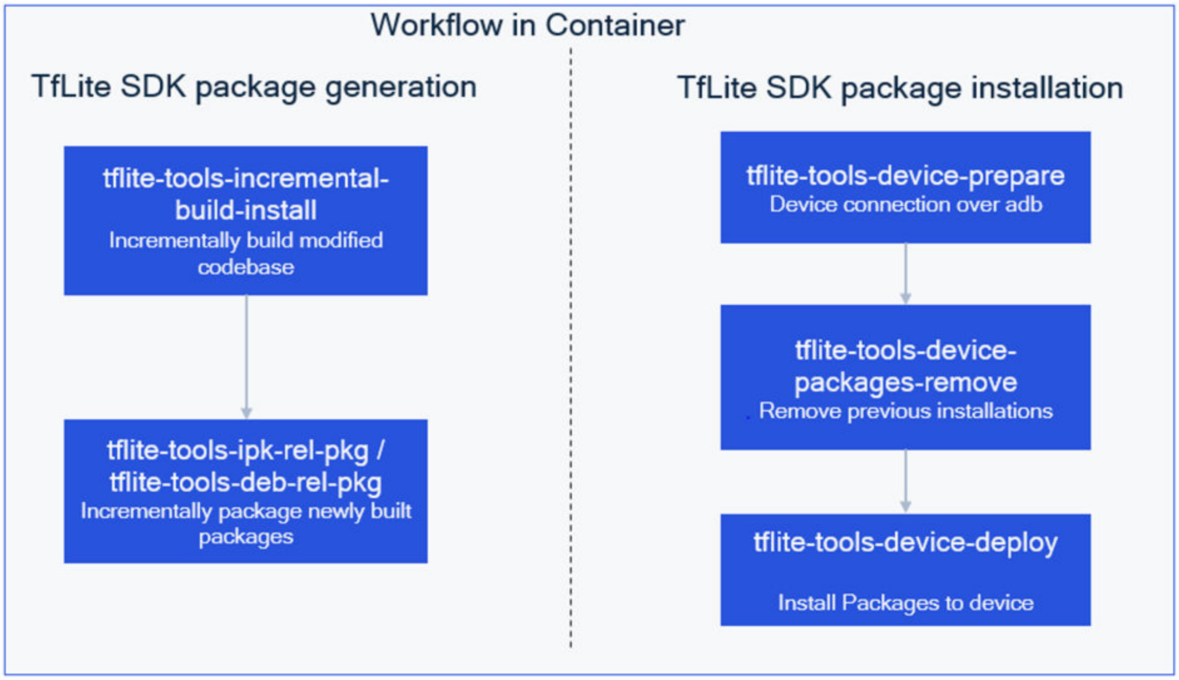

Workflow in Container
TfLite SDK package generation
tflite-tools-incremental-build-install
Incrementally build modified codebase
tflite-tools-ipk-rel-pkg / tflite-tools-deb-rel-pkg
Incrementally package newly built packages
TfLite SDK package installation
tflite-tools-device-prepare
Device connection over adb
tflite-tools-device-packages-remove
Remove previous installations
tflite-tools-device-deploy
Install Packages to device

Figure 5-1 Workflow in a container

After the code changes are completed in the code directory, do the following:

1. Compile modified code:

```
$ tflite-tools-incremental-build-install
```

2. Package compiled code:

```
$ tflite-tools-ipk-rel-pkg
```

   or

```
$ tflite-tools-deb-rel-pkg
```

3. Sync release packages with the host file system:

```
$ tflite-tools-remote-sync-ipk-rel-pkg
```

Or

```
$ tflite-tools-remote-sync-deb-rel-pkg
```

4. Prepare a dev package:

```
$   tflite-tools-ipk-dev-pkg
```

The compiled artifacts are found at in the `TFLite_rsync_destination` folder mentioned in the JSON file, which can be copied to any directory.

6 Work with QNN external TFLite Delegate

A TFLite External Delegate allows you to run your models (part or whole) on another executor using libraries provided by a trusted third party like QNN by Qualcomm. This mechanism can leverage a variety of on-device accelerators such as the GPU or Hexagon Tensor Processor (HTP) for inference. This provides developers a flexible and decoupled method from the default TFLite to speed up inference.

Prerequisites:

- Ensure that you use an Ubuntu workstation to extract QNN AI stack.
- Ensure that you use a QNN version 2.14 to be in conjunction with Qualcomm TFLite SDK

The Qualcomm TFLite SDK is enabled to run inferences on several QNN back-ends through TFLite external Delegate for QNN. The TFLite models with a common `flatbuffer` representation can be run on GPU and HTP.

After the Qualcomm TFLite SDK packages are installed on the device, do the following to install the QNN libraries on the device.

1. Download Qualcomm Package Manager 3 for Ubuntu.
   a. Clickhttps://qpm.qualcomm.com/, and click **Tools**.
   b. In the left pane, in the **Search Tools** field, type `QPM`. From the **System OS** list, select **Linux**.

      The search results display a list of Qualcomm Package Managers.
   c. Select Qualcomm Package Manager 3 and download the Linux debian package.
2. Install Qualcomm Package Manager 3 for Linux. Use the following command:

```
$ dpkg -i --force-overwrite /path/to/
QualcommPackageManager3.3.0.83.1.Linux-x86.deb
```

3. Download the Qualcomm® AI Engine Direct SDK on the Ubuntu workstation.
   a. Click https://qpm.qualcomm.com/ and click **Tools**.
   b. In the left pane, in the **Search Tools** field, type `AI stack`. From the **System OS** list, select **Linux**.

      A drop-down list containing various AI stack engines is displayed.
   c. Click Qualcomm® AI Engine Direct SDK and download the Linux v2.14.0 package.
4. Install Qualcomm® AI Engine Direct SDK on the Ubuntu workstation.
   a. Activate the license:

```
qpm-cli --license-activate qualcomm_ai_engine_direct
```

b. Install AI Engine Direct SDK:

```
$ qpm-cli --extract /path/to/
qualcomm_ai_engine_direct.2.14.0.230828.Linux-AnyCPU.qik
```

5. Push libraries to the device from the Ubuntu workstation with `adb push`.

```
$ cd /opt/qcom/aistack/qnn/2.14.0.230828
$ adb push ./lib/aarch64-oe-linux-gcc11.2/
libQnnDsp.so              /usr/lib/
$ adb push ./lib/aarch64-oe-linux-gcc11.2/
libQnnDspV66Stub.so       /usr/lib/
$ adb push ./lib/aarch64-oe-linux-gcc11.2/
libQnnGpu.so              /usr/lib/
$ adb push ./lib/aarch64-oe-linux-gcc11.2/
libQnnHtpPrepare.so       /usr/lib/
$ adb push ./lib/aarch64-oe-linux-gcc11.2/
libQnnHtp.so              /usr/lib/
$ adb push ./lib/aarch64-oe-linux-gcc11.2/
libQnnHtpV68Stub.so       /usr/lib/
$ adb push ./lib/aarch64-oe-linux-gcc11.2/
libQnnSaver.so            /usr/lib/
$ adb push ./lib/aarch64-oe-linux-gcc11.2/
libQnnSystem.so           /usr/lib/
$ adb push ./lib/aarch64-oe-linux-gcc11.2/
libQnnTFLiteDelegate.so   /usr/lib/
$ adb push ./lib/hexagon-v65/unsigned/
libQnnDspV65Skel.so           /usr/lib/rfsa/adsp
$ adb push ./lib/hexagon-v66/unsigned/
libQnnDspV66Skel.so           /usr/lib/rfsa/adsp
$ adb push ./lib/hexagon-v68/unsigned/
libQnnHtpV68Skel.so           /usr/lib/rfsa/adsp
$ adb push ./lib/hexagon-v69/unsigned/
libQnnHtpV69Skel.so           /usr/lib/rfsa/adsp
$ adb push ./lib/hexagon-v73/unsigned/
libQnnHtpV73Skel.so           /usr/lib/rfsa/adsp
```

7 Test Qualcomm TFLite SDK

The Qualcomm TFLite SDK provides certain example applications, which can be used to validate, benchmark, and get the accuracy of the models that a developer wants to assess.

After the Qualcomm TFLite SDK packages are installed on the device, the runtime is available on the device to run these example applications.

Prerequisite

Create the following directories on the device:

```
$ adb shell "mkdir /data/Models"
$ adb shell "mkdir /data/Lables"
$ adb shell "mkdir /data/profiling"
```

7.1 Label image

A label image is a utility provided by the Qualcomm TFLite SDK that shows how you can load a pre-trained and converted TensorFlow Lite model and use it to recognize objects in images.

Prerequisites:

Download sample model and image:

You can use any compatible model, but the following MobileNet v1 model offers a good demonstration of a model trained to recognize a 1000 different objects.

- Get model

```
$ curl   https://storage.googleapis.com/download.tensorflow.org/models/
mobilenet_v1_2018_02_22/mobilenet_v1_1.0_224.tgz | tar xzv -C /data
$ mv /data/mobilenet_v1_1.0_224.tflite /data/Models/
```

- Get labels

```
$ curl https://storage.googleapis.com/download.tensorflow.org/models/
mobilenet_v1_1.0_224_frozen.tgz  | tar xzv -C /data  mobilenet_v1_1.0_224/
labels.txt

$ mv /data/mobilenet_v1_1.0_224/labels.txt /data/Labels/
```

After you connect to the Qualcomm TFLite SDK docker container, the image can be found at: `“/mnt/tflite/src/tensorflow/tensorflow/lite/examples/label_image/testdata/grace_hopper.bmp”`

a. Push this file to`/data/Labels/`

b. Run the command:

```
$ adb shell "label_image -l /data/Labels/labels.txt -i /data/Labels/
grace_hopper.bmp -m /data/Models/mobilenet_v1_1.0_224.tflite -c 10 -j 1
-p 1"
```

7.2 Benchmark

The Qualcomm TFLite SDK provides the benchmarking tool to calculate the performance of various run times.

These benchmark tools currently measure and calculate statistics for the following important performance metrics:

- Initialization time
- Inference time of warm-up state
- Inference time of steady state
- Memory usage during initialization time
- Overall memory usage

Prerequisites

Push the models to be tested from TFLite Model Zoo (https://tfhub.dev/) to`/data/Models/`. Run the following scripts:

- XNN Pack

```
$ adb shell "benchmark_model --graph=/data/Models/<model file> --
enable_op_profiling=true --use_xnnpack=true --num_threads=4 --max_secs=300
--profiling_output_csv_file=/data/profiling/<csv file to dump data>"
```

- GPU Delegate

```
$ adb shell "benchmark_model --graph=/data/Models/<model file> --
enable_op_profiling=true --use_gpu=true --num_runs=100 --warmup_runs=10 --
max_secs=300 --profiling_output_csv_file=/data/profiling/<csv file to dump
data>"
```

- External Delegate
  - QNN External Delegate GPU:

    Run inference with floating point model:

    ```
    $  adb shell-command "benchmark_model --graph=/data/Models/
    <model_fp32>.tflite --external_delegate_path=libQnnTFLiteDelegate.so --
    external_delegate_options='backend_type:gpu;library_path:/usr/lib/
    libQnnGpu.so;skel_library_dir:/usr/lib/rfsa/adsp'"
    ```

  - QNN External Delegate HTP:

    Run inference with quant model:

    ```
    $ adb shell-command "benchmark_model --graph=/data/Models/
    <model_quant>.tflite --external_delegate_path=libQnnTFLiteDelegate.so --
    external_delegate_options='backend_type:htp;library_path:/usr/lib/
    libQnnHtp.so;skel_library_dir:/usr/lib/rfsa/adsp'"
    ```

7.3 Accuracy tool

The Qualcomm TFLite SDK provides an accuracy tool to calculate the accuracy of models with various run-times.

- Classification with GPU delegate

  The steps to download the necessary files to test can be found at:

  ```
  "/mnt/tflite/src/tensorflow/tensorflow/lite/tools/evaluation/tasks/
  imagenet_image_classificatio/README.md"
  ```

  The binary for running this tool is already part of the SDK, so the developer does not need to build it again.

  ```
  $ adb shell "image_classify_run_eval -- model_file=/data/Models/<Model
  file> --ground_truth_images_path=/data/<Ground truth images path> --
  ground_truth_labels=/data/<ground truth lables> --model_output_labels=/
  data/<labels file> --delegate=gpu"
  ```

- Object detection with XNN pack

  ```
  $ adb shell "inf_diff_run_eval --model_file=/data/Models/<TFLite Object
  Detection Model> --delegate=xnnpack"
  ```

LEGAL INFORMATION

Your access to and use of this document, along with any specifications, reference board files, drawings, diagnostics and other information contained herein (collectively this "Documentation"), is subject to your (including the corporation or other legal entity you represent, collectively "You" or "Your") acceptance of the terms and conditions ("Terms of Use") set forth below. If You do not agree to these Terms of Use, you may not use this Documentation and shall immediately destroy any copy thereof.

1) Legal Notice.

This Documentation is being made available to You solely for Your internal use with those products and service offerings of Qualcomm Technologies, Inc. ("Qualcomm Technologies") and its affiliates described in this Documentation, and shall not be used for any other purposes. This Documentation may not be altered, edited, or modified in any way without Qualcomm Technologies' prior written approval. Unauthorized use or disclosure of this Documentation or the information contained herein is strictly prohibited, and You agree to indemnify Qualcomm Technologies, its affiliates and licensors for any damages or losses suffered by Qualcomm Technologies, its affiliates and licensors for any such unauthorized uses or disclosures of this Documentation, in whole or part.

Qualcomm Technologies, its affiliates and licensors retain all rights and ownership in and to this Documentation. No license to any trademark, patent, copyright, mask work protection right or any other intellectual property right is either granted or implied by this Documentation or any information disclosed herein, including, but not limited to, any license to make, use, import or sell any product, service or technology offering embodying any of the information in this Documentation.

THIS DOCUMENTATION IS BEING PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESSED, IMPLIED, STATUTORY OR OTHERWISE. TO THE MAXIMUM EXTENT PERMITTED BY LAW, QUALCOMM TECHNOLOGIES, ITS AFFILIATES AND LICENSORS SPECIFICALLY DISCLAIM ALL WARRANTIES OF TITLE, MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, SATISFACTORY QUALITY, COMPLETENESS OR ACCURACY, AND ALL WARRANTIES ARISING OUT OF TRADE USAGE OR OUT OF A COURSE OF DEALING OR COURSE OF PERFORMANCE. MOREOVER, NEITHER QUALCOMM TECHNOLOGIES, NOR ANY OF ITS AFFILIATES OR LICENSORS, SHALL BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY EXPENSES, LOSSES, USE, OR ACTIONS HOWSOEVER INCURRED OR UNDERTAKEN BY YOU IN RELIANCE ON THIS DOCUMENTATION.

Certain product kits, tools and materials referenced in this Documentation may require You to accept additional terms and conditions before accessing or using those items.

Technical data specified in this Documentation may be subject to U.S. and other applicable export control laws. Transmission contrary to U.S. and any other applicable law is strictly prohibited.

Nothing in this Documentation is an offer to sell any of the components or devices referenced herein.

This Documentation is subject to change without further notification.

In the event of a conflict between these Terms of Use and the *Website Terms of Use* on www.qualcomm.com or the *Qualcomm Privacy Policy* referenced on www.qualcomm.com, these Terms of Use will control. In the event of a conflict between these Terms of Use and any other agreement (written or click-through) executed by You and Qualcomm Technologies or a Qualcomm Technologies affiliate with respect to Your access to and use of this Documentation, the other agreement will control.

These Terms of Use shall be governed by and construed and enforced in accordance with the laws of the State of California, excluding the U.N. Convention on International Sale of Goods, without regard to conflict of laws principles. Any dispute, claim or controversy arising out of or relating to these Terms of Use, or the breach or validity hereof, shall be adjudicated only by a court of competent jurisdiction in the county of San Diego, State of California, and You hereby consent to the personal jurisdiction of such courts for that purpose.

2) Trademark and Product Attribution Statements.

Qualcomm is a trademark or registered trademark of Qualcomm Incorporated. Arm is a registered trademark of Arm Limited (or its subsidiaries) in the U.S. and/or elsewhere. The Bluetooth® word mark is a registered trademark owned by Bluetooth SIG, Inc. Other product and brand names referenced in this Documentation may be trademarks or registered trademarks of their respective owners.

Snapdragon and Qualcomm branded products referenced in this Documentation are products of Qualcomm Technologies, Inc. and/or its subsidiaries. Qualcomm patented technologies are licensed by Qualcomm Incorporated.