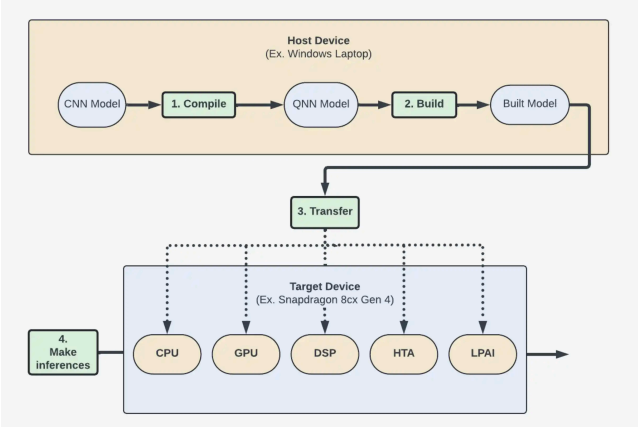
CNN to QNN for Linux Host

Updated: Jan 06, 2026 80-63442-10 Rev: AE

This guide will teach you how to convert your CNN model into an executable that can be run on a target device's processors using **Qualcomm AI Engine Direct (aka the QNN SDK)**.

In order to do that, you will learn how to:

1. Convert your Convolutional Neural Net (CNN) model to a Qualcomm Neural Net (QNN) Model.
2. Build that model for a specific target device operating system. (Ex. Android)
3. Transfer and use the model to make inferences on the desired processing unit. (Ex. GPU)



Step 1: Tutorial Setup

Install the QNN SDK

1. Follow the instructions in Setup to install the QNN SDK. 1. Make sure to install the optional Tensorflow dependency as this tutorial will use a Tensorflow model. (See Step 3 in Setup for more instructions).

> **Note**
> Using the same terminal for the Setup and these steps will speed up the process as some necessary setup steps only affect the terminal's environment variables.

2. **Check that `QNN_SDK_ROOT` is set to the folder just inside `qairt` by running `$QNN_SDK_ROOT`.**
    1. You should see the path to the folder name inside `qairt` (Ex. `.../qairt/2.22.6.240515`)
    2. If `QNN_SDK_ROOT` is not set: 1. Navigate to `qairt/<QNN_SDK_ROOT_LOCATION>/bin`
        2. **Run `source ./envsetup.sh` to set the environment variable.**
            1. Note: These changes will only apply to the current terminal instance.
3. Ensure you are in the proper virtual environment for Python. 1. If you are not in a `venv`, see Step 2 of Setup to install / activate your environment.

Set Up The Example Tensorflow Model

1. **Run `python3 -m pip show tensorflow` to verify that `tensorflow` is installed properly.**
    1. If `tensorflow` is not found, follow the steps in Step 3 of the Setup instructions to install Tensorflow.
2. Run the following 3 commands below to set the `TENSORFLOW_HOME` environment variable.

```
tensorflowLocation=$(python -m pip show tensorflow | grep '^Location: ' | aw
export TENSORFLOW_HOME="$tensorflowLocation/tensorflow"
echo "export TENSORFLOW_HOME=$tensorflowLocation/tensorflow" >> ~/.profile
```

> **Note**
> `TENSORFLOW_HOME` is needed because the `setup_inceptionv3.py` script uses TensorFlow utilities like `optimize_for_inference.py`, which are present in the TensorFlow installation directory.

3. Run `${TENSORFLOW_HOME}` to verify that the environment variable was set properly (it should be a directory).
4. Run `python3 ${QNN_SDK_ROOT}/examples/Models/InceptionV3/scripts/setup_inceptionv3.py -a ~/tmpdir -d`.

   This will create the test data for this tutorial.

   1. A model file at: `${QNN_SDK_ROOT}/examples/Models/InceptionV3/tensorflow/inception_v3_2016_08_28_frozen.pb` 2. Raw images at: `${QNN_SDK_ROOT}/examples/Models/InceptionV3/data/cropped`

> **⚠ Warning**
> If you would like to skip the tutorial steps for building the models for a target device, run the above script with the `-c` flag.

5. Run `ls ${QNN_SDK_ROOT}/examples/Models/InceptionV3/tensorflow/` to verify that the `inception_v3_2016_08_28_frozen.pb` model file exists.

6. Run `ls ${QNN_SDK_ROOT}/examples/Models/InceptionV3/data/cropped` to verify that there is a collection of image files such as `plastic_cup.raw` or `chairs.jpg`.

Step 2: Converting the CNN model into a QNN model

Converting models into QNN format allows them to be built for specific target device operating systems and processors.

This tutorial is using a TensorFlow model, so we can convert by running the `qnn-tensorflow-converter` tool. If you are using another type of model, you can look at the Tools page for a table of potential scripts to help convert them into QNN format. They will have a similar `qnn-model-type-converter` naming convention.

You can use the QNN SDK to convert either full precision models or quantized models by following the below steps.

> ⚠ Warning
> HTP and DSP target devices MUST use quantized models with the `--input_list` param.

Full Precision Model Conversion

1. Run the following script:

```
python ${QNN_SDK_ROOT}/bin/x86_64-linux-clang/qnn-tensorflow-converter \
    --input_network "${QNN_SDK_ROOT}/examples/Models/InceptionV3/tensorflow/
    --input_dim input 1,299,299,3 \
    --out_node "InceptionV3/Predictions/Reshape_1" \
--output_path "${QNN_SDK_ROOT}/examples/Models/InceptionV3/model/Inception_v
```

The flags are used to specify:

- `--input_network` - The path to the source framework model.
- **`--input_dim` - The input name followed by the dimensions of that input.**
  - `--out_node` - The name of the graph's output Tensor Names. Multiple output names should be provided separately like:
  - `--output_path` - This indicates where to put the output files.

> ⓘ Note
> `qnn-tensorflow-converter` supports a wide variety of input types. Run the script with the `--help` flag to see all options available to you.

2. **Run `ls ${QNN_SDK_ROOT}/examples/Models/InceptionV3/model` to see the following artifacts produced by the script:**
   - `Inception_v3.cpp` - This contains the sequence of API calls.

- `Inception_v3.bin` - This has the static data associated with the model and will be called by the *.cpp*.
- `Inception_v3_net.json` - This describes the model data in a standardized manner. (This is NOT needed to build the QNN model later on)

Quantized Model Conversion

To use a quantized model instead of a floating point model, you will need to pass in the `--input_list` flag to specify the input.

1. In a real situation, you would need to prepare your input data before running this command. For this tutorial, that data has already been prepared in `${QNN_SDK_ROOT}/examples/Models/InceptionV3/data/cropped/raw_list.txt`. **(No action required)**
2. Run the below command:

```
python ${QNN_SDK_ROOT}/bin/x86_64-linux-clang/qnn-tensorflow-converter \
  --input_network "${QNN_SDK_ROOT}/examples/Models/InceptionV3/tensorflow/incepi
  --input_dim input 1,299,299,3 \
  --input_list "${QNN_SDK_ROOT}/examples/Models/InceptionV3/data/cropped/raw_lis
  --out_node "InceptionV3/Predictions/Reshape_1" \
  --output_path "${QNN_SDK_ROOT}/examples/Models/InceptionV3/model/Inception_v3.
```

The flags are used to specify:

- `--input_network` - The path to the source framework model.
- `--input_dim` - The input name followed by the dimensions of that input.
- `--input_list` - The **absolute** path to input data. **(Required for quantization)**
- `--out_node` - The name of the graph's output Tensor Names. Multiple output names should be provided separately.
- `--output_path` - This indicates where to put the output files.

The above `qnn-tensorflow-converter` command will produce the following artifacts:

- `Inception_v3.cpp` - This contains the sequence of API calls.
- `Inception_v3.bin` - This has the static data associated with the model.
- `Inception_v3_net.json` - This describes the model data in a standardized manner. (This is not needed to build the QNN model later on)

You can find the artifacts in the `${QNN_SDK_ROOT}/examples/Models/InceptionV3/model` directory.

Step 3: Build the Model for a Target Device

- CNN to QNN for Linux Host on Linux Target
- CNN to QNN for Linux Host on Windows Target

Previous
Tutorial: Converting and executing a CNN model with QNN

Next
CNN to QNN for Linux Host on Linux Target

Note: Certain services and materials may require you to accept additional terms and conditions before accessing or using those items.

References to "Qualcomm" may mean Qualcomm Incorporated, or subsidiaries or business units within the Qualcomm corporate structure, as applicable.

Qualcomm Incorporated includes our licensing business, QTL, and the vast majority of our patent portfolio. Qualcomm Technologies, Inc., a subsidiary of Qualcomm Incorporated, operates, along with its subsidiaries, substantially all of our engineering, research and development functions, and substantially all of our products and services businesses, including our QCT semiconductor business.

Materials that are as of a specific date, including but not limited to press releases, presentations, blog posts and webcasts, may have been superseded by subsequent events or disclosures.

Nothing in these materials is an offer to sell or license any of the services or materials referenced herein.